Mobilszoftverek

Dr. Forstner Bertalan

A tárgy oktatói

- Kelényi Imre
  - kelenyi.imre@aut.bme.hu , QB226
- Ekler Péter
  - ekler.peter@aut.bme.hu , QB226
- Blázovics László
  - blazovics.laszlo@aut.bme.hu , QB236
- Fehér Marcell
  - feher.marcell@aut.bme.hu, QB236

A tárgy helye az AAIT AMORG mobil portfóliójában

1. Bevezetés a mobil szoftverfejlesztésbe
   - 2 kredit
   - Alapok, prototípus-fejlesztés, Python és Java ME
2. Mobilszoftverek
   - 4 kredit
   - 4 platform, gyakorlat és labor
3. Symbian alapú szoftverfejlesztés
   - 4 kredit
   - Symbian OS, okostelefonok programozása
4. Mobil Linux alkalmazások
   - 4 kredit
   - Maemo, natív Android, OpenMoko

Új tárgyak a továbblépéshez (munkacímek):

1. Android alapú szoftverfejlesztés
   - 4 kredit
   - Mindent lefedő. Előadás + labor
2. Az iOS programozása
   - 2 kredit
   - Mindent lefedő.
3. Linux és Qt alapú mobil alkalmazások
   - 4 kredit
   - Beágyazott linux, Natív Android, Symbian OS alapok, Meego alapok, Qt és Mobility API

Következő félévtől

A kurzus elvégzésével…

- Betekintést nyertek a mobil szoftverfejlesztési lehetőségekbe (platformok, képességek, nehézségek…)
- Megismeritek a mobil eszközökre való fejlesztés különlegességeit
- Alapismereteitek lesznek 4 fontos platformról

A félév felépítése

- Az előadások már önmagukban gyakorlatiasak lesznek
- Labor: következő félévben

Előadás #	Dátum	Téma
1	február 9.	bevezetés
2	február 16.	Qt
3	február 23.	Qt
4	március 2.	Qt
5	március 9.	Java ME
6	március 16.	Java ME
7	március 23.	Java ME
8	március 30.	ZH
9	április 6.	iPhone
10	április 13.	iPhone
11	április 20.	iPhone
12	április 27.	Android
13	május 4.	Android
14	május 11.	Android

Gyakorlat #	Dátum	Téma
1	február 17.	Qt
2	március 10.	Java ME
3	március 17.	Java ME
4	április 7.	iPhone
5	április 14.	iPhone
6	április 28.	Android
7	május 5.	Android

Követelmények

- ZH
  - Aláírás feltétele
  - 80% feletti rész vizsgán plusszpont
  - Március 30.
- Vizsga
  - Írásbeli

Honlap, jegyzet

- **http://www.aut.bme.hu/portal/VIAUM125**
  - Hírek, követelmények, minden más…
- Jegyzet
  - ppt☺, mert azt mindenki szereti
  - Remélhetőleg „normál” jegyzet is lesz, legalábbis egyes témakörökhöz
  - Előadásokon mondjuk a szükséges könyveket

Mobil szoftverfejlesztési versenyek

- Állandóan vannak
- Általában a tanszéki hallgatók nyerik ☺
- Eddigi tapasztalat: annyi ilyen van, hogy aki indul, és beadja az alkalmazását, nyer.
- Folyamatosan hirdetjük itt a tárgyon is
- Ha emailt is szeretnél kapni minden lehetőségről, küldj egy emailt „Mobilszoftverek” címmel a forstner.bertalan@aut.bme.hu címre, benne a várható végzésed dátumával.

Az Amorg a Twitteren

- Versenyek, ZH és vizsgaeredmények, linkek, események és sok más hasznos dolog
- Kövesd: bme_amorg_lite

1. előadás - Bevezetés

Tartalom

- Mobilpiac szereplői
- Vezetéknélküli technológiák
  - Mobilhálózatok fejlődése, generációk
  - Rövidtávú rádiós technológiák
- Mobilkészülékek tulajdonságai
- Történelmi áttekintés
- A piac feloszlása
- Cserélhető előlapok

Tartalom

- Mobilpiac szereplői
- Vezetéknélküli technológiák
  - Mobilhálózatok fejlődése, generációk
  - Rövidtávú rádiós technológiák
- Mobilkészülékek tulajdonságai
- Történelmi áttekintés
- A piac felosztása
- Cserélhető előlapok

Akkor miről is lesz szó?

- A mobil eszközök mások
  - Hardver
  - Kapacitás
  - Kapcsolódás
  - Energia kérdések
  - Felhasználói kontextus
- Megszorítások és lehetőségek

Hardver által behatárolt szoftvertervezési megfontolások

- Alacsony számítási teljesítmény
- Korlátozott RAM
- Kis méretű permanens tár
- Kis képernyő, kis felbontással
- Magas adatátviteli költségek
- Alacsonyabb átviteli sebesség, nagyobb késleltetéssel
- Megbízhatatlanabb adatkapcsolat
- Limitált energiaforrás

Tartsuk szem előtt

- Gyors és responzív alkalmazások kellenek
  - Nem bővítik a hardvert a programunk kedvéért…
- A szoftverergonómia hatványozottan fontos
  - Kis képernyő, hogy hordozható maradjon
  - Változatos input és output akár egy platformon belül
    - Touchscreen, trackball, ITU-T 12-keypad, qwerty, akár menet közben változik!
    - Álló, fekvő módok
    - Több képernyő
    - Skálázható UI: Saját grafika? Beépített?

Tartsuk szem előtt 2

- Hatékony kódot a hardverre!
  - Memory leak?
  - A gc is viszi a processzort
  - Temp fájlok a háttértáron
  - Profiling

Tartsuk szem előtt 3

- Limitált kapcsolat
  - Mi van, ha nincs?
  - Mi van, ha közben szűnik meg?
  - Mi van, ha lassú, nagy késleltetéssel? (pl. video)
- Mennyibe is kerül?
  - Adatforgalom, hívás, SMS, MMS…
  - A felhasználó tud róla?
  - Adat cache, ritka update/upgrade…

Tehát amit tanulunk…

- Milyen
  - Tervezési minták
  - UI technikák
  - Op.rendszer szolgáltatások

segítik a hatékony, responzív, ergonomikus, sőt, akár még hasznos alkalmazások fejlesztését a különböző mobil platformokon

AMORG APPLIED MOBILE RESEARCH GROUP

Symbian, Qt

Mobilszoftverek

S1

Tartalom – Symbian, Qt

- 5 előadás / ”gyakorlat”
- Témák
  - A Symbian platform bemutatása (1 előadás)
    - Szoftverfejlesztési lehetőségek
  - Symbian C++ (1 előadás)
    - Alapvető paradigmák, eszközök
    - CleanupStack, ActiveObject, ThinTemplate
  - Qt (3 előadás)
    - Alapfogalmak, Qt object model
    - Qt Mobility
    - Qt Quick

S1 – Tartalom

- A Symbian OS története, legfőbb jellemzői
- Szoftverfejlesztési lehetőségek Symbian-ra
- Natív szoftverfejlesztés Symbian-ra, SDK-k bemutatása
  - Symbian C++
  - Qt
- Natív alkalmazások terjesztése

Symbian

A platform bemutatása

Mobilplatformok piaci részesedése 2010

- Miért is tanulunk a Symbian-ról?

Worldwide Smartphone Sales to End Users by Operating System in 3Q10 (Thousands of Units)

Company	3Q10 Units	3Q10 Market Share (%)	3Q09 Units	3Q09 Market Share (%)
Symbian	29,480.1	36.6	18,314.8	44.6
Android	20,500.0	25.5	1,424.5	3.5
iOS	13,484.4	16.7	7,040.4	17.1
Research In Motion	11,908.3	14.8	8,522.7	20.7
Microsoft Windows Mobile	2,247.9	2.8	3,259.9	7.9
Linux	1,697.1	2.1	1,918.5	4.7
Other OS	1,214.8	1.5	612.5	1.5
Total	**80,532.6**	**100.0**	**41,093.3**	**100.0**

Symbian OS általános jellemzők 1/2

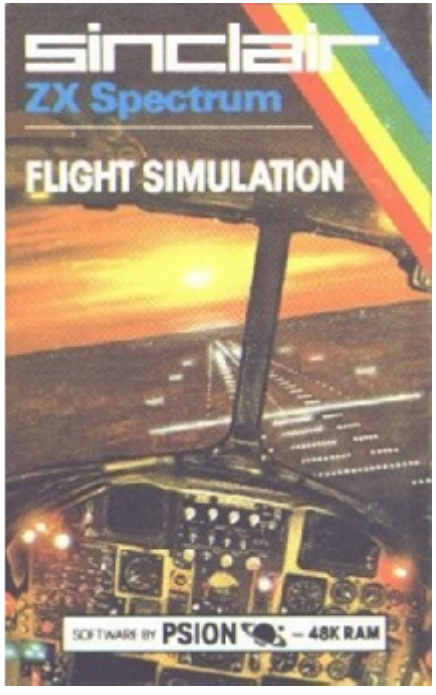
- 3 tervezési alapelv
  - A felhasználói adatok maximális védelme
  - Gyors válaszidő
  - Kevés memória és lassú processzor minél jobb kihasználása
- Kezdetektől fogva mobileszközökre tervezték
- Teljesen objektum orientált
  - C++-ban írták, még a kernelt is
  - Minden API C++
- Az operációs rendszer legtöbb szolgáltatása kliens-szerver architektúrán keresztül elérhető
- Nyílt
  - Külső alkalmazások telepíthetők a rendszerre

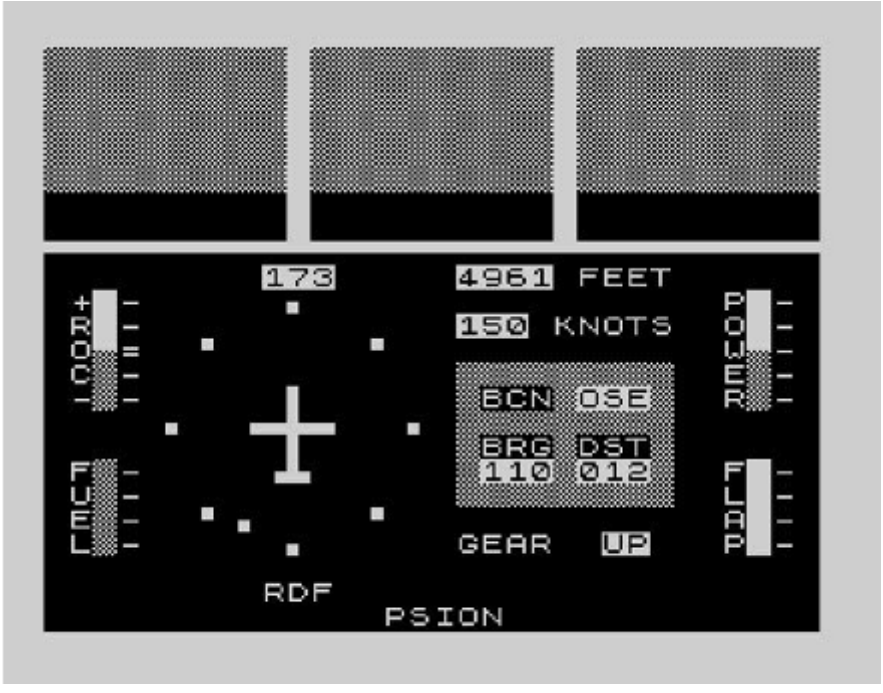
Symbian OS általános jellemzők 2/2

- Magas rendelkezésre állás (ritka reboot)
- Microkernel architektúra, real-time kernel, egyetlen processzorra
  - Alkalmazások és mobiltelefónia stack is egy processzoron (vagy processzormagon) fut
- Moduláris felépítésű
  - Szolgáltatások különálló szerverek, - az alkalmazások kliensek, IPC kommunikáció
- Együtt tervezték az alap alkalmazásokkal
  - pl. személyi adatkezelő funkciók OS szintű támogatása (PIM – Personal Information Manager): naptár, névjegyzék, stb.

Történelem

- 1981 PSION Flight Simulator

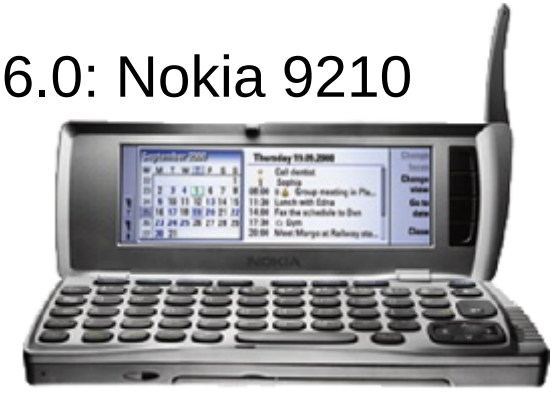




- Játékszoftver ZX Spectrumra és ZX81-re

Történelem

- 1984 PSION Organizer
- 1988 16 bites operációs rendszer: SIBO (*SIxteen Bit Organisers*)
- 1997 32 bites operációs rendszer: EPOC (*Electronic Piece of Cheese)*
- 1998 Symbian: külön társaság, Ericsson, Nokia, Motorola, Psion
- 1999 EPOC r5: Ericsson R380, MC 218. Psion 5mx, Revo, NetBook
- 2000 Symbian OS 6.0: Nokia 9210



Történelem

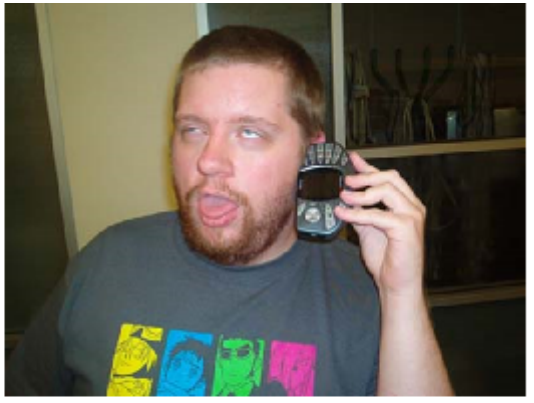
- 2001-2008: Symbian OS 6.1 – Symbian 9.5
  - Nokia 3650, N-Gage, 6600, N91, N70, E60,…
  - Sony Ericsson P800, P900, Nokia 6600




- 2008: 100% Nokia tulajdonba kerül
- 2008-tól új név és számozás: Symbian^1...

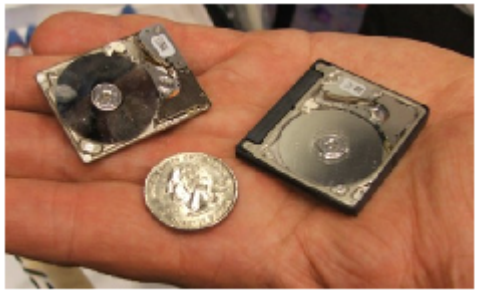
Néhány különös telefon

- Nokia N-Gage
  - ?



- Nokia N92
  - Transformers

- Nokia N91
  - Microdrive



Hardver

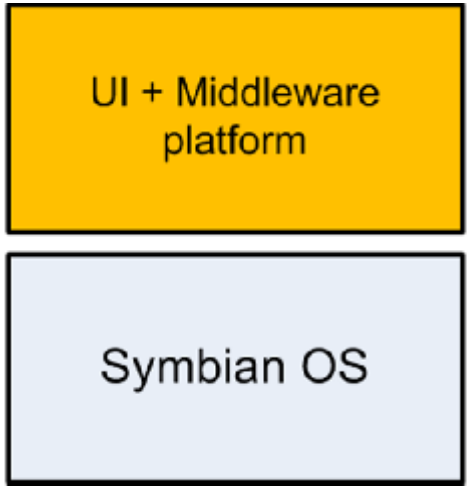
- ARM architektúrájú processzor
  - 32 bit RISC
  - 200 – 700 MHz (jelenleg ARM 9/ARM 11)
- 64-256 MB RAM
- Sok különféle periféria, rádió adó/vevő
  - Bluetooth, WLAN, GSM/3G
- Akkumulátor: 900 - 1500 mAh
- Kijelző felbontás új modelleknél 640x360

Nokia E7 – Symbian^ 3

- 680 MHz ARM 11 CPU, 256 MB RAM
- Érintőképernyő + QWERTY-keyboard
- nHD 16:9 képernyő (640x360, 16 millió színárnyalat, 3.5“)
- 8 MPixel kamera, 720p@25fps video
- A-GPS, WLAN, UMTS, HSDPA, UPnP, USB 2.0
- Radio, 720p HDMI TV-Out, 3.5 mm audio
- 16 GB beépített flash + MicroSD
- Gyorsulásmérő, iránytű, proximity sensor
- 176 g, 123x62x13 mm

UI+ middleware platformok

- A Symbian OS korábban csak az alapját képezte a készülékek operációs rendszerének
- Erre épül rá egy komplex UI+middleware platform
  - S60
  - UIQ
  - Series 80
  - MOAP(S)



S60/UIQ/MOAP

- Definiálják a készülék felhasználói felületét, az alap alkalmazásokat és a fejlesztői eszközöket (SDK)

Kihalt UI platformok

- UIQ
  - Kezdetben csak érintőképernyő, PDA-szerű
  - 3-as verziótól billentyűzet is

- Series 80
  - Nokia Communicator
  - Felváltotta S60 3rd (Nokia E90)

- MOAP(S)
  - Zárt platform, FOMA (NTT DoCoMo, Japán)

S60 (Series 60)

- S60 1st - 2nd edition

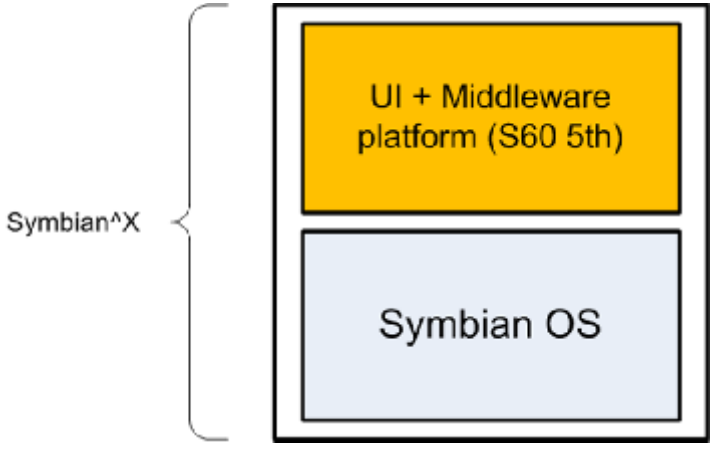
- S60 3rd edition
  - Új real-time kernel (Symbian 8.x-EKA2)
  - Platform security
  - Több képernyőméret
  - Visszafelé nem kompatibilis S60 2nd-el

S60 5th edition

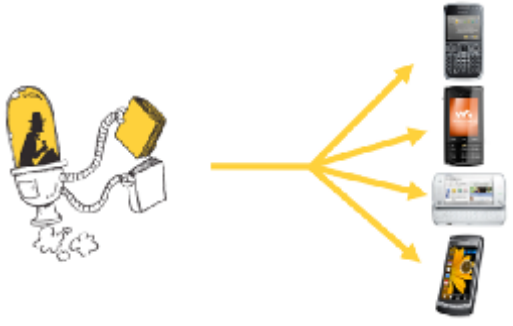
- Touch UI
- „Tactile feedback”
- 2 képernyőméret
  - qVGA 320x240
  - nHD 640 x 360
- Fejlettebb multimédia támogatása
  - Windows Media
  - DRM
- Visszafelé kompatibilis: S60 3rd alkalmazások futtathatók rajta

Az új Symbian platform

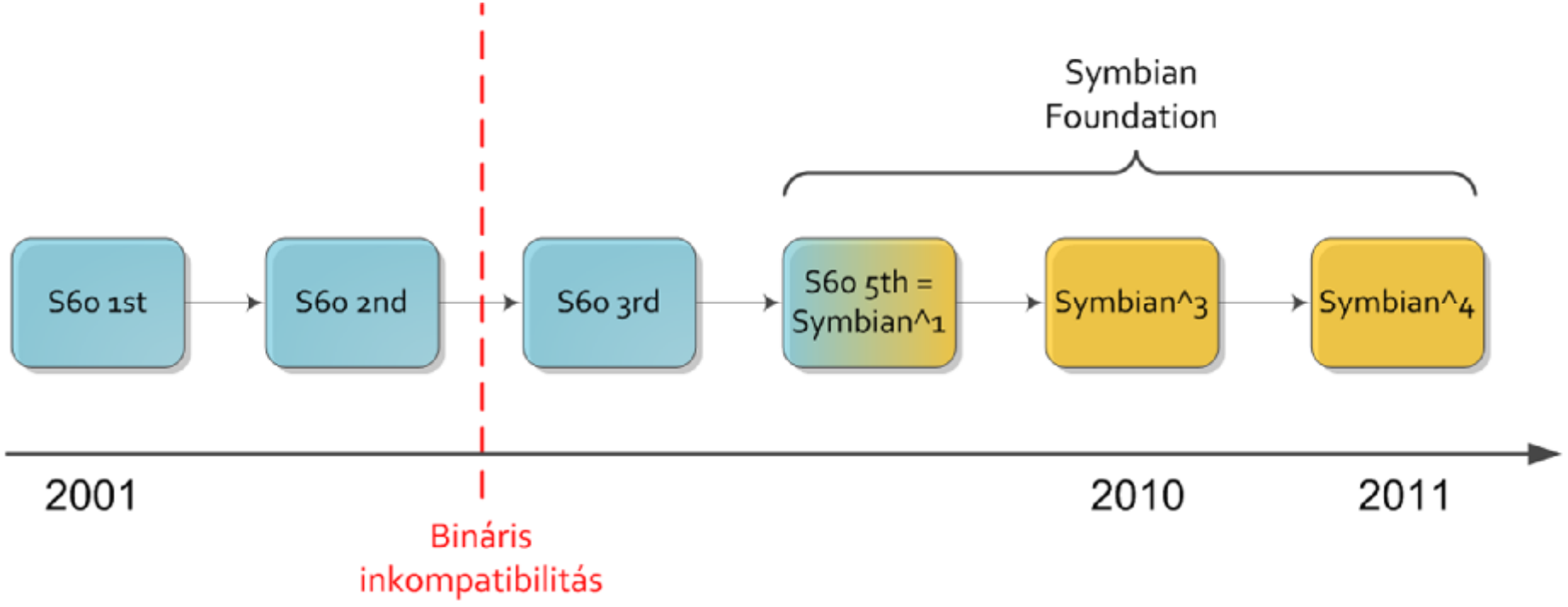
- Egy platform mind felett: „Symbian”
  - Egyesült az S60, a UIQ és a MOAP(S) egyetlen platformmá



- 2010 február: a Symbian open source
  - 2010 december: Nokia újra zárttá teszi forrást
- Egy Symbian-ra írt alkalmazás „minden” Symbian-ra épülő telefonon futni fog (tada.wav)



S60 – Symbian^ X verziók

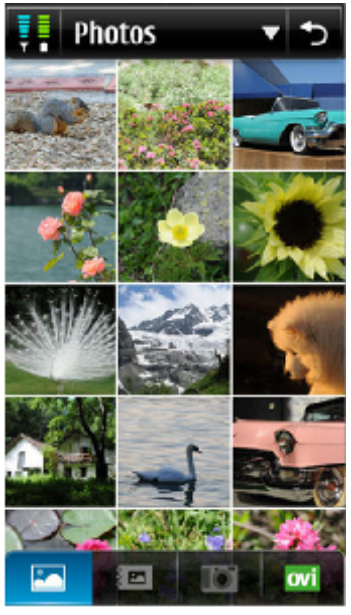


Az új Symbian^ platform

- Symbian^1 - 20
  - = S60 5th (pl. Nokia N97)
- Symbian^2
  - Nem jelent meg (tervezett funkciók patchek formájában megjelentek Symbian^1-hez)
- Symbian^3 - 2010
  - Első készülékek 2010 végén: Nokia N8, C7, E8…
  - Új home screen
  - Gyorsabb grafika
  - HD video támogatás, HDMI kimenet
  - Multi-touch
  - Qt előtelepítve

2011: Symbian^ 4

- 2011 elején érkezik
- Qt az alapértelmezet fejlesztési környezet
- Új UI layout
- Bluetooth Low Energy
- Integrált NFC (Near Field Communication)
- „Continuous evolution” model
  - Korábbi verziók (Symbian^3) frissíthető



Platformok - összefoglalás

- Régebben több szoftverplatform Symbian OS-hez (S60, UIQ, stb.)
- S60 volt magasan a legnépszerűbb platform, nagyon sok Nokia modell
- A mai új telefonok már a „Symbian kalap”-ra épülnek
  - S60 5th-et átkeresztelték Symbian^1-re

Symbian miértek

- Miért lett sikeres?
  - Nyílt platform, bárki fejleszthet rá
  - S60 (Nokia) UI könnyen kezelhető és átlátható
  - Úttörő hardver és funkciók
  - Ma is a legstabilabb, legkisebb hardver igényű mobil operációs rendszer
- Miért csökken a népszerűsége?
  - Mára a UI elavult, későn váltottak érintőképernyőre, csak 2010-ben érkezett a multitouch
  - Böngésző nem a legjobb (sebesség, kompatibilitás)
  - Fejlesztői eszközök, API-k lassan fejlődtek, lassú és néha körülményes szoftvert írni
  - Eddig nem lehetett az új Symbian verziókat feltelepíteni a korábbi telefonokra (egy telefon, egy operációs rendszer)

Ellenőrző kérdések, tippek

- Symbian OS Általános jellemzők
- UI+Middleware platformok múlt és jelen
  - Sok volt régen, egymással nem kompatibilisek, S60 volt a legelterjedtebb, most már Symbian^X van…
  - Miért jó az új „Symbian kalap” platform?

Symbian

Szoftverfejlesztés Symbian-ra

Szoftverfejlesztés Symbian OS-re

- Symbian C++
- Qt, Standard C/C++

Natív kód

- Java ME

Virtuális gépen fut, interpretált

- Python for S60

Szkriptnyelv – protípuskészítésre

- Web Runtime (WRT)

Webes technológiák – widgetek

- Flash Lite
- Silverlight

Szkriptnyelv – prezentáció, multimédia, játékok

Java ME

- CLDC 1.1, MIDP2.0
- Gyors alkalmazásfejlesztés, platformok között kompatibilitás
- Symbian-on a MIDLET futhat a háttérben, sok más telefonon nem
- Alapból fapados UI, külső librarykat lehet használni
- Nem minden funkció érhető el minden mobilon (videó lejátszás, GPS, Bluetooth)
  - Megfelelő JSR-t (Java Specification Request) támogatni kell a mobilnak
  - Bizonyos funkciók (pl. telefónia) csak korlátozottan támogatott

Python for S60 - HelloWorld

- 2 sor kód

```
import appuifw
appuifw.note(u"Hello", 'info')
```

Python for S60

- Alapvetően gyors prototípusfejlesztéshez találták ki
  - Pár soros kóddal is lehet működő programot készíteni
- Támogatja a standard UI look-and-feel-t
- Hozzáférést biztosít számos platformspecifikus funkcióhoz
  - Messaging
  - Telefonfunkciók
  - PIM kezelés
  - Hálózat / HTTP
- A kódot egy futtatókörnyezet futtatja, melyet külön telepíteni kell a telefonra
- Hátrányok:
  - Csak korlátozottan alkalmas nagyobb projektekhez, nincs normális debug, hibakezelés, IDE
  - Python futtatókörnyezetet fel kell telepíteni a telefonra

Web Runtime (WRT)

- Webes technológiákon alapuló alkalmazások (widgetek)
  - HTML, CSS, JavaScript, XML, stb.
- Ugyanúgy jelennek meg mint a standard alkalmazások (nem kell böngésző)
- Platformspecifikus funkciók is elérhetők
  - „platform services”: lokáció, üzenetek, naptár, stb.
- Hátrányok:
  - Sok funkció nem- vagy csak korlátozottan érhető el
  - Lassú

WRT példa

Widget main components

XHTML – tags, elements, events

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Slashbot</title>
  <style>
    @import "weather.css";
  </style>
</head>
<body>
<script type='text/javascript' src="weather.js'></script>
<div id="front" onmousemove='mousemove(event);
                ' onmouseout='mouseexit(event);'>
  <div id="map" class="map">
      <img src="Default.png" width="290" height="300">
  </div>
</div>
<script language="javascript">
</script>
</body>
</html>
```

JavaScript – dynamic update

```
function loadWeather() {
var url = "http://www.weatherbug/weather.rss";
  if (isLoading == 1)  {
    debugOut("loadRSS: aborting");
    return;
  };
  isLoading = 1;
  if (null == req)  {
    req = new XMLHttpRequest();
    req.onreadystatechange = processReqChange;
  }
  var dt = new Date();
  url = url + "?" + dt.getTime();
  req.open("GET", url, true);
  req.send(null);
};
```

CSS – formatting

```
.map
{
 height 100px; width 240px;
 colon orange; curson pointer;
 font-weight: bold; margin-bottom: -15px;
 padding: 0px; xborder 1px solid blue;
}
```

ForumNOKIA

Symbian C++

- Az operációs rendszer natív programozási nyelve
- A C++ egy megkötésekkel teli változata
  - Nem különálló programozási nyelv, sima C++ fordítót használunk
- Irányelvek
  - Minimális memóriahasználat
  - Hatékony kód
  - Memóriaszivárgás kizárása
  - Minél kevesebbe kerülő multitasking
- Hosszabb tanulási idő
  - Egy C++ programozó pár hét alatt belejön
  - Java / C# után nehezebb…

Memória/erőforrás takarékos technológiák

- Kivétel- és memóriakezelés biztonságosan
  - Leave
  - CleanupStack
  - Kétfázisú konstruktor
- C++ templatek által generált többletkód minimalizálása
  - ThinTemplate
- Erőforrásfájlok
- Aszinkron események kezelése egy szálon belül
  - ActiveObject

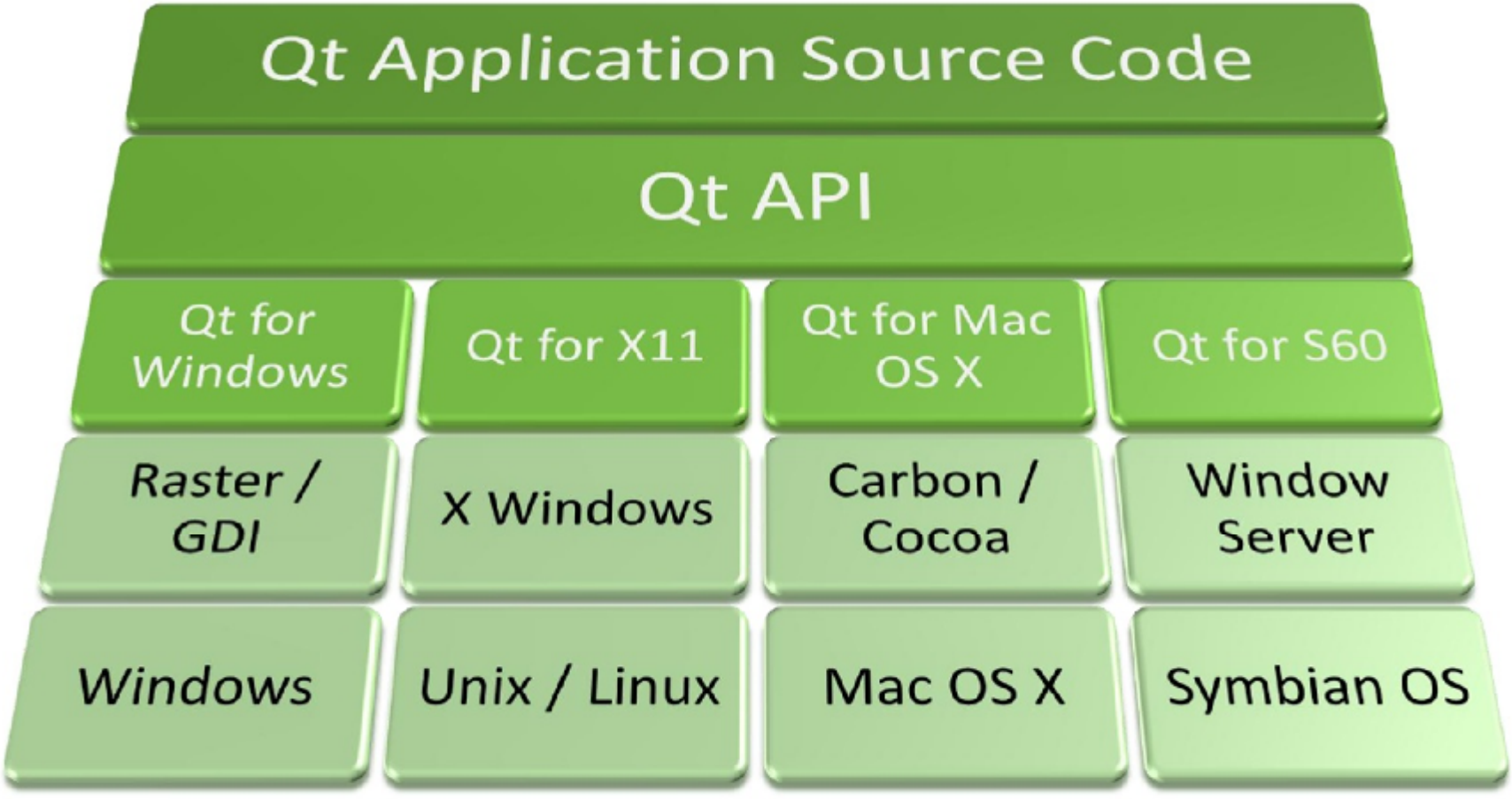
Qt

- Cross-platform C++ osztálykönyvtár (kb. 500 osztály, 4000 metódus)
  - GUI készítésére („natív look-and-feel”)
  - XML feldolgozás
  - Hálózat kezelés
  - Adatbázis támogatás
  - Szálkezelés
  - Többnyelvű alkalmazások
  - OpenGL
  - …
- C++ kibővítése új funkciókkal: „Qt object model”

Qt Hello World

```
#include <QApplication>
#include <QPushButton>
int main(int argc, char *argv[])
{
      QApplication app(argc, argv);
      QPushButton helloButton(
            "Hello World");
      helloButton.showMaximized();
      return app.exec();
}
```

Qt architektúra

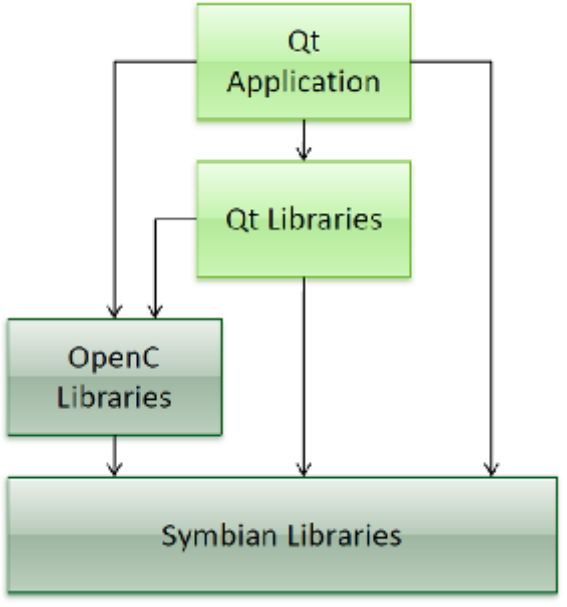


Qt és Symbian

- 2008: Nokia felvásárolta a Qt-t gyártó Trolltech-et
- 2010: Qt from Symbian már nem beta, hivatalosan is támogatott
- 2011: első Symbian^4 telefon – Qt az alapértelmezett keretrendszer

Qt for Symbian

- Lehetővé teszi teljes értékű mobilszoftverek fejlesztését Qt + standard C++-ban
- Teljesen testreszabható UI
- S60 3rd FP1 és újabb készülékekre
- Mobilspecifikus funkciók elérése: Qt Mobility API



Ellenőrző kérdések, tippek

- Milyen szoftverplatformok közül választhatunk ha Symbian-ra akarunk fejleszteni?
- Melyik platformot mikor érdemes választani, mik az előnyeik/hátrányaik?

Symbian

Bréking

Ctrl-alt-del


NOKIA
A problem has been detected and windows has been shut down to prevent damage
to your computer.
DRIVER_IRQL_NOT_LESS_OR_EQUAL
If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use safe mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical Information:
Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
ctrl
Alt
Del

Bréking: Nokia 2011.02.11.

- Nokia fő okostelefon platformja a jövőben a Windows Phone lesz (mikortól is?)
- Symbiant leépítik?
  - Jelenleg 200 millió Symbian telefon felhasználó
    - (iPhone: 4 év alatt összesen 70 millió eladott telefon)
  - 2011-ben még 150 Symbian-ra épülő mobil-t akarnak eladni
- MeeGo egyelőre marad „kísérleti platform”

Symbian

Natív alkalmazások fejlesztése Symbian-ra

Natív kód

- Előny: Gyors, minden funkciót elérünk, teljes kontroll
- Hátrány: a kód nem újrahasznosítható más platformokon (kivéve Qt)
- Minden fejlesztőeszköz ingyenes
- Lehetőségek:
  - Symbian C++ : Symbian SDK
  - Qt / Standard C++ : Nokia Qt SDK

Miért fejlesszünk Symbian C+ + -ban?

- Miért fejlesszünk Symbian C++-ban?
  - A platform minden funkcióját elérhetjük
  - Szerverek (háttérben futó szolgáltatások) írása
  - Teljes kontroll a kód és a platform felett
  - Legnagyobb hatékonyság, gyorsaság
- Miért NE fejlesszünk Symbian C++-ban?
  - Nem hordozható kód
  - Sok mindent kevesebb munkával meg lehet írni Qt/Standard C++ segítségével

Symbian C++ SDK

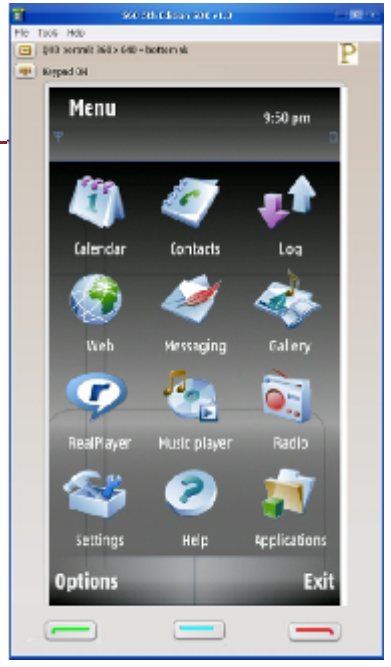
- Minden fejlesztőeszköz ingyenesen elérhető: http://www.forum.nokia.com
- Egy gépre tetszőleges számú különböző SDK-t telepíthetünk!
- Operációs rendszer
  - Hivatalosan Windows XP, 2000, 7
    - Vista is OK, kis hekkeléssel
  - Linux, Mac OS X nem hivatalos SDK patchek, NINCS emulátor
    - http://www.martin.st/symbian/
    - Wine-al még IDE (Carbide C++) is használható

Symbian C++ SDK tartalma

- Build toolchain
  - GCC (GNU Compiler Collection) ARM architektúrához
  - Header fájlok, library-k
  - Erőforrás fordító, grafika konverter, SIS készítő stb.
- Dokumentáció
- Példaprogramok
- Emulátor

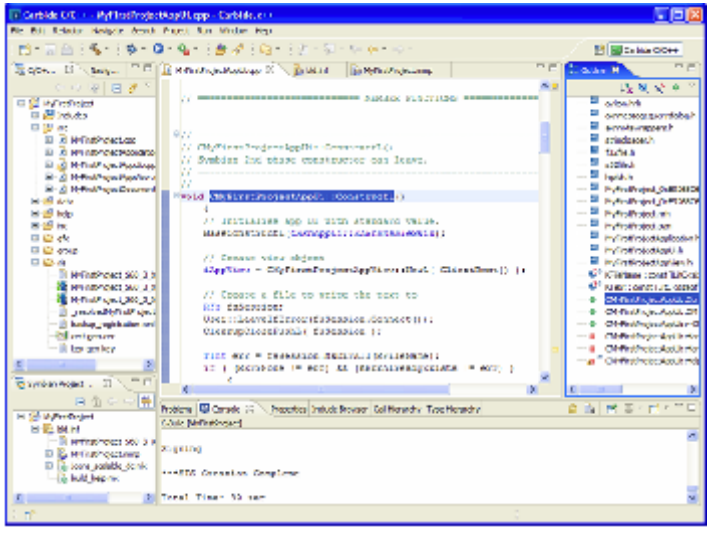
Symbian emulátor

- Támogatja a telefon legtöbb funkcióját
- Az OS kódjára épül, a telefonon futó OS nagy részét emulálja
- Amikkel gond lehet például
  - Telefonálás
  - Bluetooth kommunikáció
- Fő különbség emulátor és telefon között
  - Emulátor egyetlen processzban fut
  - Ami a telefonon processz, az az emulátoron szál
  - Emulátor/telefon teljesítménye jelentősen eltérhet egymástól



Carbide C++

- IDE Symbian C++ alapú projektekhez, ingyenes, SDK-tól külön kell letölteni
- Eclipse alapú
  - Open source, Java alapú, „IDE” keretrendszer
  - Pluginok támogatása
- 2.0 óta minden verziója ingyenes
- Támogatja a Symbian projektek minden beállítását

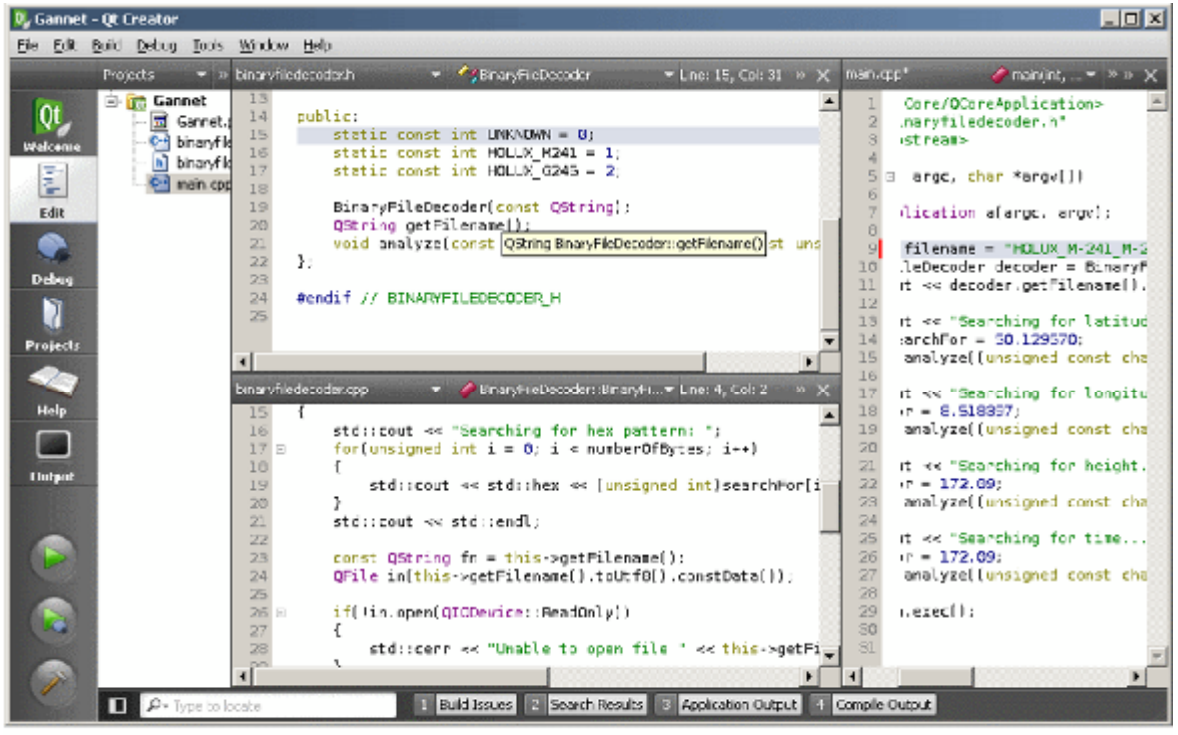


Nokia Qt SDK

- Symbian és Maemo/MeeGo alapú eszközökhöz, Qt és standard C++-ban való fejlesztéshez
  - Mellesleg Symbian C++ kódot is tud fordítani
- Minden eszköz egy csomagban
  - Build toolchain, dokumentáció, IDE, szimulátor
- Támogatott operációs rendszerek:
  - Windows XP, Vista, 7
  - Linux (Ubuntu 8.04 vagy frissebb)
  - Mac OS 10.6 vagy frissebb

Qt Creator

- IDE
  - Code complition/highlighting
  - On-device debug
  - Szimulátor
  - API help



Qt vagy Symbian C++ ?

- Qt
  - Standard alkalmazások 90%-a
  - Ha látványos felhasználói felület kell
- Symbian C++
  - Operációs rendszer szolgáltatások
  - Ha szempont a maximális teljesítmény
  - Kompatibilitás a korábbi S60 verziókkal is (Qt csak S60 3.1-től támogatott)
- A kettőt keverni is lehet

On-device debug

- Symbian SDK és Nokia Qt SDK is támogatja
- Az alkalmazás a telefonon fut, de közben a PC-n érjük el a debug funkciókat
- A telefonra fel kell telepíteni egy speciális ágenst (App TRK), mely tartja a kapcsolatot a Carbide C++-al
  - USB
  - Bluetooth
- Az alkalmazás telepítése és futtatása automatikusan történik
  - Nagy előny a manuális telepítéssel szemben, ahol esetleg 5-6 dialógust is végig kell nyomkodni

Telepítés telefonra

- Telefonra .SIS fájlok telepíthetők (= telepítő csomag), ***S****ymbian* ***I****nstallation* ***S****ource*
  - A programhoz tartozó összes fájl és metaadatok egy tömörített fájlban
- SIS-t aláírása (platform security)
- SIS felrakása a telefonra
  - Felmásolás memóriakártyára
  - Nokia PC/Ovi Suite alkalmazáson keresztül
  - Átküldés pl. Bluetooth üzenetként, MMS-ként
  - Letöltés böngészővel

Ellenőrző kérdések, tippek

- Milyen programot kell beszerezni ha Symbian C++ vagy Qt-ra akarok fejleszteni?
- Hogyan tudok feltelepíteni egy natív Symbian-os alkalmazást a telefonra?
- Mi az az on-device debug és milyen IDE-k támogatják Qt és Symbian C++ fejlesztésnél?

Symbian

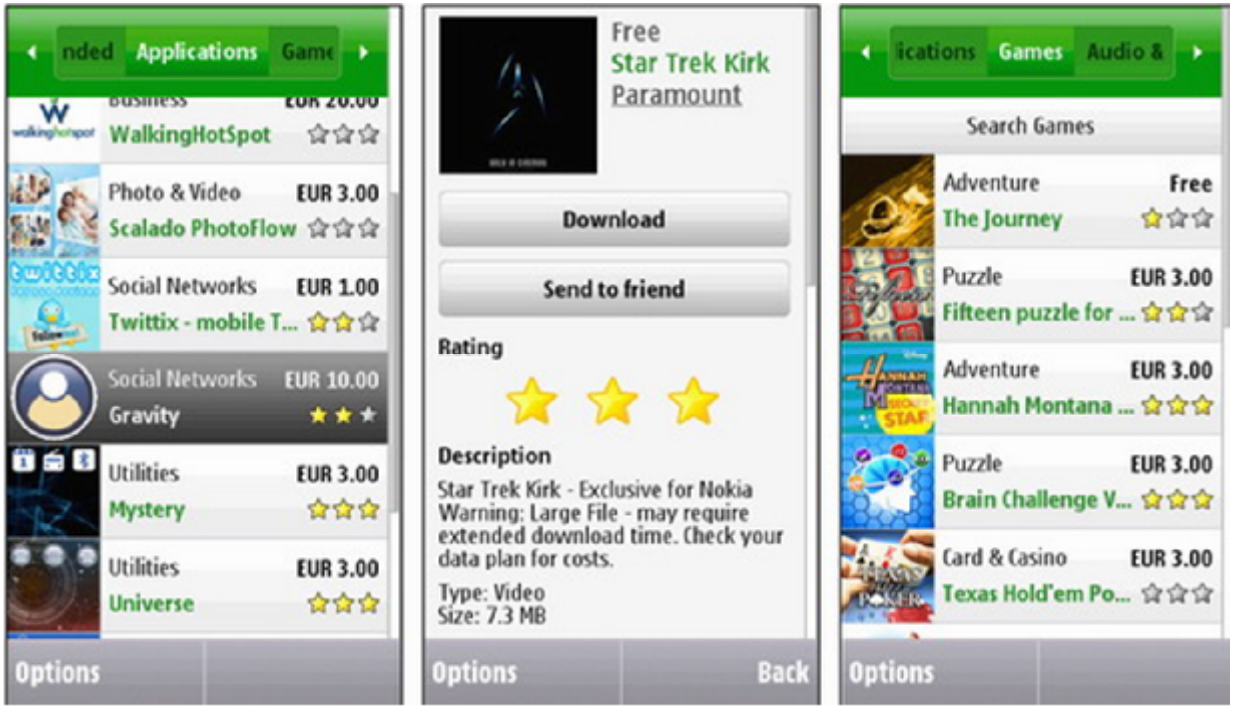
Natív alkalmazások terjesztése

Natív alkalmazások terjesztése

- Telepítőfájl (.SIS) készítése és tetszőleges módon való terjesztése (pl. webes letöltés)
  - Ha bármilyen „védett” funkciót használunk, az alkalmazást alá kell íratni ($)
- Ovi Store
  - Nokia alkalmazás boltja

Ovi Store

- http://store.ovi.com



Ovi Store

- 2009-ben indult, online alkalmazás „bolt”
  - Telefonon böngészhetők az alkalmazások
  - Fizetés, letöltés után automatikusan települnek
- Fejlesztés Ovi-ra
  - A fejlesztők jutalékot kapnak az eladásokból (70% ?)
  - 1 EUR a beugró
  - http://info.publish.ovi.com

Ellenőrző kérdések, tippek

- Hogyan terjesztheti egy fejlesztő a natív Symbian alkalmazásait?
- Hogyan működik az Ovi Store?
  - Mit kell tenni a fejlesztőnek, hogy beléphessen?
  - Mi az üzleti modell?

Symbian, Qt

Mobilszoftverek

S2

S2 – Tartalom

- Symbian C++ alapok
- Kivétel és memóriakezelés
  - TRAP – leave
  - Kétfázisú konstruktor
- Active object

Symbian

Symbian C+ + alapok

Symbian C++

- Miért lett ilyen?
  - Mikor a Symbiant megalkották még sok minden nem volt benne a C++ szabványban
  - A fordító amit a korai SDK-k mellé adtak több hiányossággal is rendelkezett
    - Például nem hívódott meg automatikusan még a stack-en létrehozott objektumok destruktora sem
  - Standard C++ megoldásainál memóriatakarékosabb és kisebb számításigény
- Hátrányok
  - Nehezebben elsajátítható, lassabb a programozás
  - Ma már szabványos C++ fordító (GCC) van a Symbian SDK-kban

Symbian C++ sajátosságok 1/2

- C++ megkötésekkel és saját megoldásokkal
- Saját kivételkezelés (**`TRAP`** és **`User::Leave`**)
- Memóriaszivárgás elkerülésére (nincs kivétel, tehát a „*Leave*” veszélyes a konstruktorban): kétfázisú konstrukció, *CleanupStack*
- Sajátos elnevezési konvenció: nem a változó típusa, hanem szerepe szerinti elnevezés; függvények végének és elejének jelölése memóriakezelés szempontjából

Symbian C++ sajátosságok 2/2

- Saját generikus adatszerkezetek
  - Nem volt STL, STL nagyobb memória igényű
- String helyett „deszkriptorok”
  - Nem volt string osztály
- Többszálúság helyett *ActiveObject*
  - Gyors
- Template-k használata csak megkötésekkel (*ThinTemplate*)
  - Template sok kódot generál, memória…
- RTTI-t (futás idejű típusazonosítást) nem használunk
  - Lassú, nem volt szabványos

Elnevezési konvenciók

- Osztályok
  - T, C, M, R osztályok
- Változók
- Függvények
  - Leave-elő (kivétel dobó) függvény nevét L-el zárjuk: FunctionL()
  - Ha a függvény valamit fennhagy a Clean Stack-en, akkor C-vel: CreateSomethingLC()

Osztályok elnevezése

- T, C, M vagy R betűvel kezdődnek
- A betű jelzi, hogy az osztályt hogyan kell kezelni a memóriafoglalás/felszabadítás szempontjából
  - R osztályok erőforrásokat azonosítnak és le kell őket zárni egy metódushívással használat után
  - C osztályok csak referencia/mutató szerint adhatók át, nem hozhatjuk létre a stack-en, csak a heap-en
  - M osztályok („mixin”) interfészek
  - T osztályok „egyszerű típusok”, érték szerint átadhatók, stacken létrehozhatók
- Pl. CBase, TInt, RSocket, RFile, MPhoneObserver, stb.

Változók

- Argumentumok
  - a-val kezdődnek (argument), pl. TInt aParameter;
- Tagváltozók
  - i-vel kezdődnek (internal), pl. CChild* iChildren;
- Lokális változók
  - Kisbetűvel kezdődnek pl. Tint i;
- Globális változók
  - Nagybetűvel kezdődnek pl. CEngine* Engine;
  - kerülendők

Ellenőrző kérdések, tippek

- Miért létezik a Symbian C++? Mi ez egyáltalán?
- Mik a legfőbb tulajdonságai/sajátosságai, miben tér el a szabványos C++-tól?
- Milyen elnevezési konvenciókat használ?
  - Változók (tagváltozó, argumentum, lokális, globális)
  - Osztályok (mi alapján nevezzük el az osztályokat és miért)
  - Függvények (L és C betűk)

Symbian OS

Kivétel- és memóriakezelés

Kivétel- és memóriakezelés

- Kivételkezelési konvenciók
- Cleanup Stack
- Kétfázisú konstrukció

Kitérő: C++ memóriakezelés 1/2

- Stack („automatikus memóriafoglalás”)
  - Ideiglenes adattárolásra a hívott metódus/függvény számára
  - A lefoglalt memória automatikusan felszabadul a függvény/metódusból való kilépéskor (vagy kivételkor)
  - Korlátos maximális méretű (Symbianon default 8 Kbyte!)
    - Átállítható
  - Memóriafoglalás a stacken:
    - int i;
    - MyClass myInstace;

Kitérő: C++ memóriakezelés 2/2

- Heap („dinamikus memóriafoglalás”)
  - Sok van belőle
  - Objektum létrehozása a stacken
    - MyClass* mc = new MyClass;
    - int* szamok = new int[10];
  - A létrehozott objektumok nem szabadulnak fel automatikusan, azokat explicit kell felszabadítani
    - delete mc;
    - delete [] szamok;

Symbian kivételkezelési konvenciók

- Sokáig nem működött a standard C++ kivételkezelés
  - Történelmi okok (régi gcc)
  - Többletmunka
- Helyette:
  - kivétel = leave
  - Throw: **`User::Leave`** („leave-elés”)
  - Catch: **`TRAP, TRAPD`**
  - Függvénydeklarációkor *throw*: záró L betű pl. FunctionL()
- **new** operátor: 0-t ad vissza, ha nincs memória
- **`new (ELeave):`** kivételt dob, ha nincs memória

Példa: TRAPD használata

```
void Fuggveny() // nincs L a végén!
{
  //leave-elő függvény hívása
  int hiba;
  TRAP(hiba, ProbaL() );
  if ( hiba != KErrNone )
  {
      // ha ProbaL() leave-el, akkor ide
      // adódik a vezérlés, és lekezelhetjük
      // a hibát.
    }
}
```

TRAP-ek egymásba ágyazhatók

```
   int hiba;
   TRAP(hiba, Függvény1L() ); // 1. trap szint (külső)
   if ( hiba != KErrNone )
   {
   }


void Függvény1L()
{
   int hiba;
   TRAP(hiba, Függvény2L() ); // 2. trap szint (belső)
   if ( hiba != KErrNone )
   {
   }
}
```

- Leave esetén mindig a legbelső TRAP-hez lép vissza

Symbian kivételkezelési konvenciók

- Ha egy függvény leavelhet (kivételt dobhat), akkor a függvény nevét kötelezően L-el zárjuk!!!

```
void DoSomethingL();
TInt ItMayAlsoLeaveL();
```

Symbian kivételkezelési konvenciók

- Függvény leavelhet (tehát L-el zárjuk a nevét):
  - Közvetlenül meghívja **`User::Leave()`**-et
  - Heapen foglal **`new (ELeave)`**-el
  - Más leavelő függvényt hív (vagyis olyan függvényt, aminek a neve L-el zárul)
- Ha egy függvény „elkapja” (TRAP-eli) az összes leave forrást, akkor ő maga nem leavelhet, tehát nem kell L

Probléma: takarítani ki fog?

```
void doExampleL()
{
  // T osztály, stacken is létrehozható
  TBuf<10> buf;

  // C osztály, a heap-en kell létrehozni
  // Memóriafoglalás, ha nem sikerül Leave
  CExample* myExample = new (ELeave) CExample;

  // nem leavelő kód
  myExample->iInt = 5;

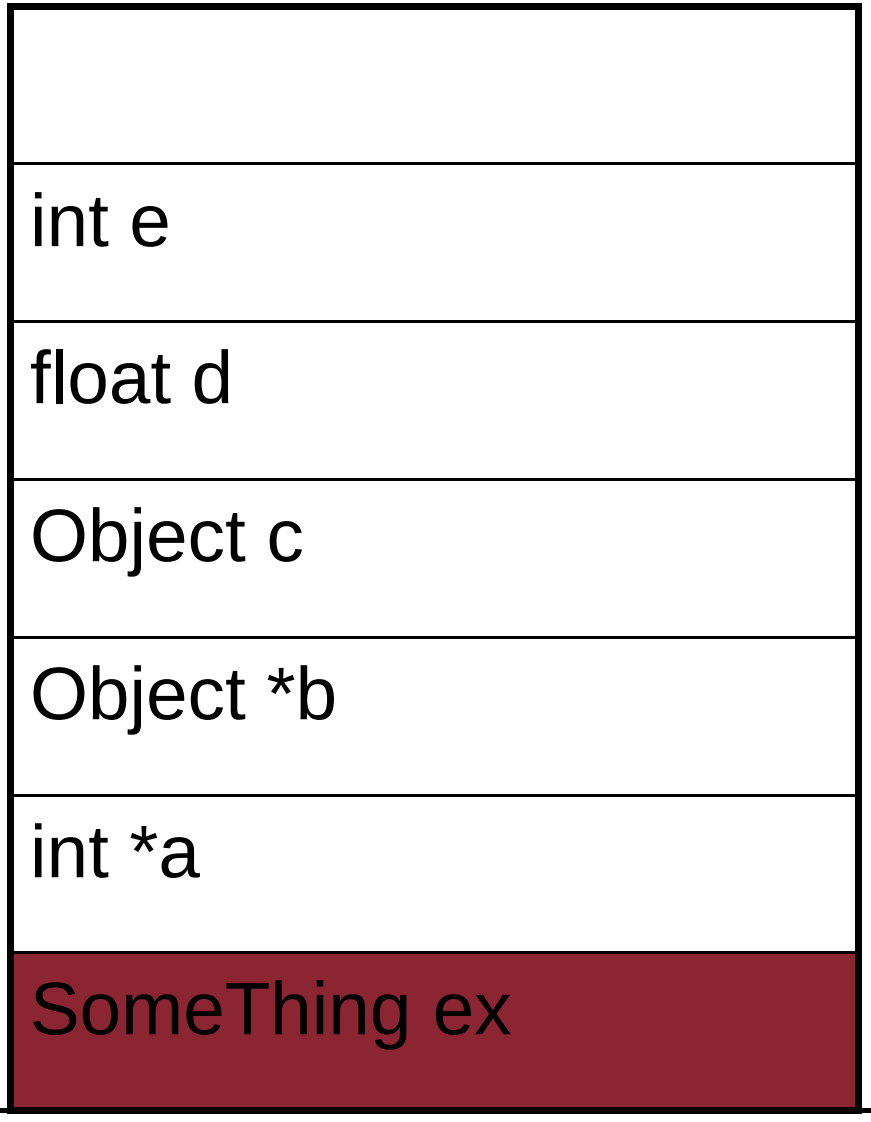
  // PROBLÉMA: Leave esetén myExample nem szabadul fel,
  // destruktora sem hívódik meg
  myExample->DoSomethingL();

  // delete
  delete myExample;
}
```

C++ kivételkezelés

Ha kivétel történik

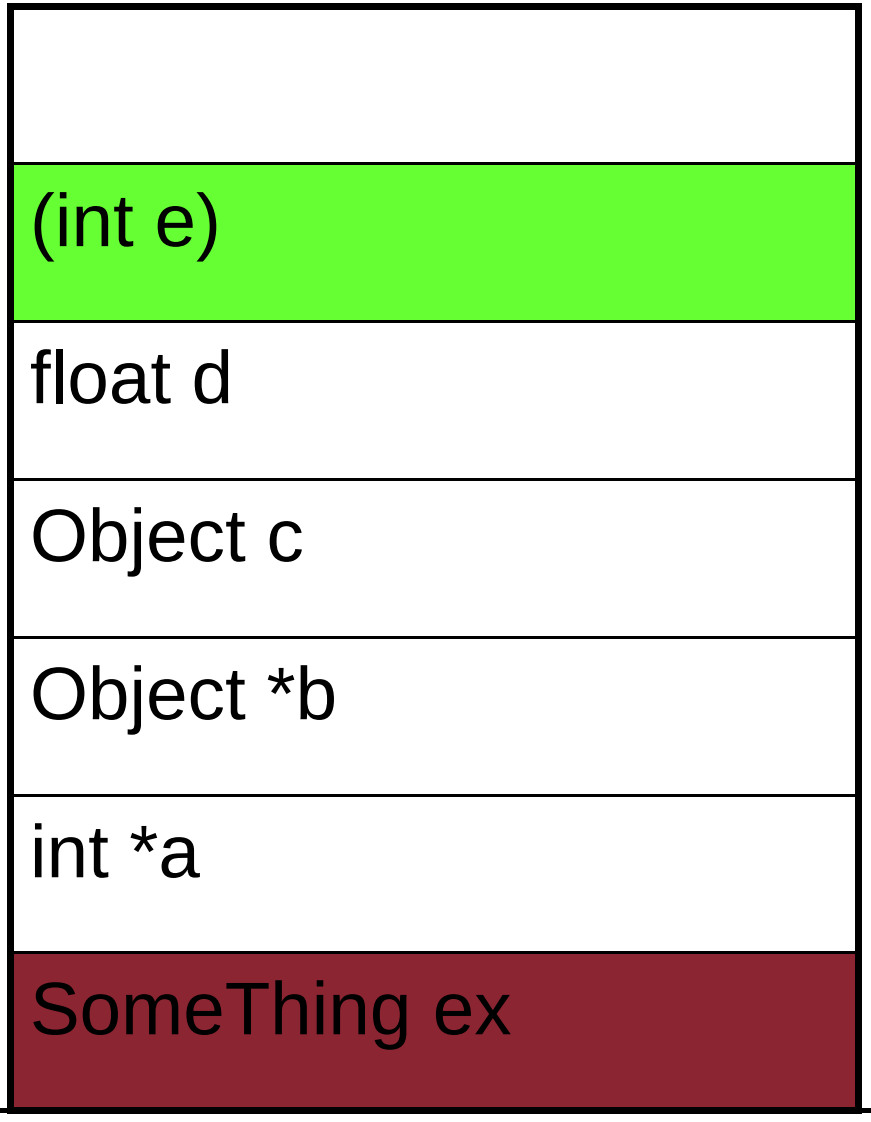
- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



C++ kivételkezelés

Ha kivétel történik

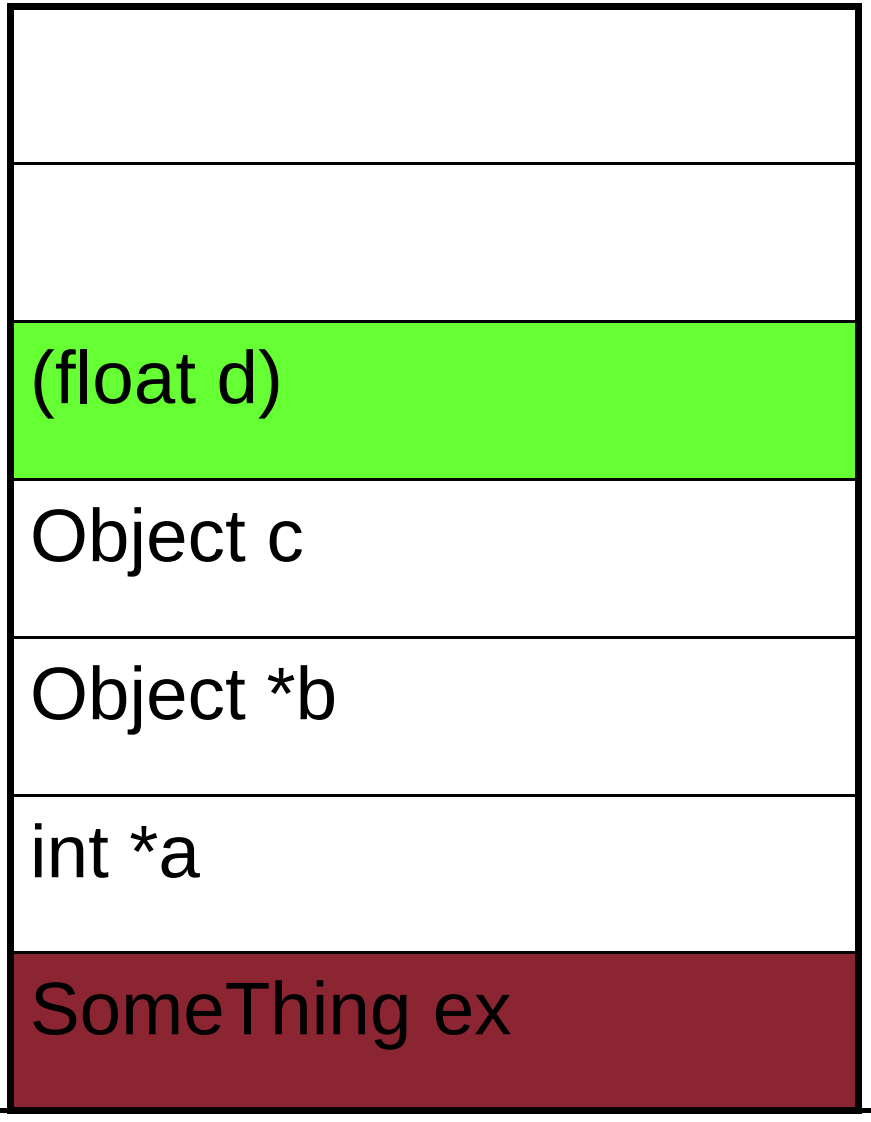
- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



C++ kivételkezelés

Ha kivétel történik

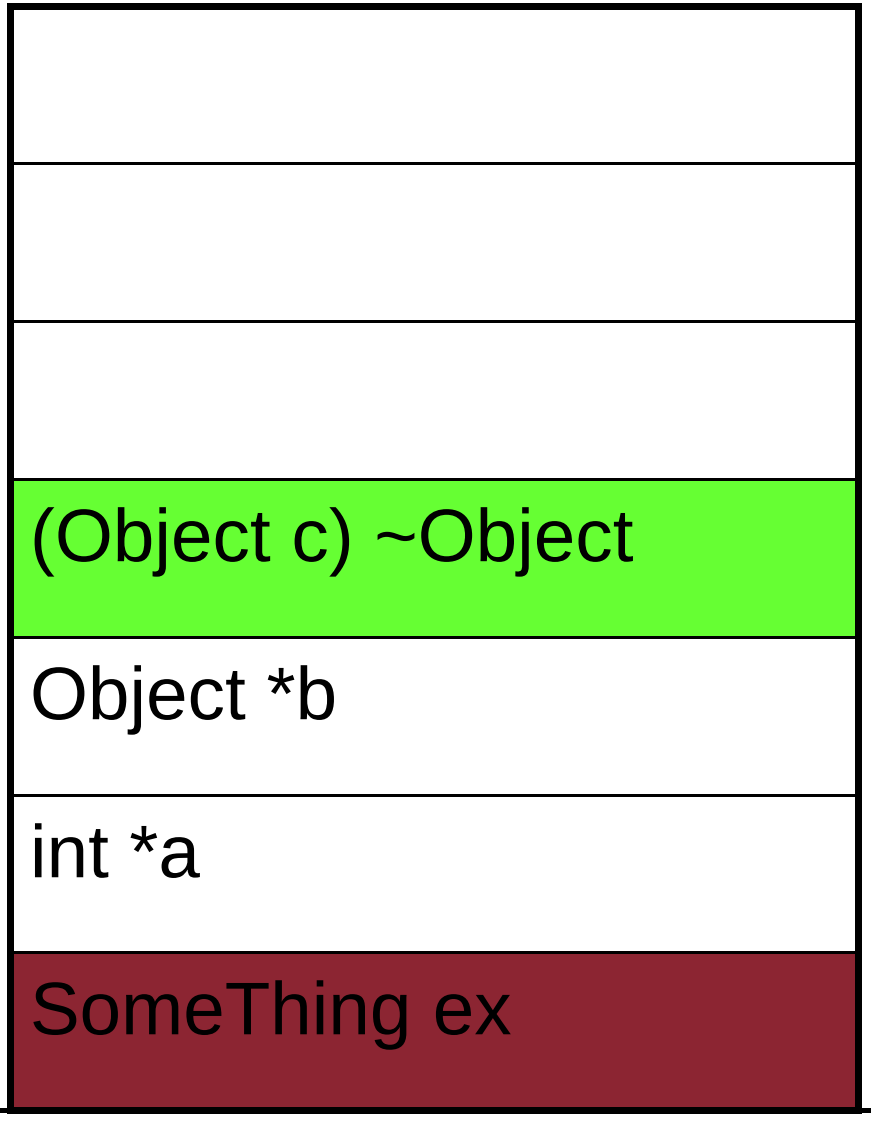
- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



C++ kivételkezelés

Ha kivétel történik

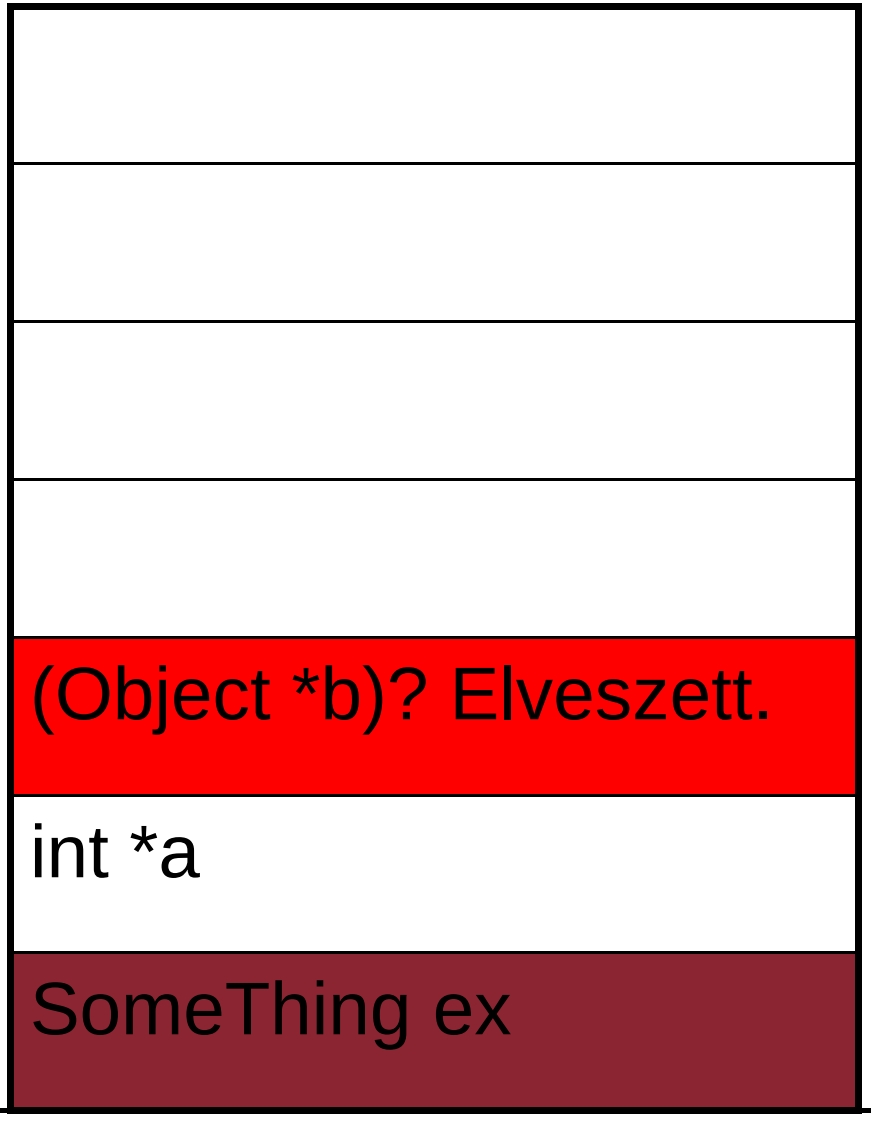
- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



C++ kivételkezelés

Ha kivétel történik

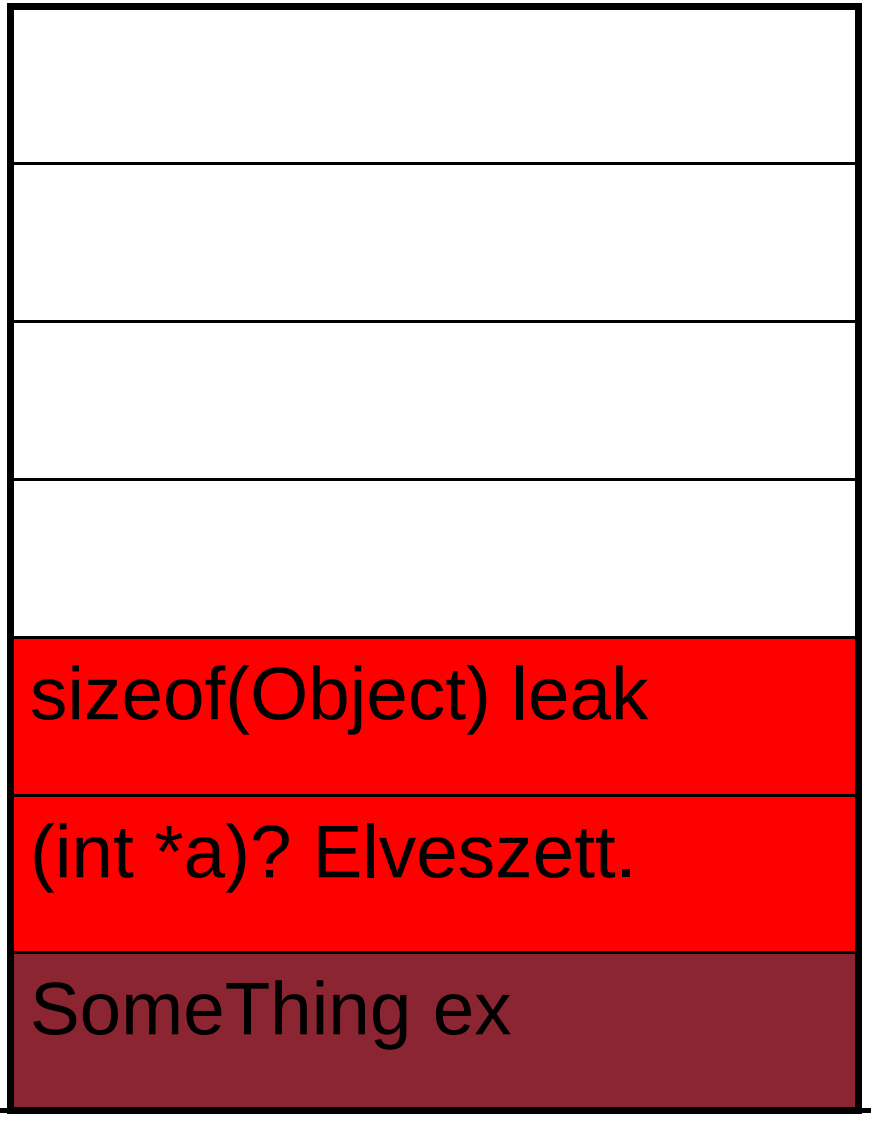
- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



C++ kivételkezelés

Ha kivétel történik

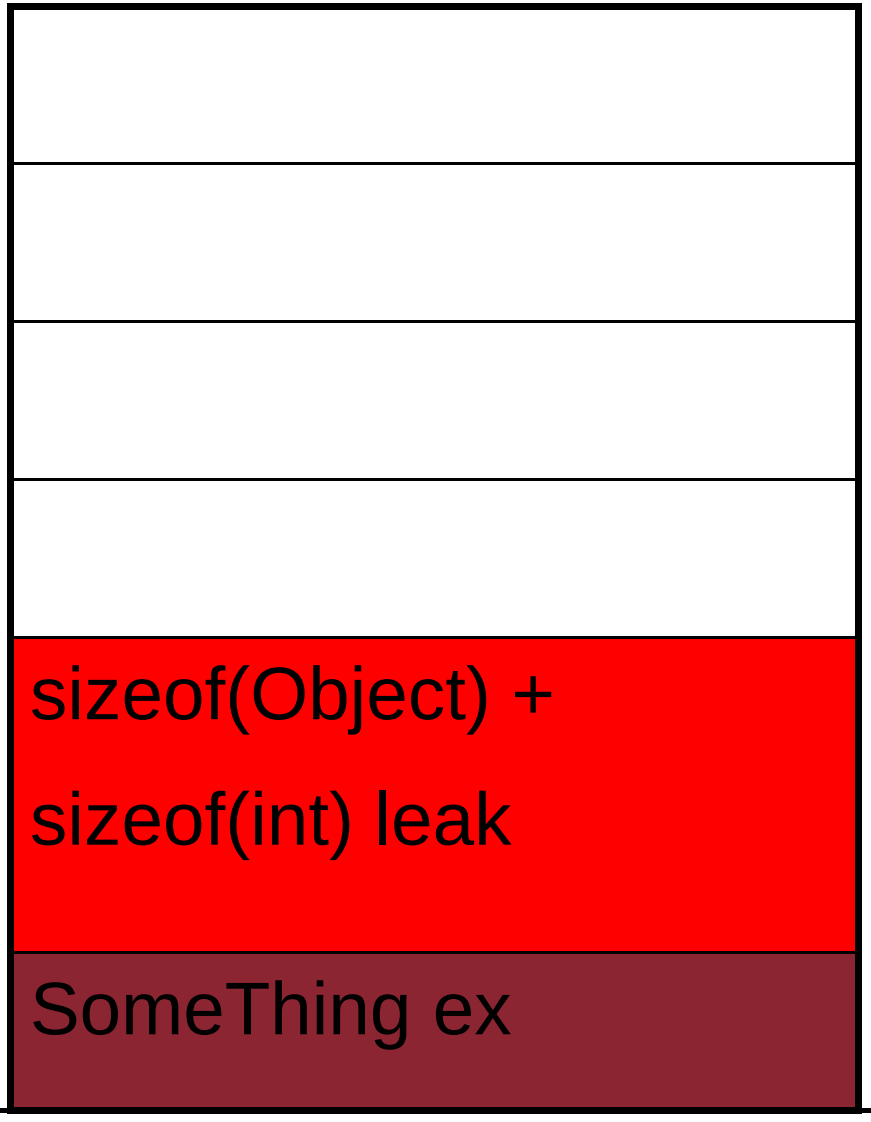
- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



C++ kivételkezelés

Ha kivétel történik

- A program-stack visszafejtésre (rollback) kerül a catch szintjéig
- Az objektumok destruktorai meghívásra kerülnek
- A mutatók alapvetően elvesznek (smart pointerekkel elkerülhető)



Cleanup Stack

- Rendeljünk a programunkhoz (szálhoz) egy „biztonsági tárolót” = Cleanup Stack
  - Mutatókat és referenciákat tárol
  - Leave esetén felszabadulnak a rárakott objektumok
- Csak olyan objektumokat rakunk fel rá, amiket más kód nem felügyel (nehogy kétszer szabadítsuk fel leave esetén)
- Tagváltozót törlünk, és újra helyet foglalunk: *delete* után állítsuk 0-ra, hogy a helyfoglaláskor fellépő leave esetén se legyen probléma

Leave menete

1. User::Leave() hívása
2. Cleanup Stack visszafejtése
3. A normál stack visszafejtése a standard C++ kivételfeldolgozás részeként
4. A TRAP makró utáni sortól folytatódik a kód futtatása
   - Itt lehet feldolgozni a Leave-et

Symbian C++ hibakezelés

A veremvisszafejtés igénylő változókat a Cleanup Stacken kell elhelyezni

(stack)	(Cleanup Stack)
int e	
float d	
Object c	
Object *b	
int *a	Object *b
int ecode	int *a

Symbian C++ hibakezelés

A veremvisszafejtés igénylő változókat a Cleanup Stacken kell elhelyezni :

- Objektum-mutató: törlésre kerül, ha a CBase-ből származik (azaz C-osztály)

(stack)	(Cleanup Stack)
int e	
float d	
Object c	
Object *b	
int *a	delete b
int ecode	int *a

Symbian C++ hibakezelés

A veremvisszafejtés igénylő változókat a Cleanup Stacken kell elhelyezni :

- Általános mutató: a lefoglalt memória felszabadításra kerül

(stack)	(Cleanup Stack)
int e	
float d	
Object c	
Object *b	
int *a	
int ecode	delete a

Symbian kivételkezelési konvenciók értelme

- Ha helyesen használjuk a Cleanups Stack-et akkor a programunk minden körülmény között memóriaszivárgás mentes lesz
- A Symbian által javasolt elnevezési konvenciók a Cleanup Stack használatát hivatottak megkönnyíteni
  - Típusok (C,T,R, M)
  - Változók (a, i, lokális…)

Objektumok a Cleanup Stack-en

- Felrakás:
  - A leggyakrabban használt: heapen létrehozott objektumok, C osztályok
    - **`CleanupStack::PushL`**
  - R osztályok
    - **`CleanupClosePushL`**
    - **`CleanupReleasePushL`**
    - **`CleanupDeletePushL`**
- Levétel
  - **`CleanupStack::Pop`**
  - **`CleanupStack::PopAndDestroy`**

Példa 1

```
void doExampleL()
{
  // foglalás ellenőrzéssel
  CExample* myExample = new (ELeave) CExample;

  // csinálunk valamit ami NEM leavelhet
  myExample->iInt = 5; // nem leavelhet

  // felkerül a celanup stackre
  CleanupStack::PushL(myExample);
  myExample->DoSomethingL(); // leavel-het

  // törlés előtt levesszük a cleanup stackről
  CleanupStack::Pop();

  // felszabadítás
  delete myExample;
}
```

Példa 2

```
TInt* szam = new (ELeave) TInt;
CleanupStack::PushL(szam);

UtasitasAmiLeavelhetL(); // leavelhet!

// szam-ot levesszük és rögtön töröljük is
CleanupStack::PopAndDestroy();
```

Kétfázisú konstrukció (1/4)

- Probléma:

```
TInt ValamilyenFuggvenyL()
{
    CKulsoOsztaly * kulso = new (ELeave) CKulsoOsztaly();
    CleanupStack::PushL(kulso);
    /*…egyéb L műveletek végzése*/
    CleanupStack::PopAndDestroy();
}

CKulsoOsztaly::CKulsoOsztaly() //konstruktor
{
  iTagValtozo=new (ELeave) CBelsoOsztaly(); //Jajj!
}
```

Kétfázisú konstrukció (2/4)

- Megoldás:

```
class CKulsoOsztaly : public CBase
{
public:
  ~CKulsoOsztaly(); //destruktor
  static CKulsoOsztaly* NewL(); //példányosító metódusok
  static CKulsoOsztaly* NewLC();
protected:
  /*védett konstruktor, közvetlenül nem példányosítható
  az osztály:*/
  CKulsoOsztaly(); //első fázis, nem leavelhet
  void ConstructL(); //második fázis, leavelhet
};
```

Kétfázisú konstrukció (3/4)

```
void CKulsoOsztaly::ConstructL()
{
    //ebből már nem lesz probléma
    iTagValtozo = new(ELeave) CBelsoOsztaly();
}
CKulsoOsztaly * CKulsoOsztaly::NewLC()
{
    //konstruktor, egyszerű inicializálások
    CKulsoOsztaly * self = new (ELeave) CKulsoOsztaly();
    CleanupStack::PushL(self);
    self->ConstructL(); /*veszélyes inicializálások, de ekkorra az objektum
       már a CleanupStack-en van*/
    return self;
}
CKulsoOsztaly * CKulsoOsztaly::NewL()
{
    CKulsoOsztaly * self = NewLC();
    CleanupStack::Pop();
    return self;
}
```

Kétfázisú konstrukció (4/4)

- Használat:

```
TInt ValamilyenFuggvenyL()
{
   CKulsoOsztaly * kulso = CKulsoOsztaly::NewLC();
   /*kétfázisú konstrukció, az objektum a
     CleanupStack-en marad*/
   /*…egyéb L műveletek végzése*/
   CleanupStack::PopAndDestroy();
   /*vagy CleanupStack::Pop();
   …néhány nem L-es művelet
   delete kulso;*/
}
```

Kétfázisú konstrukció - összefoglalás

- Az osztály standard konstruktorában nem hívunk leavelő kódot
- A leavelő hívásokat egy külön „második fázisú konstruktorba” tesszük, ennek neve **`ConstructL()`**
- Az osztály példányosításakor:
  1. Meghívjuk standard konstruktort (**`new`**)
  2. A „félig létrejött” objektumot feltesszük a CleanupStack-re
  3. Meghívjuk a második fázisú konstruktort (**`ConstructL`**)
  4. (Levesszük a CleanupStack-ről)
- Az utóbbi lépéseket összevonhatjuk NewL()-ben

Tagváltozók

- Mikor egy osztály tagváltozót hozunk létre a heap-en, akkor azt nem szabad felrakni a CleanupStack-re!
  - Pl. iName = new (ELeave) CName();
- Leave esetén meghívódik a gazda osztály destruktora és az törli a tagváltozót!
- Ha emellett még a CleanupStack-en is fenn lenne a tagváltozó, akkor kétszer próbálná törölni, HIBA!

Cleanup-safe HelloWorld 1/2

- Konzolos alkalmazásnál nincs alapból CleanupStack, azt létre kell hozni:

```
TInt E32Main() // Symbian alkalmazás belépési pont
{
  // CleanupStack létrehozása
  CTrapCleanup* cleanup = CTrapCleanup::New();

  TRAPD(error, MainL()); // MainL() meghívása
  if (error != KErrNone) // Leave történt
       User::Panic(_L("Hiba történt!"), error));
  delete cleanup;

  return KErrNone;
}
```

Cleanup-safe HelloWorld 2/2

```
void MainL()
{
    TSize size(KConsFullScreen, KConsFullScreen);
    CConsoleBase* console =
       Console::NewL(_L("HelloWorld"), size);
    CleanupStack::PushL(console);

  // itt már hívhatunk leavelő kódot is

    _LIT(KMessage,”Hello World!”);
    console->Printf(KMessage);
    console->Getch();

    CleanupStack::PopAndDestroy();
}
```

Ellenőrző kérdések, tippek

- Hogyan működik a kivételkezelés Symbian C++ alatt?
  - Hogyan működik a TRAP és User::Leave? Hogyan kell ezeket használni? (példakód)
  - Mi zajlik le kivétel dobásakor
  - Mikor kell L… hova tér vissza a vezérlés leave esetén adott esetben?
- Mi az a kétfázisú konstrukció?
  - Példakód…
  - Hogyan biztosíthatjuk, hogy mindkét konstruktort meghívják? (NewL, NewLC)
  - Miért ne rakjunk fel olyan tagváltozót a CS-re, amit a tartalmazó osztály destruktorában törlünk?

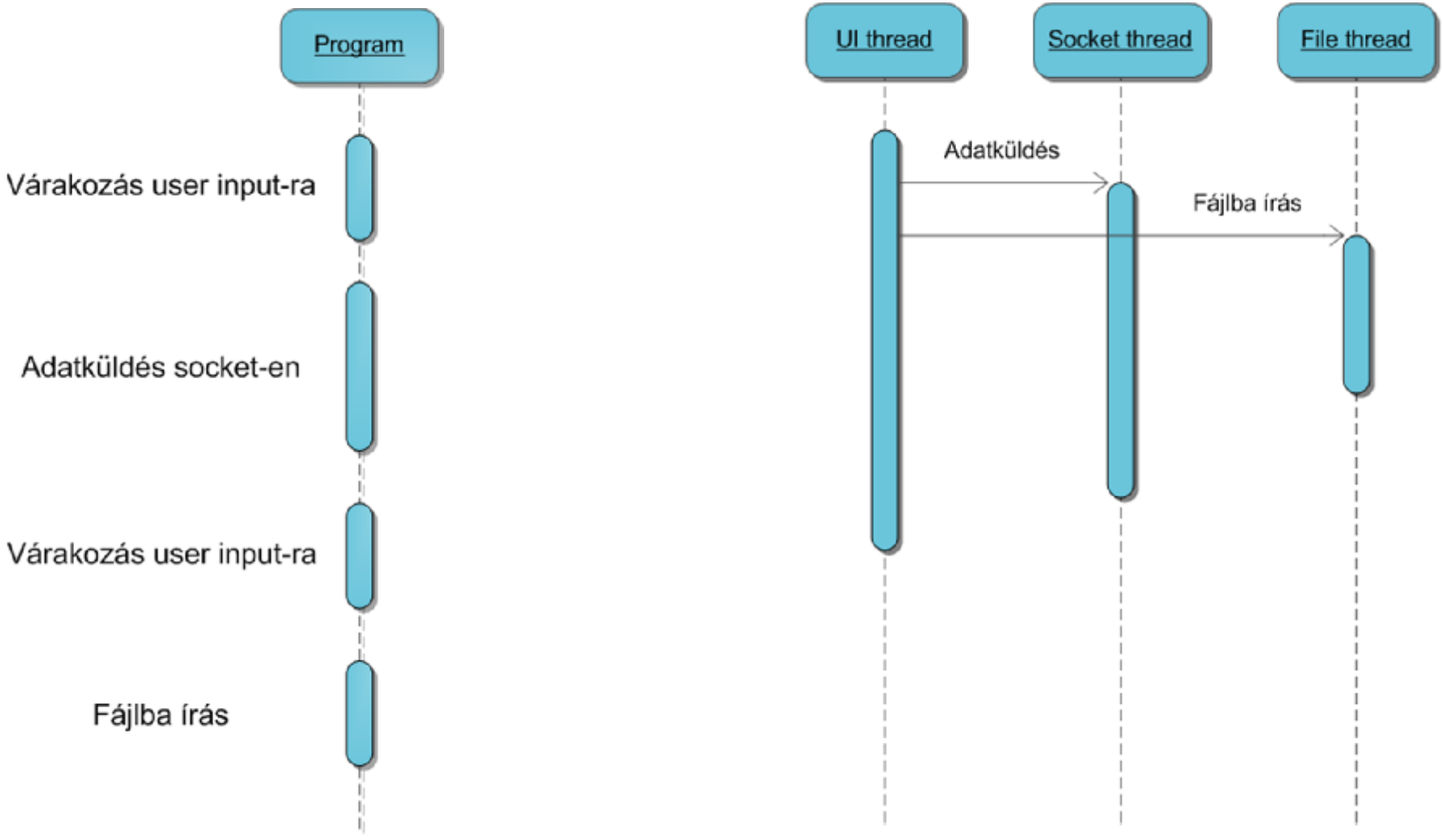
Symbian OS

Aszinkron eseménykezelés – Active Object

Konkurens programozás

- Többszálú operációs rendszer
- Hosszan tartó műveleteket külön szálakban végezzük
  - Nem blokkoljuk az egész programot
  - Billentyűnyomás, fájl művelet befejeződése, adatbázis-változásról értesítés, időzítők, …
- Több szál együttműködéséhez
  - Szálak közötti kommunikáció szükséges
  - Közös erőforrások elérését szinkronizálni kell

Single threaded vs. multi threaded



Mi a baj a szálakkal?

- Szálak menedzselése nem egyszerű (legalábbis Symbianon)
- Thread-safe adatszerkezetekre van szükség

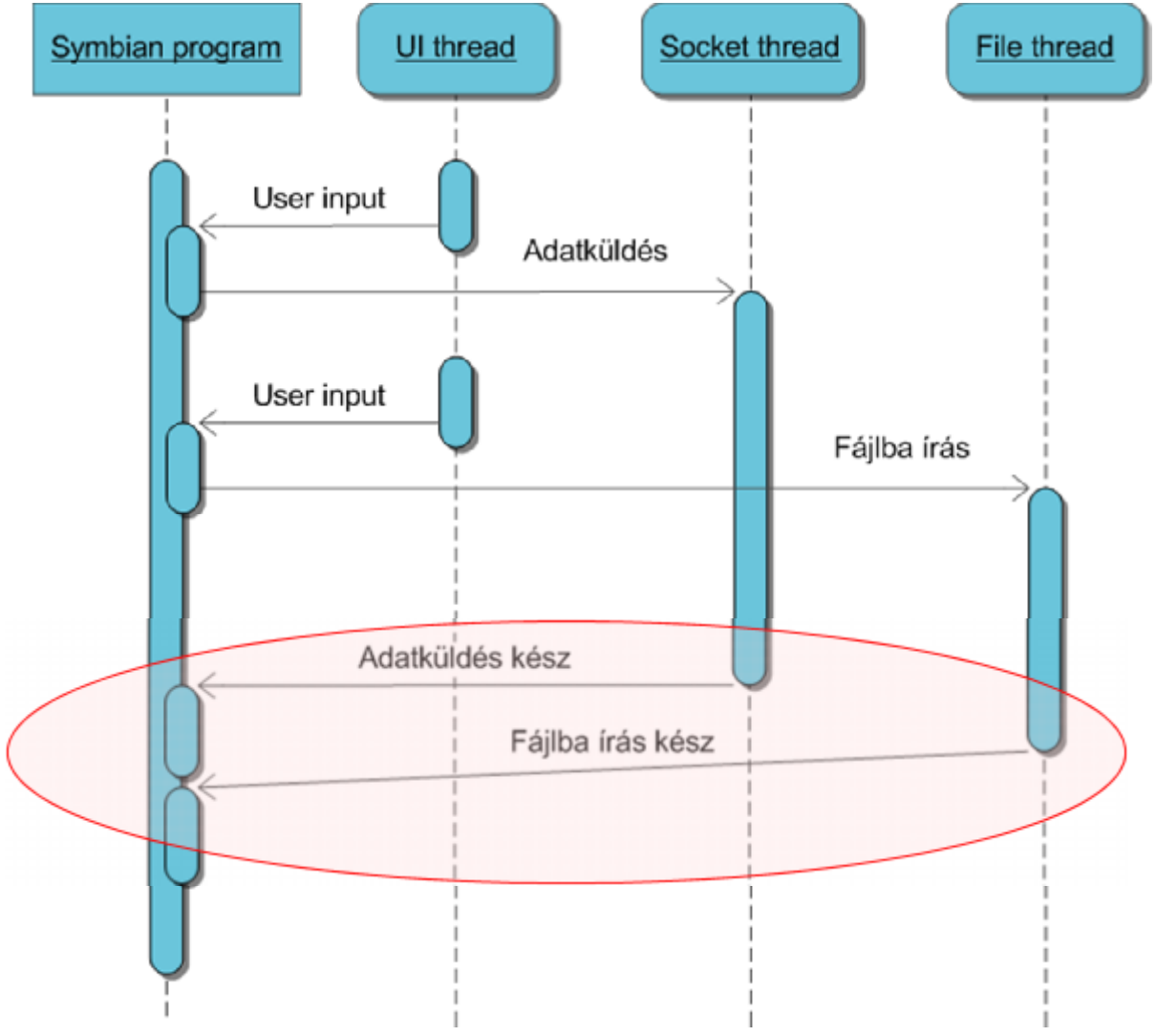
Megoldás

- Minden aszinkron visszajelzésű szolgáltatás továbbra is külön szálban fut, ám ezeket az operációs rendszer kezeli, nem a felhasználói program, pl:
  - Várakozás egy időzítőre
  - Várakozás adat beérkeztére egy socket-en
  - Várakozás hosszabb fájlművelet végére
- Ezen aszinkron eseményekre fel lehet iratkozni
  - Értesítést kapunk az esemény bekövetkezik

Virtuális párhuzamosság

- A felhasználói program csak egy szálat használ, melyben egy eseménykezelő ciklus fut (ActiveScheduler)
  - Az eseménykezelő ciklus folyamatosan ellenőrzi nem érkezett-e be egy aszinkron esemény
  - Ha bejött egy aszinkron esemény, megkapjuk a vezérlést
  - Mivel csak egy szálunk van, egyszerre csak egy aszinkron eseményt tudunk feldolgozni
    - Ha a feldolgozás elég gyorsan, a felhasználó nem érzékeli a valódi párhuzamosság hiányát

Symbian eseménykezelés



„*Fájlba írás kész*” esemény feldolgozása CSAK „*Adatküldés kész*” feldolgozása után!

ActiveObject dióhéjban

- Minden aszinkron kérés egy külön objektum: ActiveObject
  - CActive osztályból származik

1. Elindít egy aszinkron kérést egy aszinkron eseményszolgáltatóhoz
   - Aszinkron eseményszolgáltató hozza létre a szálat
2. Ha lefutott a kérés, akkor meghívódik a RunL metódusa
   - Eseménykezelő ciklus (ActiveScheduler) hívja meg RunL-t
   - RunL felfogható egy eseménykezelő callback függvényként

ActiveObject részletesebben

- Az eseménykezelő ciklust vezérli az aktív ütemező (Active Scheduler)
  - Nem-preemptív, prioriotásos ütemezés
- Aszinkron eseményszolgáltató: külön szálban, tipikusan Symbian biztosítja
  - Elérés kliens-szerver elven
  - pl. socket server, file server
- Várakozás egy aszinkron eseményre: Active Object
  - Minden olyan művelethez amire külön szálat hoznánk létre, példányosítunk egy Active Object-et

CActive

```
class CActive : public CBase {
public:
    enum TPriority { EPriorityIdle = -100 // ... };
public:
    ~CActive();
    void Cancel();
    /...
    void SetPriority(TInt aPriority);
    inline TBool IsActive() const;
protected:
    CActive(TInt aPriority);
    void SetActive();
    virtual void DoCancel() = 0; // kérés megszakítás
    virtual void RunL() = 0; // kérés lefut
    virtual TInt RunError(TInt aError); // hibakezelés
public:
    TRequestStatus iStatus;
    /...
};
```

CActive

- Minden aktív objektum közös ősosztálya CActive
- Egyetlen aszinkron kérést reprezentál
- A konstruktorában kötelező felrakni az ütemezőre CActiveScheduler::Add()-al
- TRequestStatus iStatus tagváltozója jelöli a kérés aktuális állapotát

Példa aktív objektumok

- Aktív objektumok egy tipikus Symbian alkalmazásban
  - Billentyűesemények figyelése
  - Periodikus ellenőrzések egy periodikus időzítővel (pl. 10 másodpercenként jelez)
  - Adatok küldése egy Bluetooth socketen
  - Adatok fogadása egy Bluetooth socketen
  - Írás egy fájlba
- Minden aszinkron/párhuzamos művelethez egy külön CActive leszármazott osztály

Active object műveletek

- Az aszinkron kérés indítása
- Eseménykezelés (lefutott a kérés)
- Kérés megszakítása

AO: Kérés indítása

1. Az aktív objektumok általában biztosítanak egy publikus metódust amely elindítja a kérést
   - Mindig ellenőrizni kell, hogy nincs-e már aktív kérés („tehát nem aktív az objektum”)
2. Az aktív objektum átadja az iStatus tagváltozóját az aszinkron esemény-szolgáltatónak és ezzel elindítja a kérést
3. Ha kérést elindítottuk, akkor SetActive() hívással aktív állapotba kell kapcsolni az objektumot (és innentől várakozunk)

➢ Például időzítő indítása, visszajelzés 10 mp múlva:

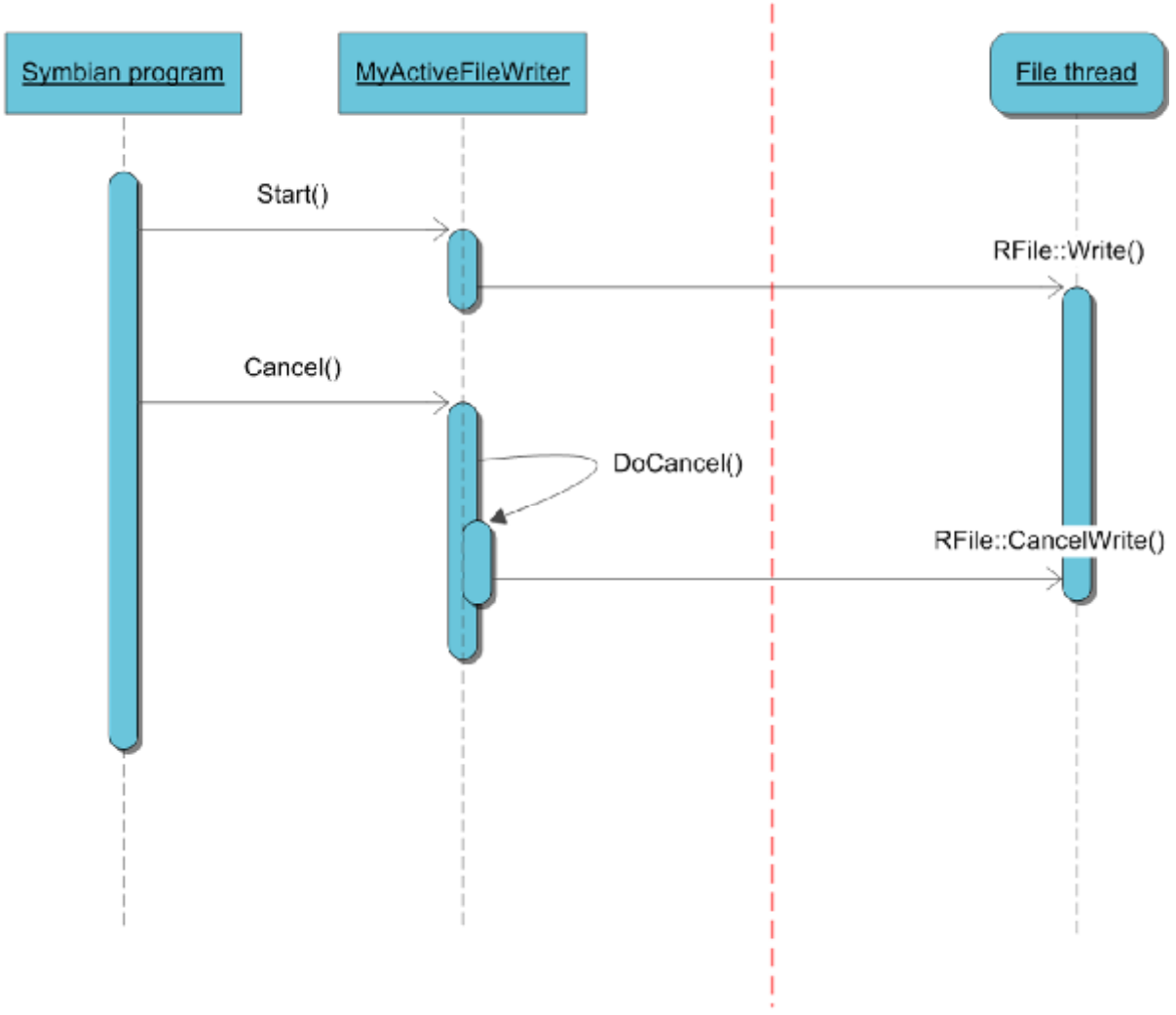
```
iMyTimer.Start(10000000, iStatus);
SetActive();
```

AO: Kérés megszakítása

- `CActive` tartalmaz egy `Cancel()` metódust, mely hatására azonnal megszakad az aszinkron kérés
- `Cancel()` a tisztán virtuális `DoCancel()`-t hívja, ezt minden leszármazott aktív objektumnak meg kell valósítania!
- `Cancel()`-t akárhányszor meg lehet hívni!
- Példa időzítő leállítása DoCancel()-ben:

```
void DoCancel() { iMyTimer.Stop(); }
```

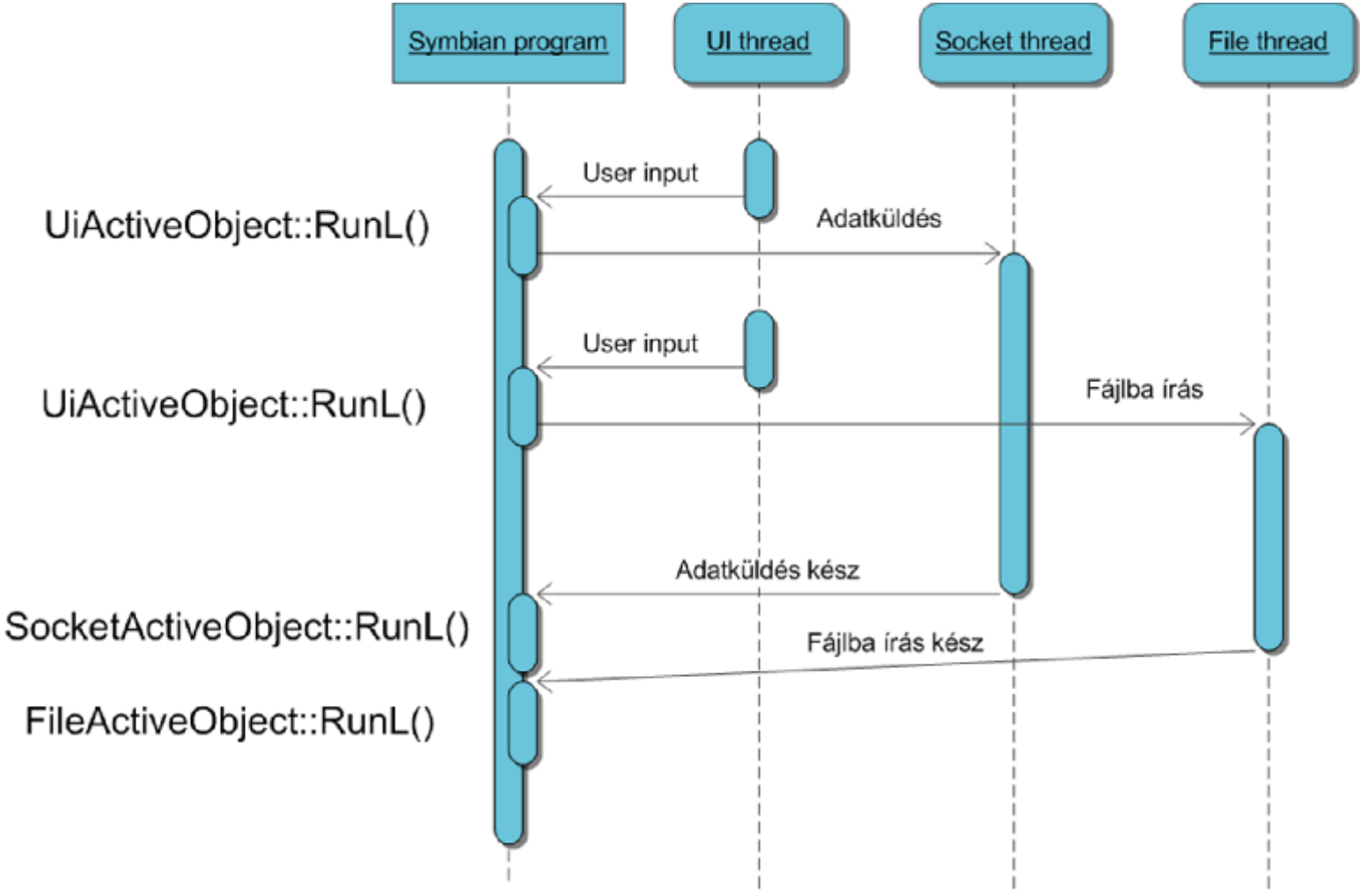
AO: Kérés indítása és megszakítása



AO: Eseménykezelés

- Ha visszatér az aszinkron kérés (lefutott de nem biztos, hogy sikeresen), akkor meghívódik az aktív objektum (ősosztályban tisztán virtuális) **`RunL()`** metódusa
- **`RunL()`**-ben általában ellenőrizzük a kérés eredményét **`iStatus`**-ban (**`KErrNone`** ha sikeresen lefutott)

AO: Eseménykezelés



ActiveObject összefoglalás

- Eseménykezelő ciklus: ActiveScheduler
  - Ez fut az alkalmazás egyetlen saját száljában
- Minden aszinkron/párhuzamos kéréshez egy külön aktív objektum
- Mikor egy aszinkron kérés lefut (= aszinkron esemény) az aktív ütemező lefuttatja a megfelelő aktív objektum RunL() metódusát
- Ha egyszerre több aktív objektumhoz tartozó kérés is lefut, a legnagyobb prioritású RunL()-je fut le legelőször

Ellenőrző kérdések, tippek

- Hogyan néz ki az eseménykezelő ciklus Symbian alatt, mik a fő komponensei (ActiveScheduler, eseményszolgáltató, ActiveObject)
- Milyen lépések zajlanak le egy aszinkron kérés (pl. fájlba írás) folyamán?
- Milyen funkciókat lát el egy aktív objektum (CActive-ből származó osztály)
- Hogyan szakítható meg egy aszinkron kérés?

Symbian, Qt

Mobilszoftverek

S3

S3 – Tartalom

- MeeGo dióhéjban
- Qt bemutatása
- Qt object model
- Qt demo

Symbian

Maemo, MeeGo

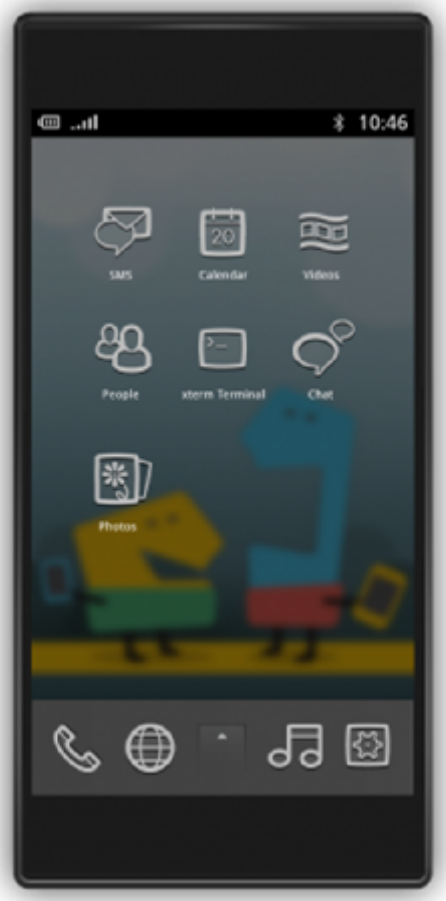
Maemo

- Linux alapú mobil operációs rendszer
  - Nokia saját platformja
- Eredetileg „internet tablet”-ekre
  - 2005-ben az első device: Nokia 770
  - Nokia N800, Nokia N810
- Első és jelenleg egyetlen teljes értékű telefon: Nokia N900
- Linux hackereknek ideális

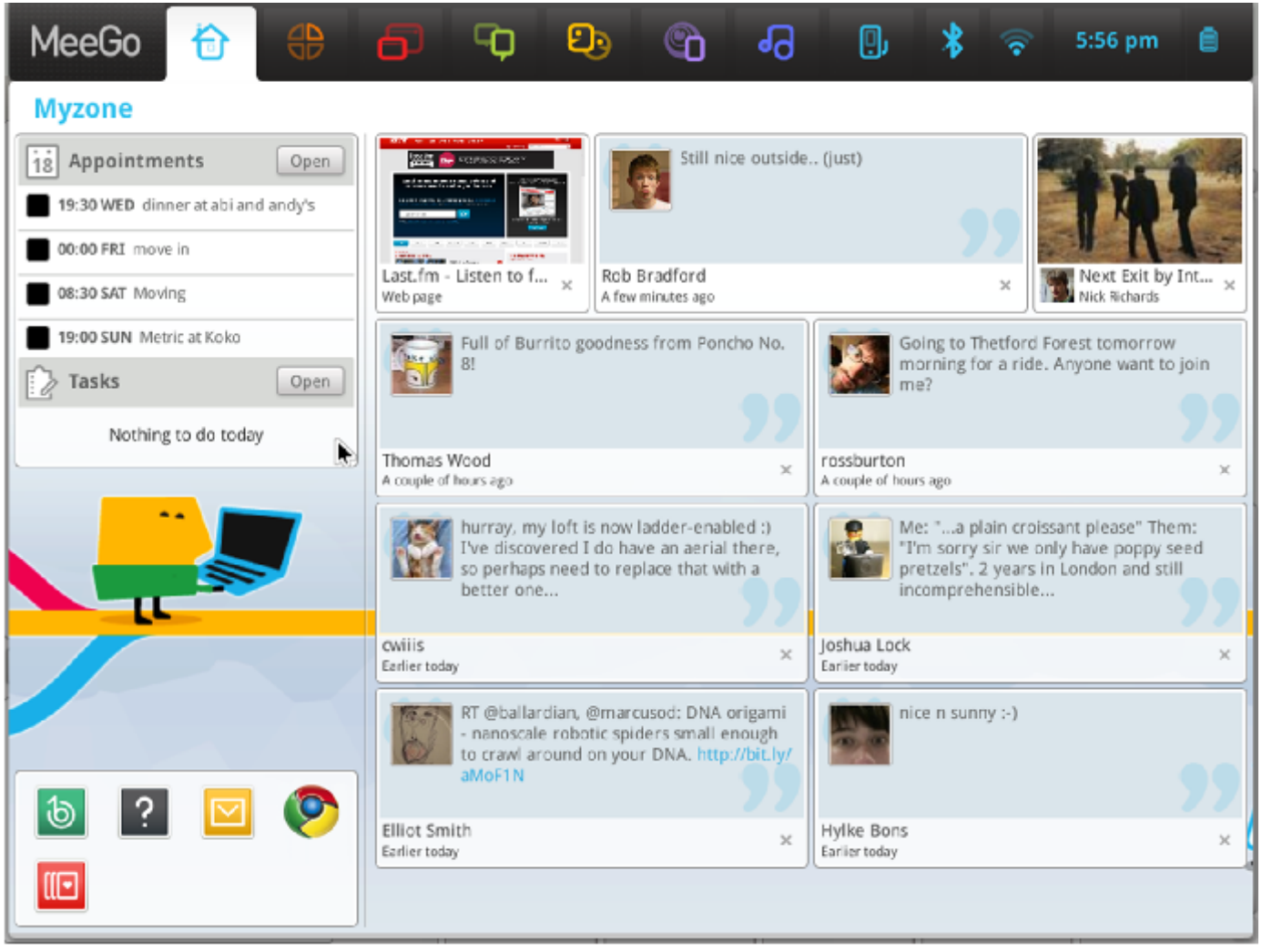
MeeGo

- A Nokia és az Intel közös platformja
- A Maemo utódja
  - Qt alapú fejlesztés
  - Nokia N900-ra már telepíthető (v1.1, lassú)
- Nem csak mobiltelefonokra, több változat:
  - **Handset**
  - Netbook
  - In-Vehicle
  - Connected TV
  - Media phone

MeeGo for Handset screenshots

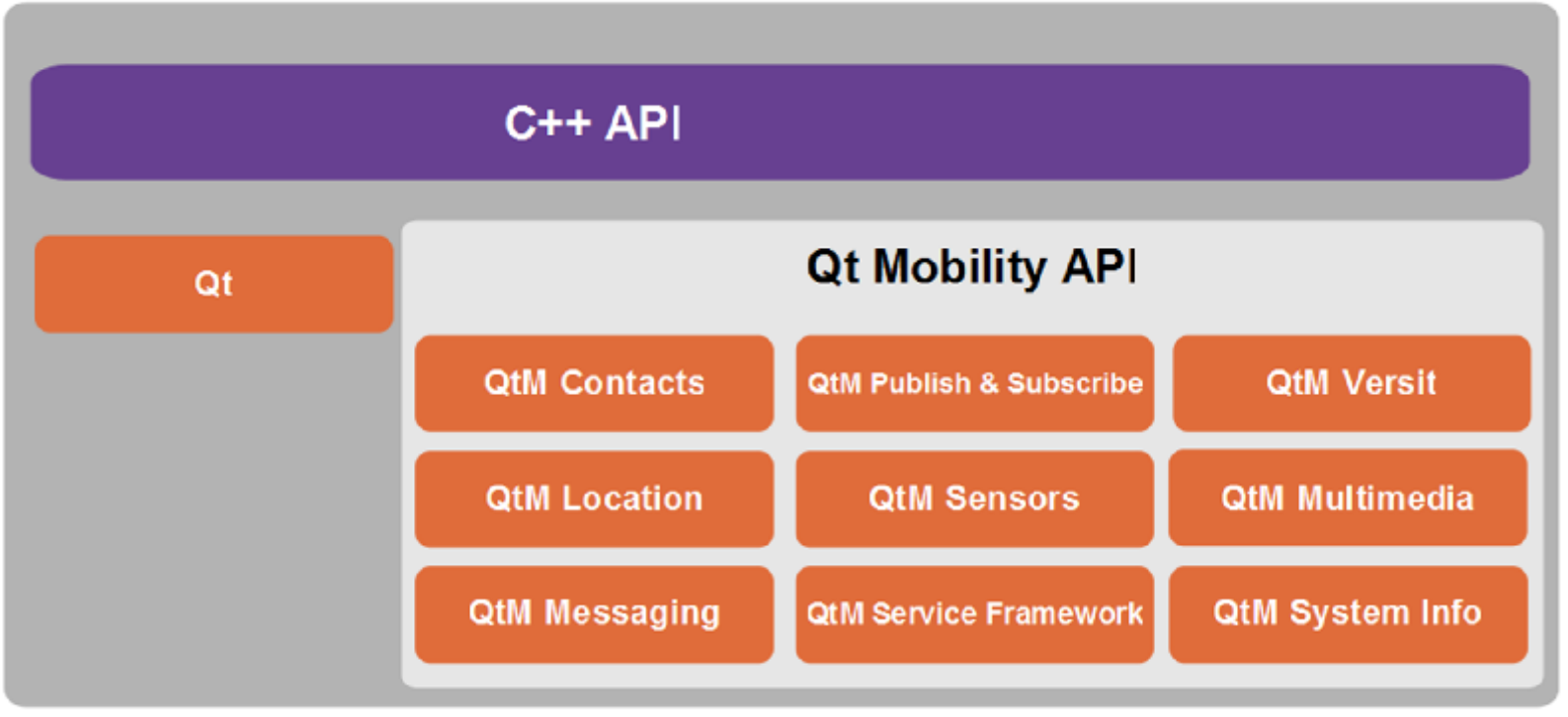


MeeGo for Netbook



Natív fejlesztés MeeGo-ra

- Qt-ban, Nokia Qt SDK használható (Windows, Linux, Mac)



Érdekesség: Anroid apps on MeeGo?

- Myriad Alien Dalvik
  - Az Android által használt Dalvik virtuális gép portja
  - APK-k (Android Package) egy az egyben futtathatók
  - 2011.02.08-as hír

Ellenőrző kérdések, tippek

- Mi a Maemo és a MeeGo?
  - Linux, Intel-Nokia, MeeGo az utód, MeeGo több platforma
- Milyen eszközökre van MeeGo?
- Miben/hogyan fejlesztünk MeeGo-ra?

Symbian

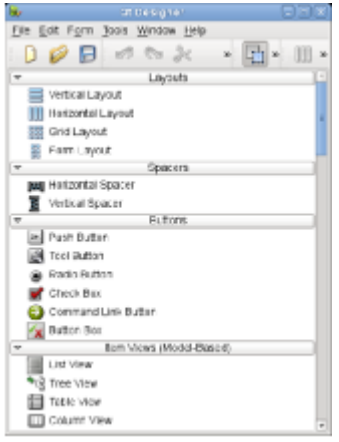
Qt bemutatása

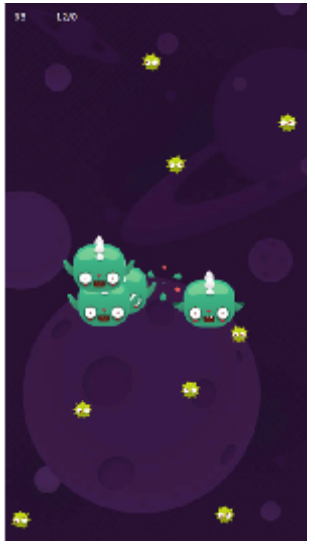
Qt

- Cross-platform C++ osztálykönyvtár (kb. 500 osztály, 4000 metódus)
  - Eleinte csak UI
  - Később kiegészítettek egy általános alkalmazás keretrendszerré
- Tartalmaz néhány nyelvi kiegészítést, melyek túlmutatnak standard C++-on
- Saját deklaratív UI leíró nyelv: Qt Quick (QML)
- Saját fejlesztőeszközök, IDE (is)
- Open-source (GPL + több fajta licenc)

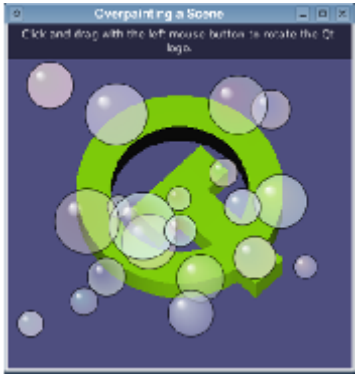
Qt szolgáltatások 1/3

- UI
  - Széles widget választék
  - UI editor
  - Adott platform look-and-feel





- 2D grafika
  - Saját objektum-orientált világmodell, nézetekkel (kb. mini játék-engine)
  - Nagyszámú elem vizualizálható



- 3D grafika
  - Mobilon egyelőre csak Windows CE és Embedded Linux

Qt szolgáltatások 2/3

- Szálkezelés, eseményvezérelt programozás
  - Eseménykezelő ciklus
  - Signals and slots
- Kommunikáció
  - HTTP, FTP, TCP/UDP socketek
- XML
- WebKit
  - Az Apple és a Google által is használt böngészőmotor

Qt szolgáltatások 3/3

- Adatbázis motor
  - Összess főbb adatbázis támogatása
- QtScript
  - Saját ECMAScript (~JavaScript, ActionScript) alapú script motor
  - Objektumok futási időben módosíthatok scripteken keresztül (property-k, illetve signals and slots is)
- Mobilspecifikus funkciók: Qt Mobility

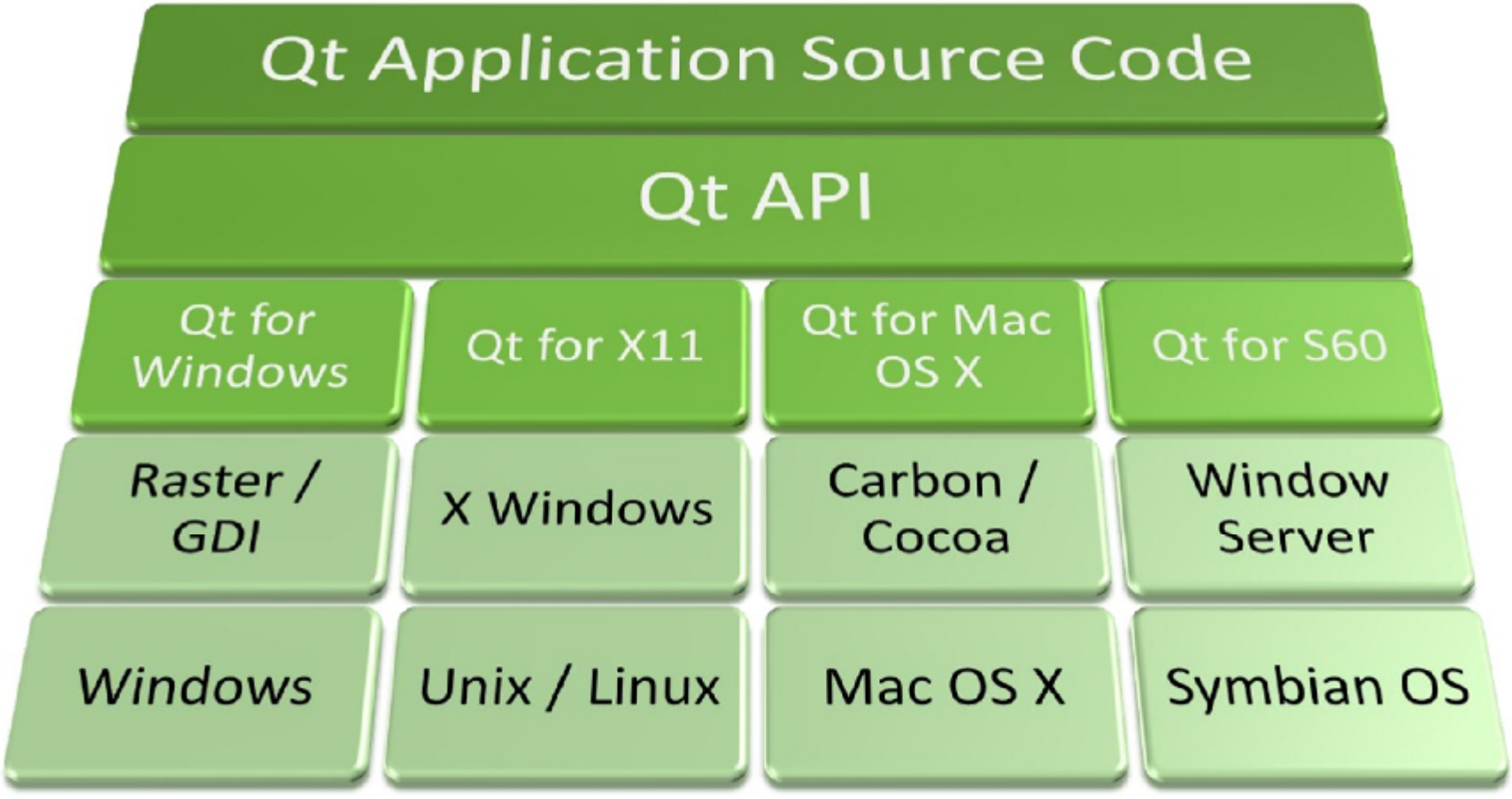
Qt C++ vs. Qt Quick (QML)

- Qt két fejlesztési mód:
  - C++
  - Qt Quick: deklaratív + Javascript-szerű
- Qt Quick
  - Saját deklaratív nyelv: QML
  - Látványos, animált UI készíthető vele
  - C++ tudás nem kell (de keverhető Qt Quick és C++ kód)
  - Qt 4.7-ben jött be (még friss)

Támogatott platformok

- Desktop
  - Windows
  - Linux
  - Mac OS X
- Mobil
  - Symbian
  - Maemo / MeeGo
  - Windows CE (NEM Windows Phone)
- Minden platformra ugyanaz a C++ kód fordul le
- Qt funkcióit egyes platformokon más programozási nyelvekből is el lehet érni (language binding), pl. Java, C#, Python

Qt architektúra



Qt történelem

- 1994: megjelenik az első Qt
  - Két norvég programozó munkája, céljuk: „megalkotni a világ legjobb C++ GUI keretrendszerét”
  - „Q” (tetszett a betű), „t” (toolkit)
  - A fejlesztő cég: Quasar Technologies → Troll Tech → Trolltech → Qt Software
- 1997: Qt 1.2, felhasználják KDE megírásához
  - Qt a Linux GUI fejlesztés egyik de facto standardja
- 1999-2005: Qt folyamatosan bővül, több támogatott platform, funkciók bővítése jelenleg 4-es verziónál tart
- 2008: Nokia felvásárolja a Trolltech-et

Qt-ra épülő alkalmazások

- Adobe Photoshop Elements
- Skype
- Google Earth
- Opera
- KDE
- Mathematica
- VirtualBox

Qt mobil platformokon

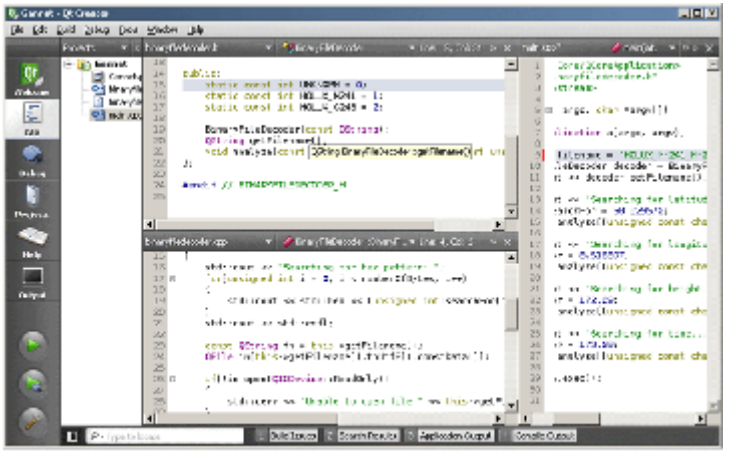
- Qt for Embedded Linux
  - Linux PDA-k
  - Greenphone
- Maemo, MeeGo
  - Nokia Internet tablet (N770, N800, N810)
  - Nokia N900
- Qt for Windows CE
  - Windows CE 5.0/6.0, Windows Mobile 5.0/6.0
  - Fejlesztés Visual Studio-val
- Qt for Symbian

Nokia Qt SDK

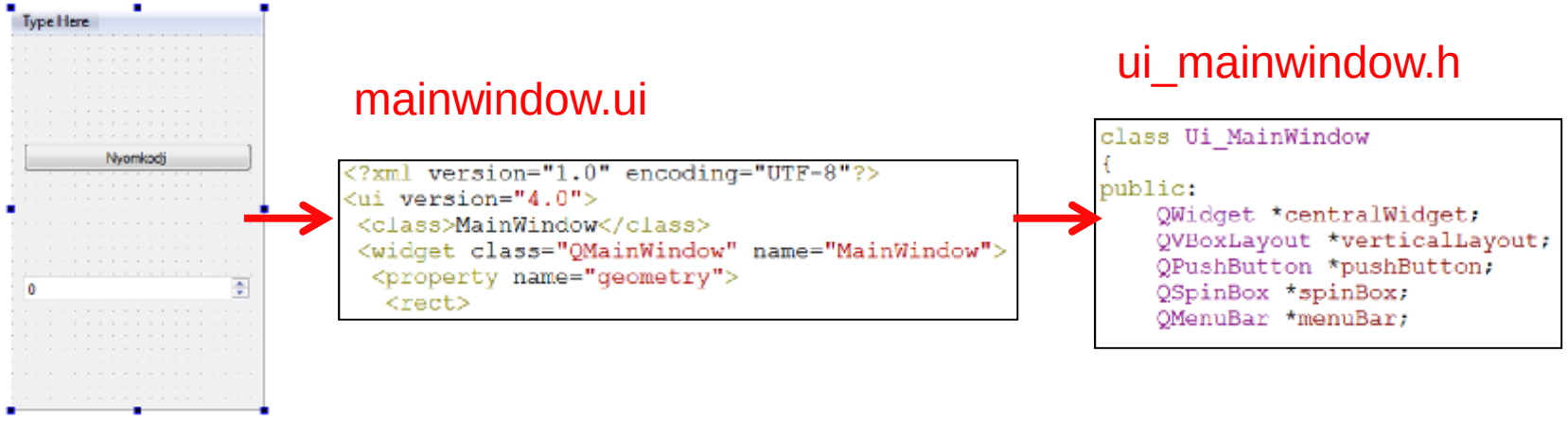
- Minden ami Symbian/Maemo fejlesztéshez kell, egy csomagban
  - Qt Creator IDE
  - Qt Simulator
  - Symbian SDK (emulátor és egyéb sallangok nélkül)
  - Maemo SDK (MADDE)
  - Qt SDK
  - Qt Mobility

IDE: Qt Creator

- Beépített Qt simulator
- Device-ra tud fordítani/debuggolni
- Támogatja az asztali platformra való fejlesztést
- UI design alapján kódot generál



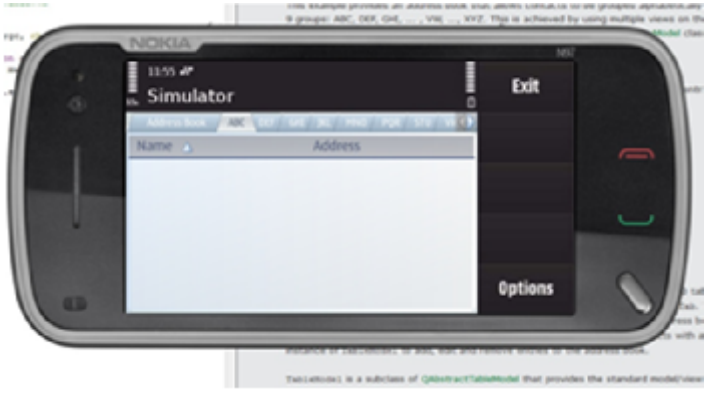
Qt GUI Designer



- Minden form-hoz (widget-hez) egy XML fájl (.ui)
  - Ez alapján kódot generál, fordítás előtt
  - Nem keveredik a generált és a programozó kódja
  - Generált kód változóit, signal/slot-jait elérjük saját kódból

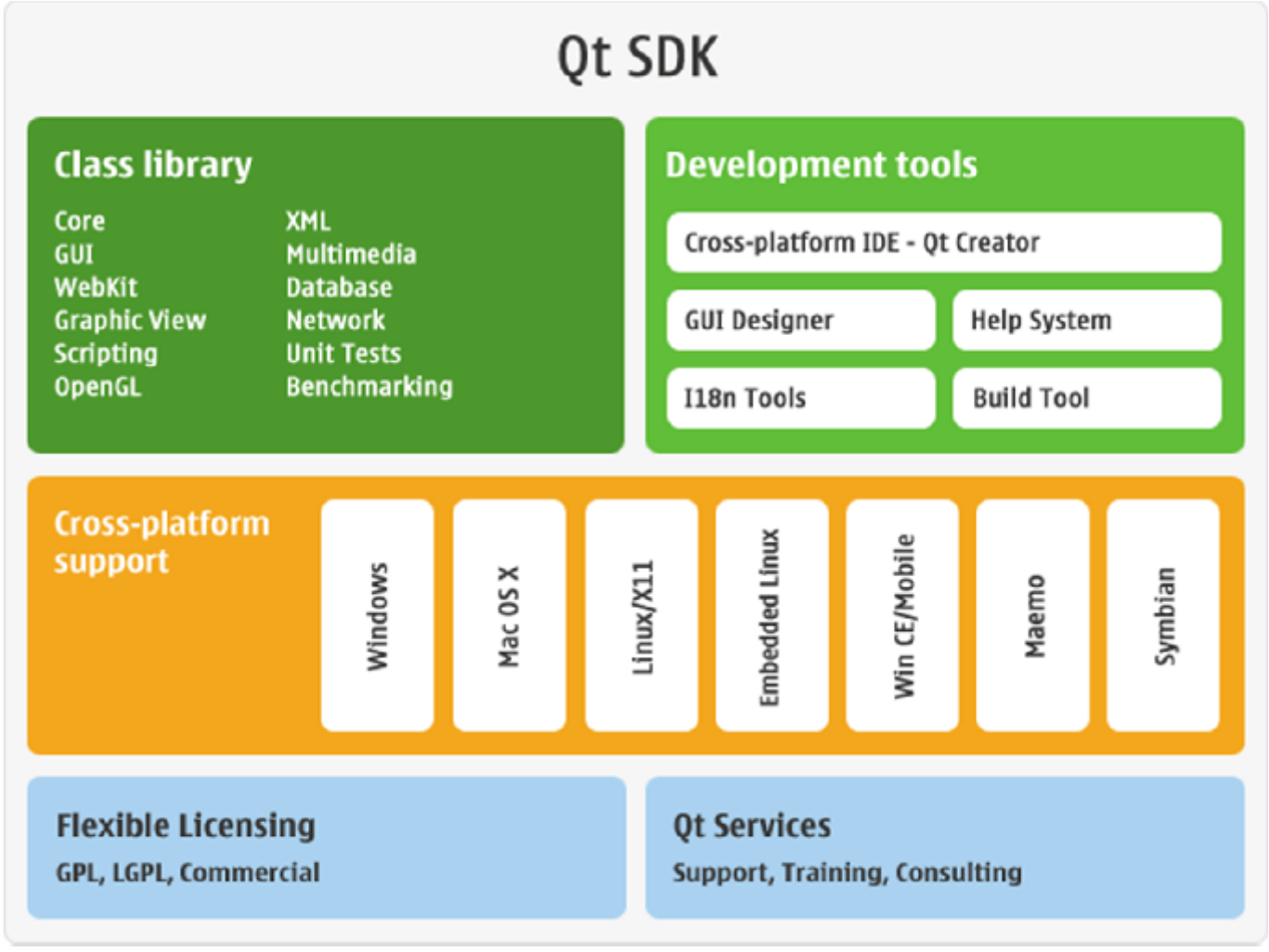
Qt Simulator

- Különálló build target
  - Gyors!
  - Több fajta készüléket támogat
  - Valójában nem emulálja a Symbian/Linux oprendszert



- Támogatja a Mobility API-t
  - A mobilspecifikus paraméterek szimulálhatók és állíthatok
  - Pl. GPS koordináták, akku szint

Qt SDK összefoglalás



HelloWorld

```
#include <QtGui/QApplication>
#include <QtGui/QLabel>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QLabel label("Hello, world!");
    label.showMaximized();

    // Eseménykezelő ciklus indítása
    return app.exec();
}
```

HelloWorld – Osztályok

- QApplication
  - GUI alkalmazásonként egy példány
  - Futtatja az alkalmazás eseménykezelő ciklusát
  - Menedzseli az alkalmazás-szintű erőforrásokat (default betűkészlet, egérkurzor, stb.)
- QPushButton
  - Egy alap gomb widget (QWidget-ből származik)
  - Reagál a felhasználói bemenetre (eseménykezelés: megnyomták), kirajzolja magát

Projektleíró, fordítás

- Platformfüggetlen projektleíró fájl (.pro)
  - Összefogja a projekthez tartozó fájlokat
  - Konkrét, platform specifikus make fájlok ez alapján generálódnak (qmake)
- Fordítás az adott platformra tipikusan GCC-vel (GNU make hívja)
  - Parancssorból vagy Qt Designer-el

Platform független projektleíró

- A projekt adatait (forrásfájlok, elérési utak, library-k) a projektleíró .pro fájl tartalmazza

```
TEMPLATE = app
TARGET = HelloWorld
QT += core \
      gui
HEADERS += Valami.h
SOURCES += main.cpp \
           Valami.cpp
FORMS += mainwindow.ui

symbian:TARGET.UID3 = 0xEDA1FFE5
```

alkalmazás típusa és neve

Qt modulok (~library)

forrásfájlok

GUI designer forok

Projekfájlok létrehozása

- Platform független projekleíró (.pro)
  - Lehetőség van manuálisan megírni
  - Automatikus generálás: qmake –project
    - Végignézi a könyvtárban található forrásfájlokat (.h, .cpp, .ui) és létrehozza .pro fájlt
- Platformspecifikus projektfájlok: qmake
  - Legenerálja a szükséges Symbian projektfájlokat (.mmp, .rss, stb.)

```
qmake –project
qmake HelloWorld.pro
```

Projekt fordítása

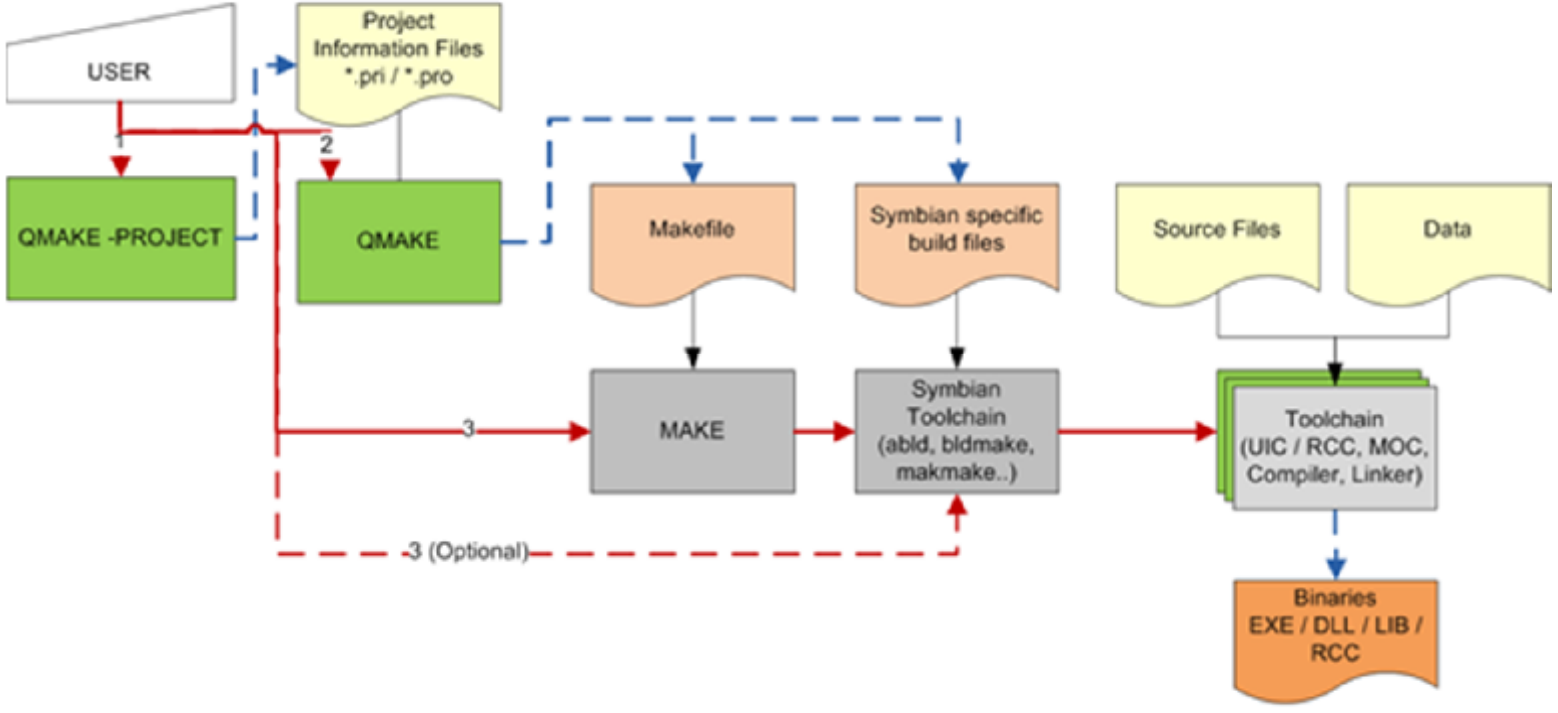
- Előfeltétel: projektfájlok már le vannak generálva qmake-el
- Projekt fordítása standard GNU make-el
  - make target_platform

```
make gcce-release
```

```
make winscw-debug
```

Build toolchain

- A fejlesztő csak make / qmake-et használ
  - Megússzuk a Symbian specifikus tool-okat



Qt alkalmazások telefonon

- A Qt DLL-eknek fenn kell lenniük a telefonon
- Lehetséges módszerek
  - Telepítés egyszer, Qt SDK mellé adott `qt_installer.sis` tartalmaz mindent
  - Alkalmazás telepítő .SIS fájljába ágyazzuk Qt-t (+12 MB)
  - Qt smart installert ágyazzuk az alkalmazás mellé, mely automatikusan letölti a szükséges csomagokat az Internetről

Ellenőrző kérdések, tippek

- Milyen platformokra/mobilplatformokra van Qt?
- Hogyan lesz a Qt projekt platformfüggetlen? Mi a .pro fájl? Mit csinál a qmake?
- Mi a Qt Creator és milyen szolgáltatásokat nyújt? Mik a Qt simulator legfőbb ismérvei?

Symbian

Qt object model

Qt object model

- Funkciók
  - **Signals and slots**
  - **Properties**
  - **Események és esemény szűrők**
  - **Objektum hierarchia (fa-alapú)**
  - Védett pointerek
  - Futás idejű típusazonosítás (RTTI)
  - Internalizáció, nyelvfüggő sztringkezelés
- Legtöbb funkció QObject-re épül

QObject

- Qt objektumok ősosztálya, sok extrával
- Nem adható át érték szerint
  - Egyedi objektumokat reprezentálnak (nem „értékeket”)
  - Nevet adhatunk a példányoknak: QObject::objectName()
  - Copy-constructor nem publikus
  - Másolás helyett „klónozás”: clone()

```
class QObject
{
public:

 QObject(QObject* parent=0);

 QObject* parent() const;

 QString objectName() const;

 void setParent(
   QObject* parent );

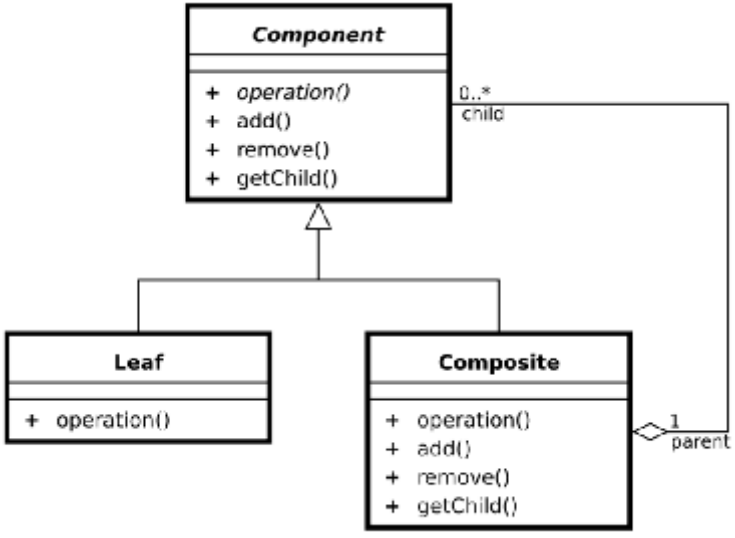
 const ObjectList& children()
   const;

 // ...

};
```

Composite design pattern

- Definíció
  - „Az objektumokat a rész-egész kapcsolatokat definiáló fastruktúrába szervezzük. Ebben az esetben a *Composite* tervezési minta lehetőséget ad arra, hogy a kliens mind az önálló, mind az összetett objektumokat egyformán, teljesen átlátszóan kezelje.”



- Tipikus felhasználási terület
  - GUI-t használó alkalmazásoknál a felhasználó felület elemeinek csoportosítására, tartalmazási viszony kifejezésére

QObject hierarchia

- Fa-alapú hierarchia:
  - Minden objektumhoz maximum egy szülő
  - Tetszőleges számú gyereke lehet
- Egy QObject gyerekei törlődnek a szülő törlésekor
- Gyerek törlésekor automatikusan kikerül a szülő gyerekei közül

QObject lekérdezések

- Gyerek objektumok megtalálása
  - **`QList<T> parentObj.findChildren<T> ( const QString& name) const;`**
  - T típusszűrő, ha name üres sztring, akkor visszatér az összes T típusú gyerekkel
- Példa, az összes „Product” típusú gyerek lekérdezése

```
QList<Product*> childlist =
  parent.findChildren<Product*>();

foreach (Product* current, childlist) {
  qDebug() << current->toString();
}
```

QWidget

- Minden UI elem ősosztálya
- Téglalap alakú terület a képernyőn
- Eseményeket kezel
  - Billentyű- és egéresemények, stb.
- QObject-ből származik
  - Egy szülője és több gyereke lehet (QObject!)
  - Szülő nélküli widget: Window
  - Widgetnek csak widget lehet a szülője!
- Widget és gyerekeinek kirajzolása: show(), showMaximized()

QWidget – QObject hierarchia

```
QMainWindow* mainWindow =
  new QMainWindow();
QPushButton* button =
  new QPushButton(mainWindows);
QLabel* label = new Qlabel(mainWindow);
QScrollArea* scroll =
  new QScrollArea(mainWindow);
QLabel* bigLabel = new Qlabel(scroll);

/…

delete mainWindow; // gyerekek törlődnek
```

- Mi történik ha egy gyerek elemet törlünk?

Óvatosan!

- Stack-en létrehozott objektumokkal csak óvatosan

```
{
  QObject* parent = new QObject();
  QObject child(parent);
  delete parent; // child is törlődik
} // child-ot újra törli a runtime a stack
  visszagörgetésekor, FATAL ERROR
```

Ellenőrző kérdések, tippek

- QObject-ek „egyedi objektumok”
- Objektum hierarchia (szülő gyerek)
  - Composite design pattern
  - Hogyan történik a memória felszabadítása? (szülő gyerekei)
  - Mi történik ha törlünk egy gyerek elemet? (szülőből is kikerül)
- Mi az a QWidget?
- Mi köze a QWidget-nek a QObject-hez? Milyen QObject tulajdonságt használ ki QWidget?

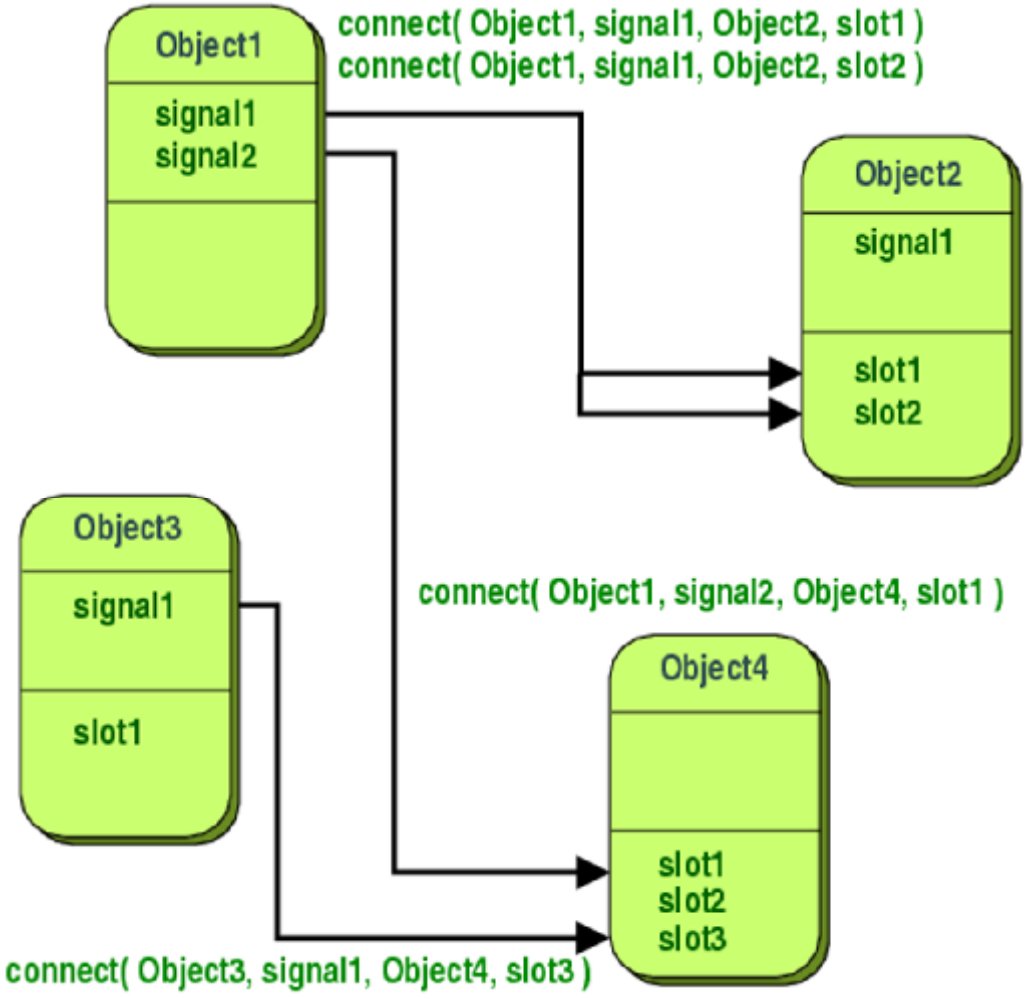
Observer design pattern

- Definíció
  - Objektumok az állapotuk megváltozásáról szeretnék értesíteni egymást anélkül, hogy az függjön a konkrét osztálytípustól
- Lehetséges megvalósítások:
  - Standard C++: interfész öröklés
  - C: callback (függvény mutató)
  - .NET: delegate
  - Qt: Signals and slots

Signals and slots

- Qt által használt speciális mechanizmus az eseménykibocsátó és -feldolgozó objektumok összekapcsolásához
- Signal: egy eseményforrás (üzenet, jelzés)
  - Egy adott osztályhoz/objektumhoz tartozik
  - Egy osztályhoz több signal is tartozhat
  - „Ki lehet bocsátani” (ez jelenti az esemény bekövetkeztét)
- Slot: egy eseményfeldolgozó
  - Egy objektum metódusa, melyet meg szeretnénk hívni adott esemény bekövetkeztekor
- A signal-ok összekapcsolhatók a slot-okkal
  - A „slot” meghívódik a „signal” kibocsátásakor

Signals and slots



S&S példa – Osztály deklaráció

- Egyszerű Qt számláló osztály

```
#include <QObject>


class Counter : public QObject
{
Q_OBJECT

public:
  Counter() { m_value = 0; }
  int value() const { return m_value; }

public slots:
  void setValue(int value);

signals:
  void valueChanged(int newValue);

private:
  int m_value;
};
```

Qt példa – Kibocsátás és feldolgozás

- Signal kibocsátása

```
void Counter::setValue(int value)
{
  if (value != m_value) {
    m_value = value;

    emit valueChanged(value);
  }
}
```

- Eseménykibocsátó és –feldolgozó összekapcsolása

```
Counter a, b;
QObject::connect(&a, SIGNAL(valueChanged(int)),
  &b, SLOT(setValue(int)));

a.setValue(12); // a.value() == 12, b.value() == 12
b.setValue(48); // a.value() == 12, b.value() == 48
```

S&S – További tulajdonságok

- Typesafe (signal és slot szignatúrájának meg kell egyeznie)
- Egy signal tetszőleges számú slot-hoz köthető és egy slot-hoz tetszőleges számú signal kapcsolható
  - Lehetőség van egy signal egy másikhoz kötésére, ilyenkor az első signal kibocsátásakor a második is kibocsátásra kerül
- Signals and slots használatának előfeltételei
  - Objektumok közvetlenül vagy közvetetten QObject-ből származzanak
  - Osztálydeklaráció Q_OBJECT direktívával kezdődjön

S&S – További tulajdonságok 2

- Paraméterek neve nem adató meg

```
connect( m_slider, SIGNAL( valueChanged(int value)),
         this, SLOT( setValue( int newvalue )))
```

- Egy signal hozzákapcsolható egy kevesebb paraméterrel rendelkező slothoz

```
connect( m_slider, SIGNAL( valueChanged(int)),
         this, SLOT( updateUI()))
```

- Nem köthetünk egy signalt olyan slothoz, mely több paraméterrel rendelkezik

```
connect( but, SIGNAL( clicked() ) )
         this, SLOT( setText(const QString&)))
```

Példa

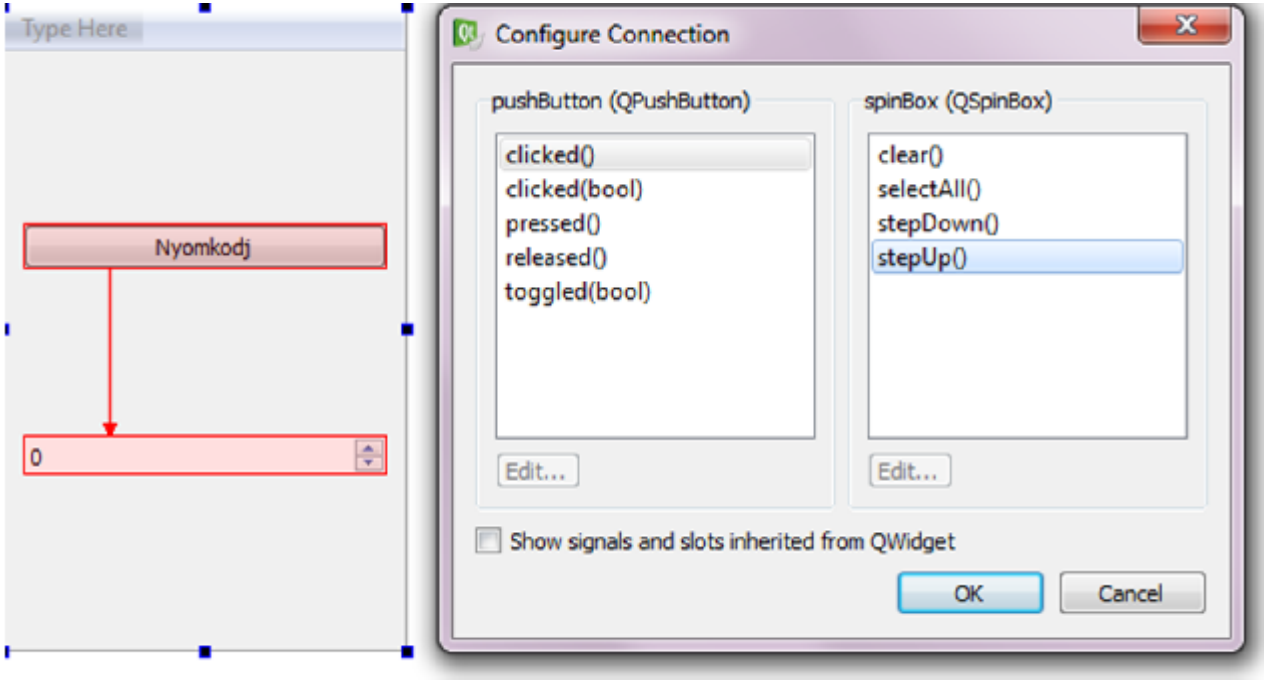
- Gombnyomásra számláló növelés

```
QPushButton* pushButton = …
QSpinBox* spinBox = …

QObject::connect(
   pushButton, SIGNAL(clicked()),
   spinBox, SLOT(stepUp()));
```

Ugynez Qt Creatorban

- (ugyan az a kód generálódik hozzá)



Qt Properties

- Hozzáférés egy objektum (QObject) „tulajdonságaihoz”
  - Nem sima tagváltozók! Csak írható/csak olvasható, stb.
  - C++ setter/getter metódusokon túlmutató funkcionalitás
  - Platformfüggetlen
- QObject-hez akár fordítási időben, akár dinamikusan hozzárendelhetünk property-ket!
- Futás közben, dinamikusa
  - Lekérdezhetők, pl. név alapján
  - Módosíthatók
  - Enumeráció típus elemei is módosíthatók
- Használatos: animáció, scriptelés

Properties példa - Deklaráció

```
class MyClass : public QObject
{
Q_OBJECT
Q_PROPERTY(Priority priority READ priority WRITE setPriority)
Q_ENUMS(Priority)

public:
  MyClass(QObject *parent = 0);
  ~MyClass();

  enum Priority { High, Low, VeryHigh, VeryLow };

  void setPriority(Priority priority);
  Priority priority() const;
};
```

Meta Object Compiler (MOC)

- Qt által használt C++ kiegészítések megvalósítását végzi
  - pl. Signals and slots, metaobjektum, properties
- Forrásfájlokat dolgozza fel (szövegfeldolgozó)
  - Végignézi a .cpp fájlokat és ha olyan osztályt talál, melyben Q_OBJECT direktíva szerepel, abból standard C++ forrást generál
  - osztály.cpp -> moc_osztály.cpp

Ellenőrző kérdések, tippek

- Hogy működik a signals and slots? (kódolni is kell) Mik az előnyei?
  - Lehet-e egy signalhoz több slotot, több slothoz egy signalt?
  - Teljesen egyeznie kell a signal és a slot signatúrájának?
  - Milyen osztályoknak lehetnek signaljai/slotjai (QObject leszármazottaknak…)?
  - Milyen design pattern valósítható meg vele (observer)
- Mi az a MOC és mit csinál?

AMORG APPLIED MOBILE RESEARCH GROUP

Symbian, Qt

Mobilszoftverek

S4

S4 – Tartalom

- Qt Mobility
- Qt Quick

Qt

Qt Mobility

Qt Mobility

- Mobilspecifikus funkciókat tömörítő API-k
- Sok funkció desktop operációs rendszeren is használható
  - Messaging, Multimedia, Publish and Subscribe, …
- Jelenleg (2011 február) Qt Mobility 1.1.0

Mobility 1.1. tartalma (1/2)

API/Framework	Leírás
Bearer Management	A telefon hálózati kapcsolat menedzselő API
Contacts	Contact (címtár) adatok lekérése helyi vagy távoli tárolóból
Document Gallery	Dokumentumok metaadat alapján történő keresése, böngészése
Feedback	A vibramotor és a piezo (tactile) rezgő visszajelzések kezelése
Location	Helymeghatározáshoz, POI-k kezeléséhez, térkép és navigáció kezeléséhez szükséges gyűjtemény
Messaging	A különböző üzenetkezelési lehetőségek elérése (SMS, MMS, e-mail)

Mobility 1.1. tartalma(2/2)

API/Framework	Leírás
Multimedia	Fotó, audio, video felvétele, visszajátszása
Organizer	Naptár, ütemezés, személyes adatok (feljegyzések, napló/blog) elérése lokális vagy távoli szolgáltatóktól
Publish and Subscribe	A Publish and Subscribe API, a Value Space segítségével, lehetővé teszi különböző célra értékek eltárolását, visszaolvasását, illetve feliratkozást az érték megváltozására.
Qt Service Framework	Különböző célú szolgáltatások felfedezésére és használatára szolgáló általános API
Sensors	A készülék számos szenzor eleméhez való hozzáférést lehetővé tevő API
System Information	A futtató rendszerrel kapcsolatos információk, illetve képességek (capabilities) lekérdezése
Versit	vCard and vCalendar formátumú adatok kezelése

Mobility API a projektfájlokban

.pro

```
TEMPLATE = app
TARGET =
DEPENDPATH += .
INCLUDEPATH += .
CONFIG += mobility
MOBILITY = systeminfo
# Input
SOURCES += main.cpp
```

- Forrásfájlokba:

```
#include <QSystemInfo>

QTM_USE_NAMESPACE
```

API-k kicsit részletesebben

- Messaging
- Organizer
- Sensor
- System Info
- Location
- Contacts

Messaging API - Üzenetkezelés

- Az üzenetkezelő szolgáltatásokhoz férhetünk hozzá
  - Üzenetek keresése és rendezése
  - Készítés, módosítás
  - Küldés-fogadás
  - Meghatározott üzenetkezelő alkalmazás indítása
  - Egy meglévő üzenetre való reagálás
- Egységes kezelőfelületet biztosít
  - SMS
  - MMS
  - Email

Az üzenetkezelés főbb osztályai

- Üzenetek összeállítása, kezelése
  - QMessage
  - QMessageAddress
- Üzenetfiókok elérése
  - QMessageAccount
  - QMessageFolder
- Rendezés és szűrés
  - MessageManager
  - QMessageSortOrder
  - QMessageFilter
- Üzenetkezelő szolgáltatásokhoz való hozzáférés
  - QMessageService

Üzenetek lekérése a QMessageStore-ból

```
// Az “Incoming” flaggel ellátot üzenetek lekérdezése
// (További státuszok: Read, HasAttachment, Removed)
QMessageFilter filter(QMessageFilter::byStatus(QMessage::Incoming));
// Érkezés alapján csökkenő sorrendbe helyezzük őket.
QMessageOrdering ordering(
  QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder));


const int max = 100;
QMessageIdList matchingIds(
  QMessageStore::instance()->queryMessages(filter, ordering, max));
```

Üzenet összeállítás és küldés

- Üzenet küldése

```
// Az üzenet küldésre kész, így létrehozunk egy service objektumot, majd küldünk
QMessageService *m_service = new QMessageService( this );
if (!m_service->send(message) ) {
        QMessageBox::warning(0, tr("Failed"), tr("Nem tudtam elküldeni az
    üzenetet"));
```

- Üzenet összeállítása az alapértelmezett composerben

```
// Az alapétrelmezett composer meghívása a már létező üzenettel
QMessageService *m_service = new QMessageService( this );
m_service->compose(message);
```

Organizer API

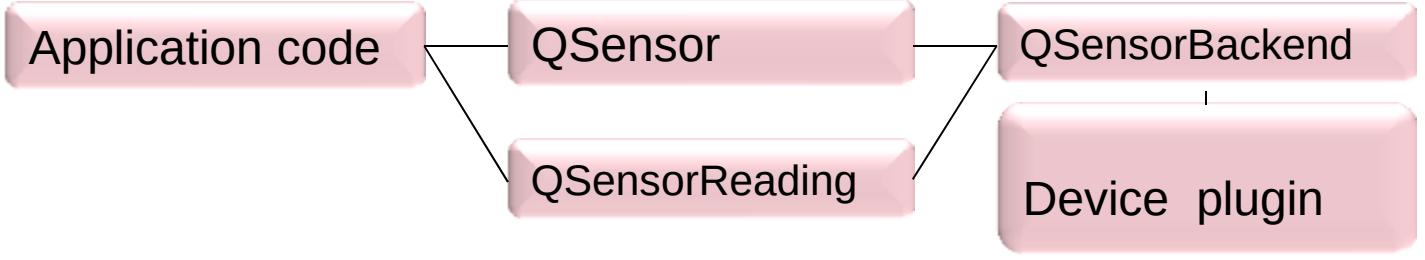
- Hozzáférés a készülék naptárjához, ütemezett eseményekhez, személyes adatokhoz
- Platform-függetlenül és a konkrét tárolási mód ismerete nélkül
- Az API által támogatott elemek:
  - Events
  - Journals
  - ToDo
  - Notes
- Az egyes elemek detail-ekből (részinformációkból) állnak, amik külön megadhatóak

Naplóbejegyzés készítése

```
QOrganizerJournal journal;
journal.setDescription("A megbeszélésen bullshit bingo-
  t játszottunk. Majdnem nyertem, majd legközelebb.");
defaultManager.saveItem(&journal);
```

Sensor API

- Az API hozzáférést biztosít mind a push, mind a poll jelleggel működő szenzorokhoz
- QSensor a leszármazott osztályaival együtt biztosítja a szenzor adatokhoz való hozzáférést
  - Minden szenzorhoz egy QSensor-leszármazott
  - A szenzor adatok kiolvasása QSensorReading (-vagy leszármazottjaival)
  - Szenzor adatok megváltozását a QSensorFilter (-vagy leszármazot) interfészek implementálásával kapjuk
- A leszármazottak mindig kényelmi osztályok



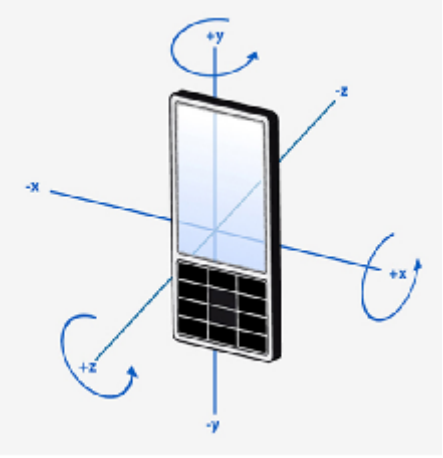
QSensor leszármazottak

Class	Description
QAccelometer	Gyorsulásszenzor x,y,z tengelyek mentén
QAmbientLightSensor	Környező fény erősségét mérő szenzor
QCompass	Iránytű
QMagnetometer	Magnetometer
QOrientationSensor	A orientációját adja meg
QProximitySensor	Közelség szenzor (általában pl. arra, hogy a fülünknél tartjuk-e a telefont)
QRotationSensor	x,y,z – tengely mentén való elforgatás mértéke
QTapSensor	érintés/koppintás események 6 irányból

A gyorsulás szenzor használata

```
// start the sensor
QSensor sensor("QAccelerometer");
sensor.start();

// mért érték kiolvasása
QSensorReading *reading =
  sensor.reading();
qreal x =
  reading->property("x").value<qreal>();
qreal y =
  reading->value(1).value<qreal>();
```



System Information API

- API készlet rendszerinformációk, illetve -képességek lekérdezésére
- Támogatott hardver elemek megléte: kamera, bluetooth, GPS, FM radio stb.
- Mobilhálózat információk
  - infó a kapcsolatról, térerőről
  - A hálózat típusa
- Kijelző és háttértár információk
  - Beépített vagy eltávolítható memória
  - Kijelző típus
- Készülékkel kapcsolatos információk
  - Telep információk
  - Aktív profil
  - SIM kártya(IMEI stb.)
  - Beviteli módok (key, keypad, qwerty, touch screen, multitouch stb.)

System Information API osztályok

Class	Description
QSystemDeviceInfo	Készülék infó (telep, power state, beviteli módok, IMEI, gyártó, profil, stb.)
QSystemDisplayInfo	Kijelző infó (színmélység, fényerő)
QSystemInfo	Általános információk (nyelv, szoftver verzió, stb.)
QSystemNetworkInfo	Hálózat információk (mobilháló neve, típusa, térerő, …)
QSystemScreenSaver	Képernyővédő elérése
QSystemStorageInfo	Memória és háttértár infók (meghajtó típusok, szabad hely, …)

Contacts – Példa

- Egy új kapcsolat létrehozása

```
QContactManager*  contactManager  =  new  QContactManager(  this  );
QContact  homer;

// Név megadása
QContactName  name;
name.setFirst("Homer");
name.setLast("Simpson");
homer.saveDetail(&name);

// Telefonszám
QContactPhoneNumber  number;
number.setContexts(QContactDetail::ContextHome);
number.setSubTypes(QContactPhoneNumber::SubTypeMobile);
number.setNumber("555112233");
homer.saveDetail(&number);

// Elmentés a telefonkönyvbe
contactManager->saveContact(&homer);
```

Location API – Helymeghatározás

- A helyszín információk műholdas vagy egyéb segítséggel jönnek létre, mint pl.
  - GPS (A legpontosabb metódus, amennyiben a műhold adatok rendelkezésre állnak)
  - Cella ID (Közelítő adat a látható cellák alapján)
- Az API a következő információkat nyújtja:
  - Földrajzi szélesség és hosszúság
  - A mérés ideje
  - A készülék sebessége
  - A készülék magassága és haladási iránya (északhoz képest)

Lokáció adatforrás

- Location adatforrások a QGeoPositionInfoSource leszármazottjának példánuosításával történnek, a QGeoPositionInfoSource::positionUpdated() signal által küldött QGeoPositionInfo objektumokkal
- A kliensek feliratkozhatnak a folyamatos adatközlésre a positionUpdated() signal segítségével, valamint a startUpdates() vagy requestUpdate() hívásaival
- Biztosított egy alapértelmezett forrást, amelyet a QGeoPositionInfoSource::createDefaultSource() hívásával érhetünk el
- A QGeoAreaMonitor osztály segítségével monitorozni tudjuk, ha a készülék egy adott helyet elhagy vagy megközelít megadott sugárban

Location – Példa

- Pozíció változás értesítésre való feliratkozás

```
// Default pozíció szolgáltató (pl. GPS) lekérése
QGeoPositionInfoSource  *source  =
   QGeoPositionInfoSource::createDefaultSource();
if  (source)  {
   // Pozíció változás figyelése
   connect(source,  SIGNAL(positionUpdated(QGeoPositionInfo)),
   this, SLOT(handlePositionUpdated(QGeoPositionInfo)));
   source->startUpdates();
}

//  Saját slot az értesítések kezeléséhez
void  handlePositionUpdated(const  QGeoPositionInfo  &info)
{
   double  latitude  =  info.coordinate().latitude();
   double  longitude  =  info.coordinate().longitude();
};
```

Ellenőrző kérdések, tippek

- Soroljon fel a Qt Mobility moduljai közül legalább 5-10-et és egy mondatban ismertesse mire jó az API!
- Mire jó a Messaging API? Milyen típusú üzenetek elküldésére alkalmas?
- Melyik API-val tudunk rendszerinformációkat lekérdezni? Milyen információkat kaphatunk így meg?
- Soroljon fel legalább 4 szenzor típust, amit a Sensor API támogat!
- Mire jó az Organizer API? Milyen elemeket támogat?
- Milyen módon tudjuk lekérdezni a telefon aktuális pozícióját?

Qt

Qt Quick

Qt Quick

- QML nyelv
  - Magas szintű szkriptnyelv
  - Deklratív UI leírás, animáció, effektek, interakció
  - Vezérlés/logika: JavaScript (egyszerűbb mint C++)
- QDeclarative modul: kapcsolat C++ és QML között
  - QML dokumentumok betölthetők és megjeleníthetők C++-ban
  - QML kibővíthető C++-ban
- QML dokumentumokat a QML runtime „futtatja” dinamikusan

Hello Qt Quick

```
import QtQuick 1.0
Rectangle {
        width: 200
        height: 200
        Text {
                text: "Mobilszoftverek"
                font.pointSize: 10
                font.bold: true
                anchors.centerIn: parent
        }
}
```

Hello Qt Quick + animáció

```
import QtQuick 1.0
Rectangle {
        width: 200
        height: 200
        Text {
                text: "Mobilszoftverek"
                font.pointSize: 10
                font.bold: true
                anchors.centerIn: parent

                PropertyAnimation on rotation {
                        to: 360
                        duration: 1000
                        loops: Animation.Infinite
                }
        }
}
```

QML

- **Q**t **M**eta-Object **L**anguage
- Magas szintű szkriptnyelv
- Objektum = elem (element)
- Saját, újrafelhasználható komponensek is írhatók

QML objektumok

- Objektumok típusa: nagy kezdőbetű
  - Rectangle, Text, Item, Image, stb.
- Objektum létrehozása a típus kiírásával
  - Név után kapcsos zárójelben az objektumhoz tartozó információk (pl. propery-k)

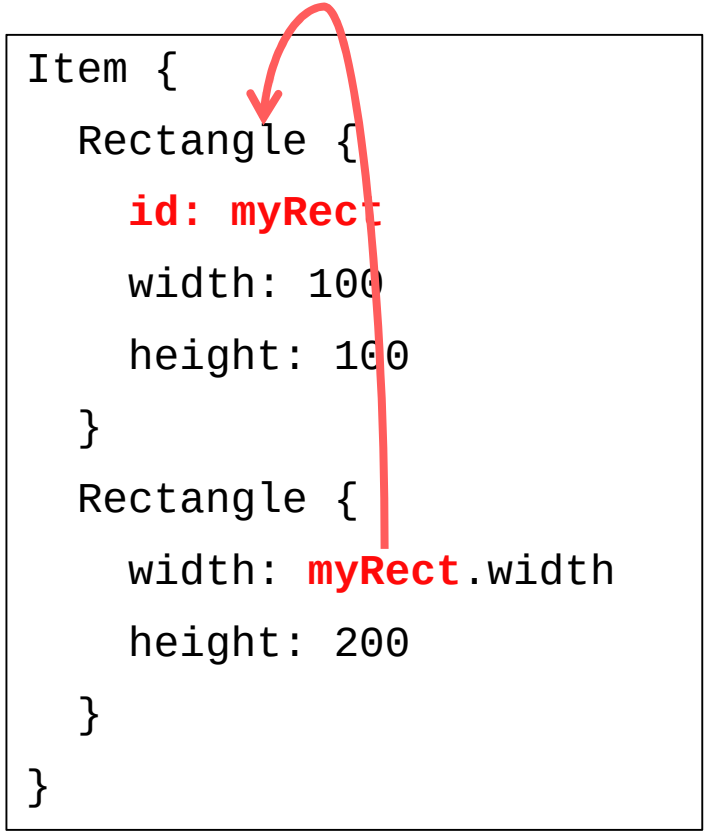
```
Rectangle {
        width: 200; height: 200
        Text {
                text: "Mobilszoftverek"
                font.pointSize: 10
        }
}
```

Property

- Objektum attribútum
- Nevük kisbetűvel kezdődik
- Értékek vagy JavaScript kifejezések rendelhetők hozzá
  - `text: "Mobilszoftverek"`
  - `rotation: 3 * 120`
- Typesafe: csak azonos típusú érték rendelhető hozzá
  - Beépített típusok: int, string, bool, list, stb.

Objektumok azonosítása

- id property
  - Az objektumot azonosítja
  - Ezen a néven hivatkozhatunk az objektumra
  - Egy QML dokumentumon belül egyedi kell, hogy legyen
  - Minden QML objektum rendelkezik id-vel



Property binding

- A propertyk értéke egy JavaScript kifejezés
  - `color: { if (width > 10) "blue"; else "red" }`
  - `height: otherItem.height`
  - `width: 2`
- Ha a JavaScript kifejezés megváltozik, a property értéke is automatikusan újra kiértékelődik!
- Értékadás != binding
  - Property értékadás JavaScript kódból törli a bindingot

```
Rectangle {
  width: otherItem.width
  Component.onCompleted: {
  width = 13; // binding
  }
}
```

Eseménykezelés

- Signal handler: eseménykezelő
  - A signal handler-hez rendelt JavaScript kód lefut az esemény bekövetkeztekor
- Mindig „on”-al kezdődik a nevük

```
MouseArea {
     onPressed: console.log("mouse button pressed")
 }
```

- Rejtett paramétereket is kaphatnak

```
MouseArea {
     onPressed: if (mouse.button == Qt.RightButton)
                    console.log("Right mouse button pressed")
}
```

QML dokumentum

- Egy QML komponenst definiál
- Fájlban (.qml), vagy memóriában
- Mindig UTF-8 kódolású
- Mindig kell legalább egy import direktíva
  - import QtQuick 1.0
  - Csak a névfeloldásban segít (nem „emel be” kódot)
- Egy dokumentumon belül minden id azonosító egyedi névvel rendelkezik

QML és C++ - QML futtatás

- QDeclarativeEngine
  - QML feldolgozás, finomhangolás, kapcsolat C++ és QML dokumentum között
  - Nem jeleníti meg a dokumentumot
  - Mindig van belőle egy a háttérben
- QDeclarativeView
  - QWidget leszármazott, megjelenít egy QML dokumentumot
  - Elfedi QDeclarativeEngine-t
  - Beágyazható más, standard Qt widgetek közé
  - Teljesen QML alapú alkalmazásnál az egyetlen C++ widget

QML alapú alkalmazás

```
#include <QApplication>
#include <QDeclarativeView>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QDeclarativeView view;
    view.setSource(QUrl::fromLocalFile("application.qml"));
    view.show();

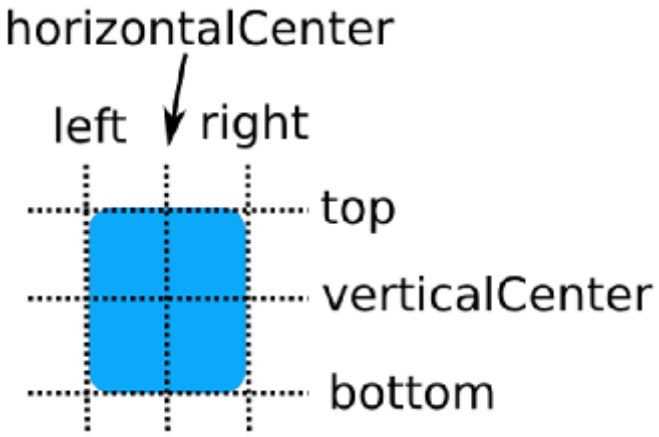
    return app.exec();
}
```

QML elemek

- Alapelem: Item
- Grafika
  - Image, Rectangle, Gradient, …
- Szövegkezelés
  - Text, TextInput, TextEdit, …
- Egér és érintőképernyő
  - MouseArea, Flickable, GestureArea, …
- Animáció
  - PropertyAnimation, Transition, …
- Segédelemek
  - Timer, Loader
- Részecske (particle) effektek, adatkezelés, nézetek (Views), állapotok (States), …

Anchor-based layout

- Minden QML elemnek van 7 „segédvonala” (anchor)
- Egy elem segédvonalai más elemek segédvonalaihoz igazíthatók




```
Rectangle { id: rect1; x: 0; ... }
Rectangle { id: rect2; anchors.left: rect1.right
            anchors.right: rect3.left; ... }
Rectangle { id: rect3; x: 150; ... }
```

Komponensek

- Komponens:programozó által definiált, újrahasznosítható QML elem
  - Külön fájl: <Komponens_neve>.qml
  - Egyetlen gyökérszintű QML elemet tartalmazhat
    - A gyökérelem property-jai az új komponens property-jaiként elérhetők
  - Az azonos könyvtárban lévő QML dokumentumok használhatják: „Komponens_neve”
  - Saját property-k, metódusok, signal-ok

Box.qml

```
import QtQuick 1.0


Rectangle {
        width: 100
        height: 100
        color: "blue"
}
```

Komponens példa

Box.qml

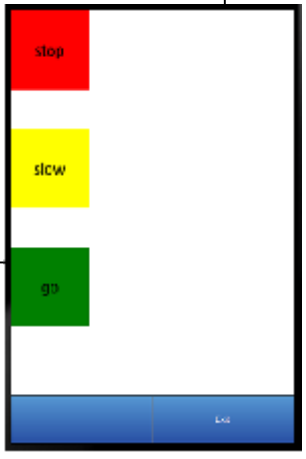
```
import QtQuick 1.0

Rectangle {
  property alias text: myText.text
  width: 100
  height: 100
  color: "blue "
  Text {
    id: myText
    anchors.centerIn: parent
     }
}
```

myApplication.qml

```
import QtQuick 1.0

Rectangle {
  width: 100; height: 400;
  Box { x: 0; y: 0
    color: "red"; text: "stop" }
  Box { x: 0; y: 150
    color: "yellow"; text: "slow" }
  Box { x: 0; y: 300
    color: "green";
    text: "go" }
 }
```



Állapotok (states)

- A QML objektumhoz állapotokat rendelhetünk
  - Mindig egy állapot aktív
  - Mindig van egy alapértelmezett (default) state
  - Pl. Button: On, **Off**
- Az állapothoz egy konfiguráció tartozik, ami például:
  - Beállíthatja property-k értékét
  - Elindíthat egy animációt
  - Elrejthet/megjeleníthet UI elemeket
- Állapotváltáskor az állapothoz tartozó konfiguráció lép érvénybe
- Az állapotváltás animálható

Állapotok definiálása

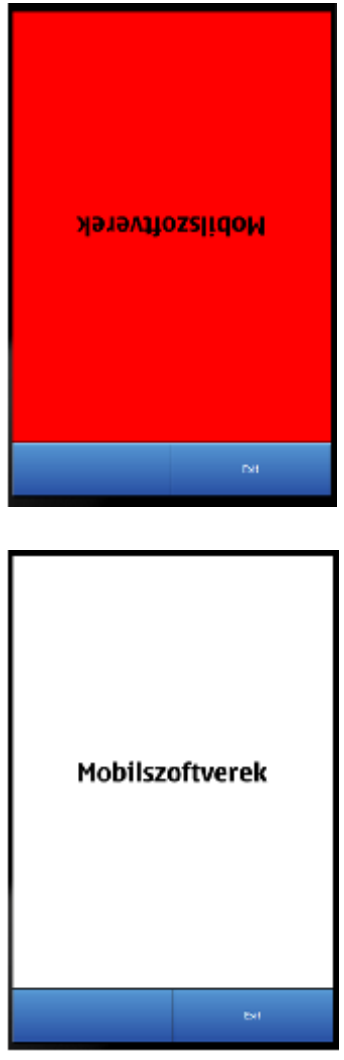
- Minden objektum rendelkezik state property-vel (lista)
- Állapot létrehozása: State típus
  - Egyedi név, name
  - Megadható mikor aktív: when
  - Tartalmazza az összes szükséges változást

```
Rectangle {
    ...
    states: [
        State {
            name: "moved„
            when: mousearea.pressed == true
            PropertyChanges { target: myRect; x: 50; y: 50;}
            PropertyChanges { target: someOtherItem; width: 1000 }
        }
    ]
}
```

States példa

```
import QtQuick 1.0
Rectangle {
        id: mainRect
        width: 200; height: 200
        Text {
          text: "Mobilszoftverek„
          font.pointSize: 10
          font.bold: true
          anchors.centerIn: parent
        }
        MouseArea { id: mousearea; anchors.fill: mainRect }

        states: [ State {
                    when: mousearea.pressed == true
                    PropertyChanges { target: mainRect;
                      color: "red" }
                  },
                  State {
                    when: mousearea.pressed == false
                    PropertyChanges { target: mainRect;
                      color: "white" }
                  }
        ]
}
```



Animáció

- Animációs objektumok hozzárendelés property-khez
  - Az adott property értékének változtatása adott idő alatt
  - Kiindulási és a végső érték
  - Időtartam
  - Tetszőleges property pl. méret, szín, elforgatás

```
PropertyAnimation on rotation {
                          to: 360
                          duration: 1000
                          loops: Animation.Infinite
}
```

Állapotátmenet animálása

- Az állapotokat tartalmazó objektum transitions property-jéhez
- Transition típusú objektum, tartalmazza:
  - Kiindulási és vég állapotot
  - Az animációk típusát és időtartamát

```
transitions:
Transition {
  from: "on"; to: "off"
  ColorAnimation { duration: 500 }
  PropertyAnimation { properties: "rotation"; duration: 1000 }
}
```

Ellenőrző kérdések, tippek

- QT Quick alapok
  - Mi ez, mire lehet használni?
  - QML objektumok, property binding, id property
  - Alap QML elemek (Rectangle, Image, Text, MouseArea)
- Hogyan tudjuk megjeleníteni a QML dokumentumot C++ kódból?
- Mik az állapotok (State)?
  - Hogyan lehet állapotátmenetet animálni? (Transition)
- Hozzunk létre egy új QML komponenst, amely kirajzolja a következőket…
- Mit rajzol ki a következő QML dokumentum…

Qt Quick tanfolyam a BME-n

- Március 3-4, QBF15
- Ingyenes
- Hivatalos Nokia oktató tartja, angolul
- Regisztrálni kell! (csak ha tényleg eljössz)
  - http://events.forum.nokia.com/invitation/qtquickbudapest

Windows Phone 7

Mobilszoftverek

W1

W1 – Tartalom

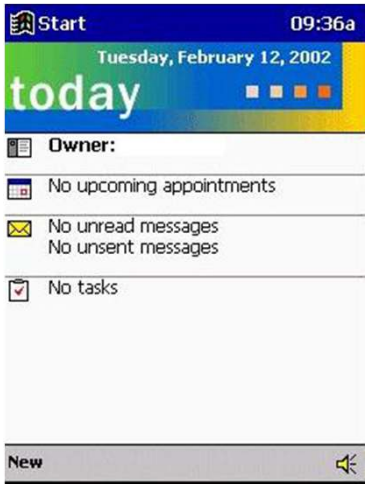
- A platform bemutatása
- Szoftverfejlesztés WP7-re
- XAML
- UI / alkalmazásfejlesztés, alapok, példák

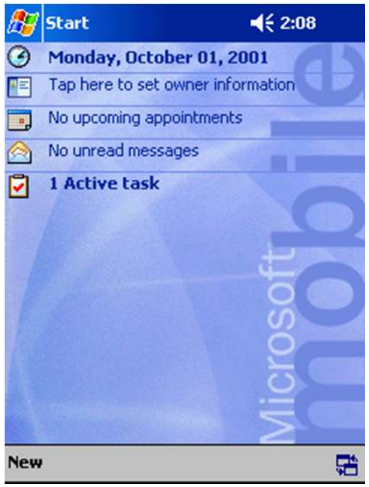
Windows Phone 7

A platform bemutatása

Történelem 1/2

- Első verzió: Pocket PC 2000
- Elsősorban „üzleti felhasználásra”
  - Pen based UI
  - Manager funkciók
- Két készüléktípus:
  - Pocket PC (PDA)
  - Smartphone (kisebb kijelző „egykezes”)
- Pocket PC 2000, 2002, 2003





Történelem 2/2

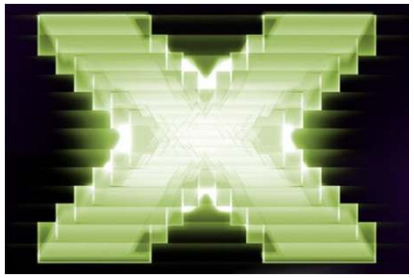
- Windows Mobile 2003, 5, 6, 6.5
- Microsoft termékek mobil verzióival (Outlook, IE, Office, Media Player, stb.)
- Programozás: C++ és .NET Compact Framework
- Multitasking, szabad hozzáférés OS legtöbb funkciójához
- UI elavult (nehéz ujjakkal használni, nincs multi-touch), lassú, nem megbízható, felhasználó hivatalosan nem frissíthet verziót

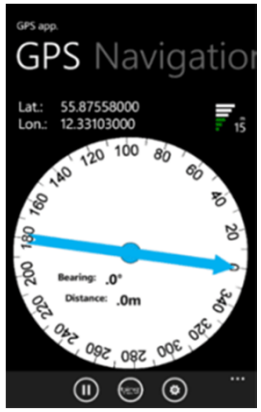
Windows Phone 7

- 2010. október - új név, új világ
- Új UI: Metro
- Új fejlesztőkörnyezet
  - Alkalmazások: Silverlight
  - Játékok: XNA
  - Cloud szolgáltatások (Azure)
- Egységes hardverkövetelmények
- Egyelőre WinMo-hoz képest kevesebb funkció, sok megkötés
  - Nincs multitasking
  - Nincs copy-paste

Hardver követelmények 1/2

- Egységes hardver követelmények minden Windows Phone 7 eszközhöz
  - Kijelző felbontás: WVGA (800x480)
  - Kapacitív, 4-pontos multi-touch kijelző
  - Hardveres DirectX 9 támogatás (3D grafika)
  - GPS és szenzorok (gyorsulásérzékelő, iránytű, közelségérzékelő)





Hardver követelmények 2/2

- Egységes hardver követelmények minden Windows Phone 7 eszközhöz
  - Fényképezőgép/kamera és vaku/fényforrás
  - 3G és Wi-Fi internet eléréshez
  - Start, Search és Back gombok a készüléken
  - Minimum 256 MB RAM, minimum 8 GB flash háttértár

Megjelent WP7 telefonok

- Samsung Focus
- LG Quantum
- HTC Surround
- HTC HD7
- Dell Venue Pro

Nokia Windows Phone concepts

- 2011 október?

Metro UI

- Szöveg (tipográfia) alapú
- Tartalomközpontú
  - Pl. nincsenek felesleges díszítések, peremek
- Touch friendly
- Gyors, animált
- Platformok:
  - WP 7
  - Windows Media Center
  - Zune
- Fekete háttér: energiatakarékos

Ellenőrző kérdések, tippek

- Mi a Windows Phone platform és mi köze a Windows Mobile ill. Pocket PC-hez?
- Mi a WP filozófiája a hardverkövetelményekkel kapcsolatban?
  - Milyen követelmények vannak? Soroljon fel legalább X darabot.
- Mi az a Metro UI (Metro Design Language)? Milyen eszközökön fordul elő?

Windows Phone 7

Szoftverfejlesztés WP7-re

.NET egy dián (?)

- .NET fordító IL (Intermediate Language) kódot generál, ezt a runtime (CLR) futási időben (Just In Time) fordítja gépi kóddá
- Több programozási nyelv lehetséges (20+)
  - Silverlight-hoz C#, Visual Basic (tech preview)
  - XNA-hoz C#
- Managed code: rendszer védelme, garbage collection, reflection, stb…

Silverlight

- Eredetileg böngészőben futó alkalmazásokhoz (mint Flash)
- WPF (Windows Presentation Foundation): .NET 3.0 óta a Windows desktop alkalmazások készítéséhez, komplex UI keretrendszer
- Silverlight: böngészőben futó (RIA) és mobil alkalmazásokhoz
  - WPF-ből nötte ki magát
  - Több platformon/böngészőben kell futnia
  - .NET Framework CLR (Common Language Runtime) helyett CoreCLR
  - Sok apró különbség minden téren WPF-hez képest
  - Néhány funkció csak Silverlight-ban: pl. hivatalosan WPF-hez nincs Deep Zoom

WP7 - Miért Silverlight?

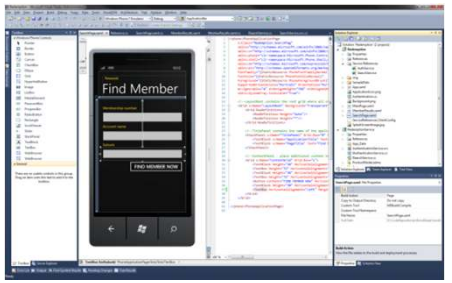
- Eleve multiplatform, erőforrásban szegényebb környezethez tervezték
- Biztonság, „Sandboxed” környezet
  - Csak a Silverlight API-k által nyújtott funkciókhoz férnek hozzá a programok
  - Isolated storage: minden alkalmazásnak saját tárhely (mappa)
- UI: XAML, deklaratív, animáció, vektor alapú
  - Jövőben könnyű lehet áttérni más felbontásokra

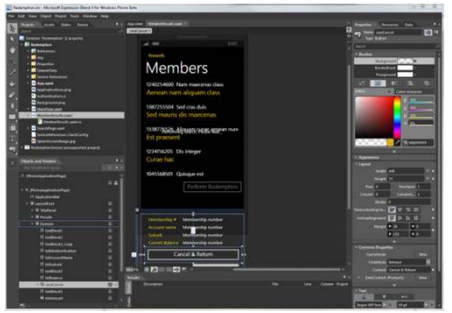
XNA

- XNA is Not an Acronym
- Gazdag programozói keretrendszer és eszközcsomag elsősorban játékfejlesztéshez
- Render loop modell, 2D és 3D támogatás (shaderek is)
- Támogatott platformok:
  - Xbox 360
  - Windows (XP, Vista, 7)
  - Zune
  - **Windows Phone 7**
- XNA Framework
  - .NET CF 2.0-ra épül

Fejlesztőeszközök

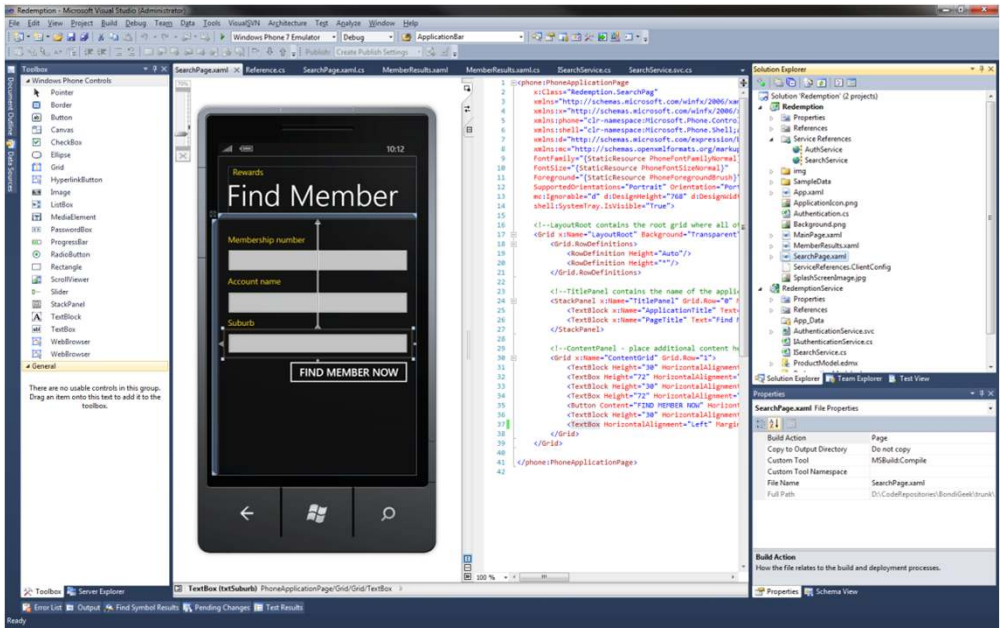
- Visual Studio
- Expression Blend
- Windows Phone Emulator





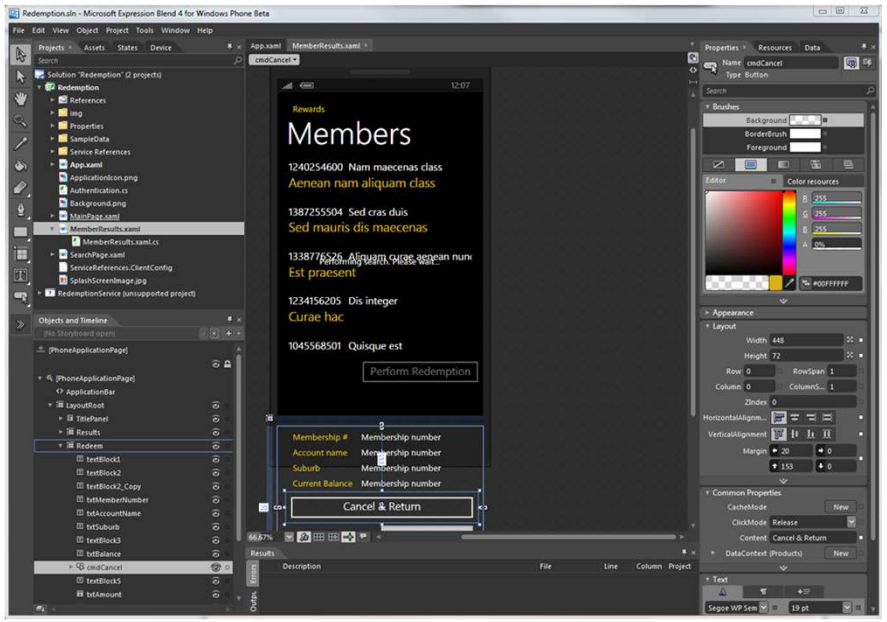
Visual Studio

- Windows Phone fejlesztéshez
  - Bármelyik fizetős verzió + addonok letöltése
  - Ingyenes: Visual Studio 2010 Express for Windows Phone
  - XNA-hoz: XNA Game Studio 4
- Megszokott, gazdag funkcionalitás
- Alap UI (XAML) editor
- Debug funkciók
  - Emulator
  - On device



Expression Blend

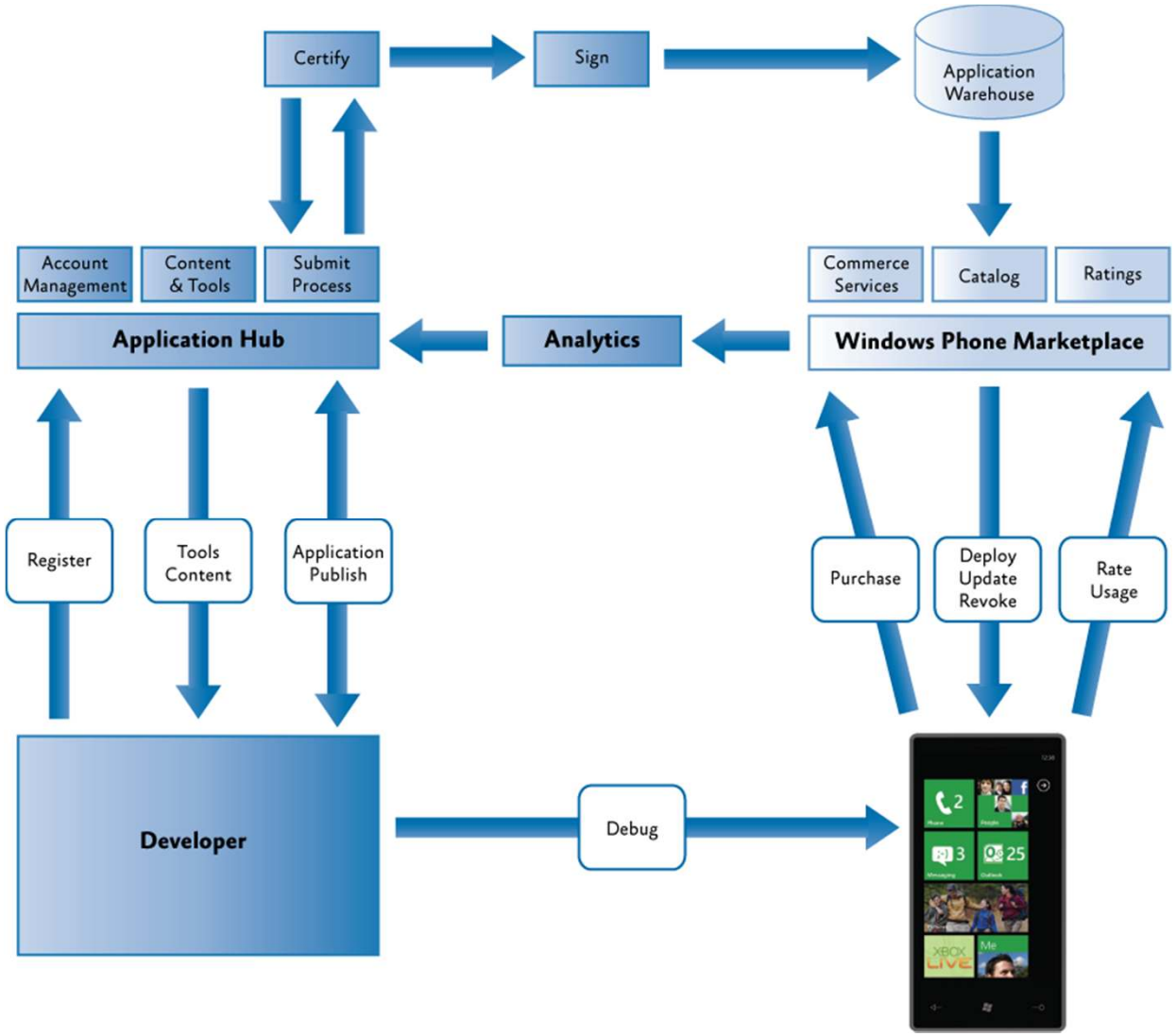
- UI készítéshez
- Nem igényel programozói tudást
- Többet tud mint Visual Studio UI editora
  - Animáció, stílusok, stb.
- XAML-be tud menteni



Windows Phone Emulator

- A mobilon futó operációs rendszer x86 portja
- Kihasználja a PC hardverét
  - 3D gyorsítás
  - Virtualizáció
- Hálózat támogatása
- Mobile Internet Explorer
- Egyelőre hiányozik sok mobilspecifikus funkció emulációja (pl. GPS)

Alkalmazásfejlesztés és -publikálás



App Hub Account

- Mihez kell:
  - Telefonon való teszteléshez / debughoz kell
  - Marketplace-be való alkalmazás feltöltéshez
- 99$ / év
- Regisztrálás App Hub-on
  - http://create.msdn.com/

Windows Phone Marketplace

- Fejlesztőnként 5 ingyenes, végtelen fizetős alkalmazást tölthető fel
  - Minden további ingyenes alkalmazás 20$
- Alkalmazás árának 70%-a a fejlesztőé
  - Kiszabható ár: 0.99$ - 499.99$
- Alkalmazást feltöltés után, megjelenés előtt tesztelik: certification
- 2011. február: 8000 alkalmazás
- Mi az a Magyarország? (nálunk egyelőre nincs)
- Diákoknak ingyenes Marketplace: DreamSpark - www.dreamspark.com

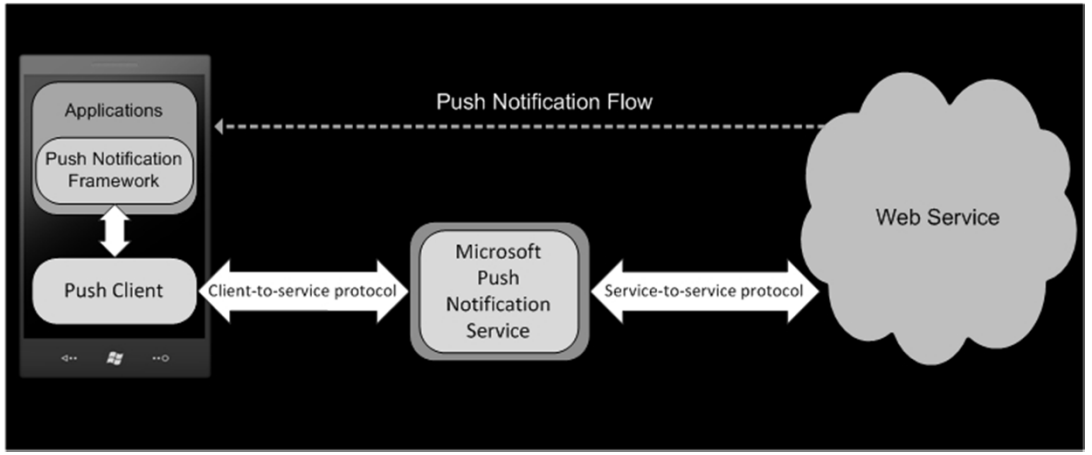
Azure

Windows Azure

- Cloud-computing és tárhely platform
  - Windows Azure Compute Service (pl. web service futtatás)
  - Windows Azure Storage Service (adat és file tárolás)
  - SQL Azure (cloud SQL szerver)
  - Windows Azure Marketplace DataMarket (adatforrások publikálása/lekérése)
- Smartphone: kevés tárhely és korlátos erőforrások

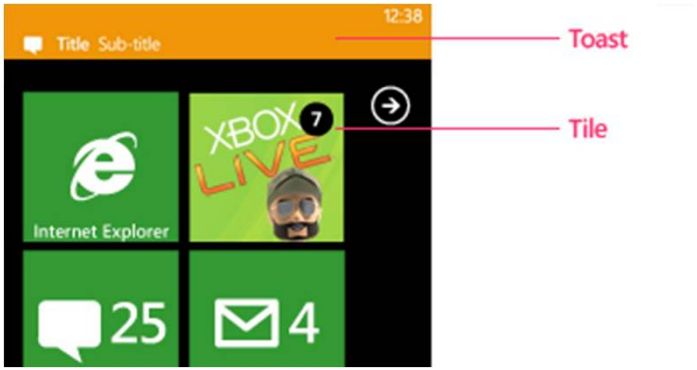
Push notification 1/2

- „Push” értesítések küldhetők alkalmazások számára
  - Poll-ing helyett, alkalmazás aszinkron, azonnali értesítésére
  - Microsoft Push Notification Service-en keresztül
    - Web service hívás indítja (HTTP POST)
- Kis adatforgalom
- Alacsony akkuszint esetén kikapcsol



Push notification 2/2

- Push notification jelzés típusok
  - Toast
    - 10 másodpercig látható
  - Title, átírható:
    - Háttér
    - Számláló
    - Alkalmazás név
  - Raw
    - Nincs külön jelzés
    - Csak ha fut az alkalmazás, egyébként elvész az értesítés



A jövő

- Windows Phone 7.5 (Mango)
  - 2011 szeptemberre várható
  - IE Mobile 9
    - HTML5
    - Silverlight támogatás böngészőben is
  - Multitasking
  - In-app-downloads
  - Sok új API, testreszabhatóság
  - Távolkeleti nyelvek támogatása
- Windows Phone 8 (Apollo)
  - 2012

Ellenőrző kérdések, tippek

- Mi az XNA és a Silverlight? Melyik mire jó? Mi köztük a különbség?
  - Ha egy ilyen vagy olyan alkalmazást szeretnék fejleszteni, melyiket használjam?
- Milyen fejlesztő eszközök léteznek WP7-hez?
- Hol és hogyan tudunk WP7 alkalmazásokat terjeszteni? (Marketplace)
- Mi az az Azure? Mi köze van a WP világhoz?
- Mi az a push notification és milyen megjelenítési módokat biztosít hozzá a platform?

Windows Phone 7

XAML alapok

XAML alapok

- XAML elemek és attribútumok
  - XAML

```
<Button
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  Content="OK" Click="button_Click"/>
```

  - C#

```
using System.Windows.Controls;
using System.Windows;
     Button b = new Button();
      b.Click += new RoutedEventHandler(button_Click);
      b.Content = "OK";
```

XAML

- XAML: XML alapú, deklartív objektum példányosító nyelv
  - Nincs mágia, nincs kódból el nem érhető funkció
- Silverlight esetén a programozók és designerek, fejlesztőeszközök és designer eszközök közös nyelve
- A XAML „csak” egy eszköz, nyelv a felhasználói felület deklaratív leírására
- Kicsit hasonlít az XHTML-re, de sokkal kifejezőbb, erőteljesebb
- Elemek hierarchiája, layout, grafika, média, animáció, adatkötés, triggerek, stílusok, … mind leírhatók benne!

A XAML feldolgozása

- XAML plain text/XML formában tárolva
  - WPF-el ellentétben nincs bináris formára (BAML) konvertálás
- SilverLight alkalmazás futtatásakor
  - XAML beolvasása
  - Objektummodel felépítése

XAML alapok 1.

- A tag-ek osztályokat példányosítanak
- Az attribútumok tulajdonságokat állítanak be
- Névterek, szerelvények
  - Az alapértelmezett névtérben Silverlight vezérlők
  - Az ‘x’ névtér az általános XAML motor
    - például az x:Name – a generált változó neve
  - Egyéb szerelvények és névtereik importálhatók
    - Bármelyik szerelvényhez attribútummal névtér rendelhető

XAML alapok 2.

- Egy tag tartalma (gyerek elemek) az „alapértelmezett” tulajdonság értékét határozza meg
- Az attribútum értelmezéséhez lehet írni konvertert
  - `<TextBlock Width= "12 cm" />`
- Az attribútum értéke lehet összetett objektum is
  - Pl: LinearGradientBrush

```
<LinearGradientBrush>
  <LinearGradientBrush.GradientStops>
    <GradientStop Offset="0" Color="Red"/>
    <GradientStop Offset="1" Color="Green"/>
  </LinearGradientBrush.GradientStops>
</LinearGradientBrush>
```

- Eseménykezelők
  - Pl. `Click="button_Click"`

XAML példa

- Property Elemek
  - XAML:

```
<Button>
    <Button.Content>
        <Rectangle Height="40" Width="40” Fill="Black"/>
    </Button.Content>
</Button>
```

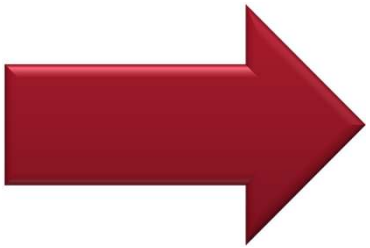
  - C#:

```
System.Windows.Controls.Button b = new System.Windows.Controls.Button();
System.Windows.Shapes.Rectangle r = new System.Windows.Shapes.Rectangle();
r.Width = 40;
r.Height = 40;
r.Fill = System.Windows.Media.Brushes.Black;
b.Content = r;
```

XAML alapok – Content property

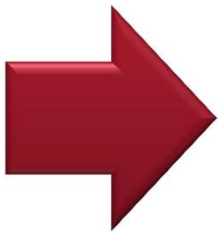
- Content: egy kiemelt property

```
<Button>
        <Button.Content>
           OK
        </Button.Content>
</Button>
```



```
<Button>
        OK
</Button>
```

```
<Button>
   <Button.Content>
      <Rectangle Height="40"
         Width="40" Fill="Black"/>
    </Button.Content>
</Button>
```



```
<Button>
    <Rectangle Weight="40“
       Width="40" Fill="Black"/>
</Button>
```

Code-Behind

- XAML-ben definiált osztályokhoz plusz kód
  - Egy külön forrásfájlban: Code-Behind file (pl. App.xaml és hozzá App.xaml.cs)
  - XAML-en belül <x:Code> BAD PRACTICE
- Code-Behind file
  - Egy partial class-ra adhatunk hivatkozást x:Class propertyvel
  - A partial class-nak XAML-ben definiált gyökérelem osztályából kell származnia
    - Pl. PhoneApplicationPage

XAML előnyök / hátrányok

- ✓ Jobban olvasható
  - Fejlesztőnek és eszközöknek egyaránt
- ✓ Követi a felhasználói felület hierarchiáját
- ✓ Tömör
- ✓ Dinamikus példányosítás (bár lassabb)

- ☝ Nehézkesebb a hibakeresés (debuggolás)
- ☝ Bár ritkábban, de itt is történhetnek „mágikus” dolgok mint kódban
  - ? Vajon ez most itt miért fehér?
  - ? Miért nem működik az adatkötés?

Ellenőrző kérdések, tippek

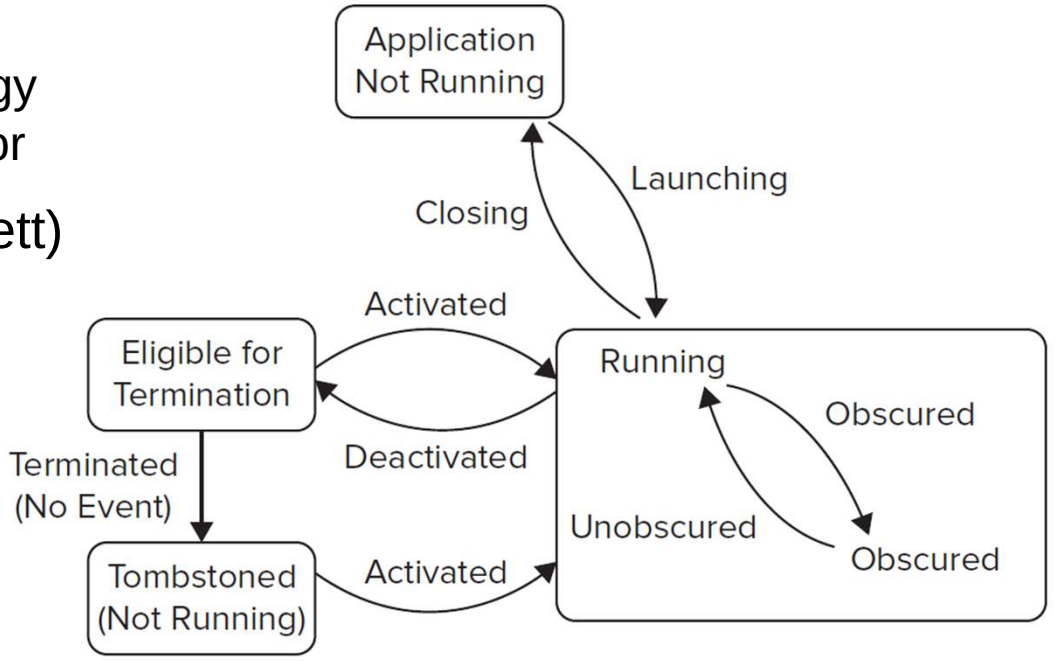
- Mire jó a XAML? Mik az előnyei, hátrányai?
- Hogyan lehet a XAML-ben definiált osztályhoz programkódot (C#) írni?
- Milyen kód generálódik egy adott XAML részletből, vagy milyen XAML tartozik egy adott C# kódhoz (konkrét feladat, egyszerű kóddal)
- Mire szolgál az x névtérből az x:Name és az x:Class elem?
- Hogyan dolgozza fel a Silverlight a XAML fájlt?

Windows Phone 7

Silverlight alkalmazások programozása WP7-re

Alkalmazás életciklus

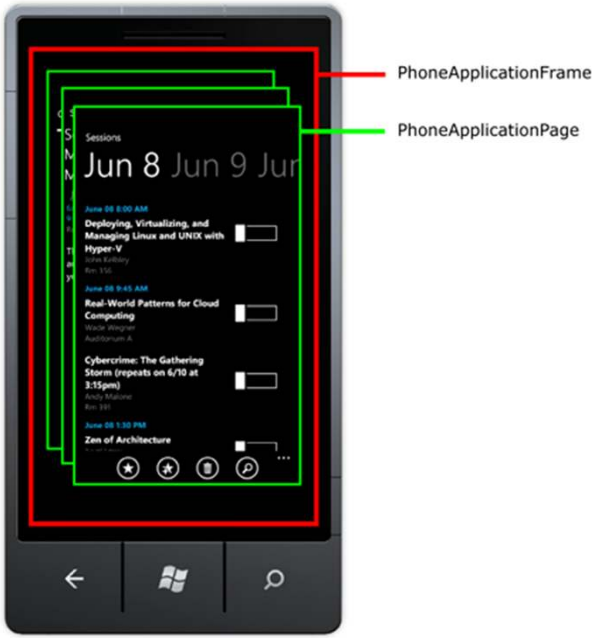
- Egyszerre egy aktív, előtérben lévő alkalmazás
  - Háttérbe küldéskor benn maradhat a memóriában, de erre nem lehet alapozni
- Négy állapot
  - Running
    - Obscured: kitakarja egy nézet pl. telefonáláskor
  - Not Running (user kilépett)
  - Eligible for Termination (háttérbe került, Start gomb hatására)
  - Tombstoned (rendszer kilőtte)



Navigáció - Alap UI osztályok

- PhoneApplicationFrame
  - A gyökér UI elem
  - Mindig egy PhoneApplicationPage-et mutat
  - Lapváltás: „navigáció”

```
this.NavigationService.Navigate(
  new Uri(”/SecondPage.xaml”,
  UriKind.Relative));
```



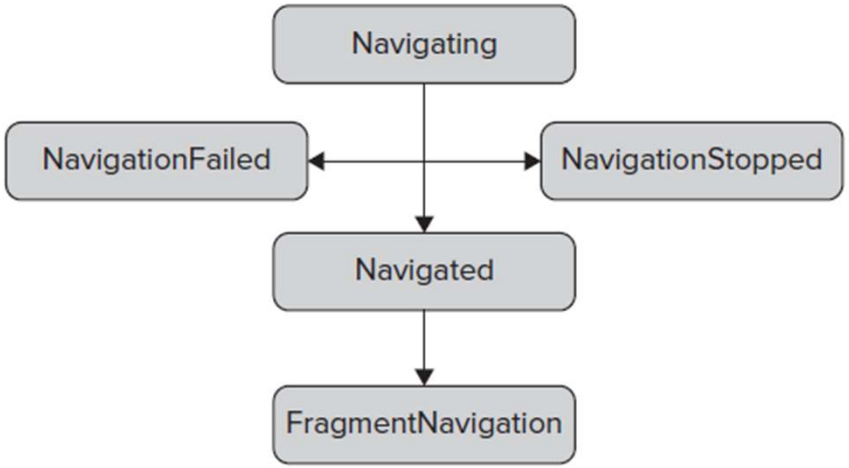
  - Lapváltáskor paraméterek is átadhatok

```
this.NavigationService.Navigate(
    new Uri("/SecondPage.xaml?msg=" + MyTB.Text,
    UriKind.RelativeOrAbsolute));
```

- PhoneApplicationPage
  - Az alkalmazás egy különálló „képernyője”, nézete

Navigációs események

- NavigationService események



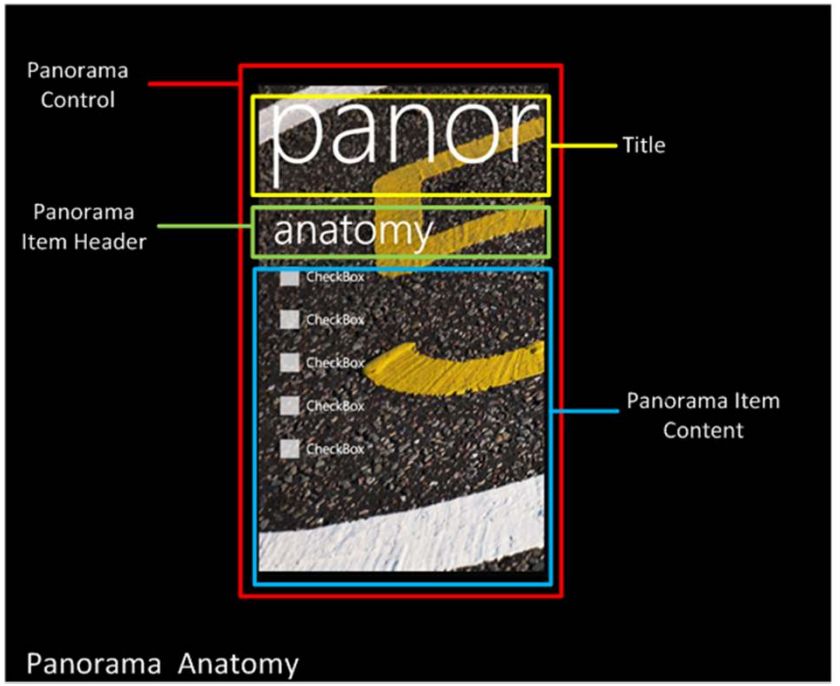
- PhoneApplicationPage virtuális metódusok
  - OnNavigatingFrom (lap elhagyása előtt)
  - OnNavigatedFrom (lap elhagyása után)
  - OnNavigatedTo (lapra érkezés)
  - OnFragmentNavigation (lapon belül navigálás URI fragmens alapján)

Lap elrendezések

- Panoramic
- Pivot
- List
- Full-screen
- …

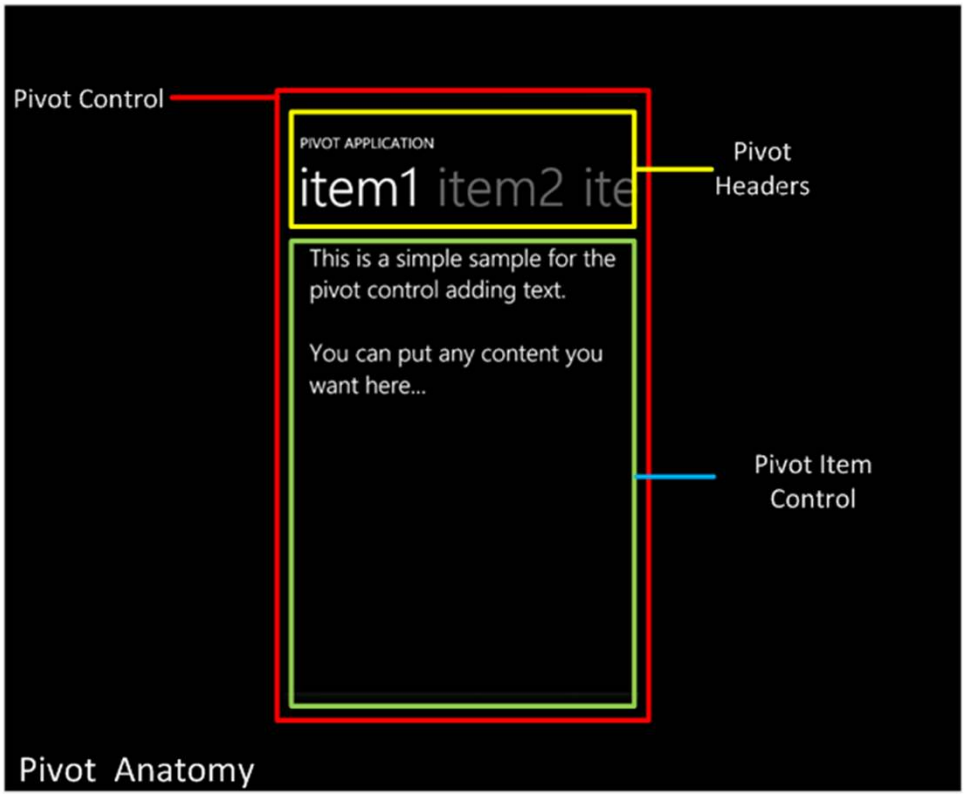
Panoramic

- Horizontális scroll
- Gyakran „hub”-okhoz



Pivot

- Tab-okra osztott



Controls (példák)

- Button, CheckBox, RadioButton, TextBox, PasswordBox

- TextBlock, Image, Slider, ProgressBar

TextBlock XAML

```
<TextBlock Height="152" HorizontalAlignment="Left"
Margin="12,250,0,0" Name="textBlock1"
VerticalAlignment="Top" Width="438" TextWrapping="Wrap">
        Lorem ipsum dolor sit amet, consectetur adipiscing
        elit.
        <doc:Run FontStyle="Italic" > Lorem ipsum dolor
        sitamet, consectetur adipiscing elit.
        </doc:Run >
        <doc:LineBreak />
        <doc:Run FontWeight="Bold" >Lorem ipsum dolor sit
        amet, consectetur adipiscing elit.
        </doc:Run >
</TextBlock>
```

Gesztúrák

- Single-touch
  - Tap: klikk
  - Double-Tap: duplaklikk
  - Pan: scrollozás, mozgatás
  - Flick: kinetikus scroll (felengedés után tovább mozog)
  - Touch and Hold: „hosszú” klikk
- Multi-touch
  - Pinch and Stretch: két érintési pont közelítése / távolítása, tipikusan nagyítás/kicsinyítés („pinch to zoom”)

Silverlight gesztúra támogatás

- Egyelőre csak minimális beépített gesztúra támogatás
- Több Control alapból támogatja a Tap gesztúrát

```
<Button Content="Simple Gestures" Name="MyButton"
  Click="MyButton_Click" />

private void MyButton_Click(object sender, RoutedEventArgs e) {
  MessageBox.Show("Ne nyomkodd a gombot!");
}
```

- Ha nincs Tap, a UIElement-ből származó osztályoknál van 2 event
  - MouseLeftButtonDown
  - MouseLeftButtonUp
- Egyéb gesztúrákhoz saját eseménykezelőt kell írni
  - Pl. Double-Tap: mérjük az időt két Tap között, ha kevesebb mint 1s telt el, akkor ez egy Double-Tap

Multi-touch

- Emulátor támogatja
  - Érintőképernyős gépen fejlesztés
  - 2 USB egérrel: Multi-Touch Vista projekt http://multitouchvista.codeplex.com
- Multi-touch események alkalmazás szinten
  - Touch.FrameReported event minden képernyőérintéskor / érintési pont elmozduláskor
  - Az alkalmazás összes érintési pontja nyilván van tartva
- Multi-touch események UIElement szinten
  - 3 esemény
    - ManipulationStarted (első érintés)
    - ManipulationDelta (elmozdulás, új ponton érintés, elengedés)
    - ManipulationCompleted (elengedés minden ponton)
  - Lekérhetők az érintési pontok

Silverlight Toolkit

- Microsoft által kiadott új API-k, komponensek (később bekerülhetnek teljes Silverlight-ba)
- Silverlight for Windows Phone Toolkit
  - Pl. DatePicker, ToggleSwitch
  - GestureHelper (Flick, Pinch…)


```
<Image Source="dividbyzero.jpg">
  <toolkit:GestureService.GestureListener>
    <toolkit:GestureListener
      PinchDelta="OnPinchDelta" />
  </toolkit:GestureService.GestureListener>
</Image>

private void OnPinchDelta(object sender,
                          PinchGestureEventArgs e)
{
  ImageScaling.ScaleX = e.DistanceRatio;
  ImageScaling.ScaleY = e.DistanceRatio;
}
```

  - TransitionService: animált lapváltás

Ellenőrző kérdések, tippek

- Ismertesse a WP7 alkalmazás életciklusát, állapotait és az állapotátmeneteket!
- Mik azok a lapok? Mi a navigáció? Hogy épül fel egy Silverlight mobilalkalmazás felhasználói felülete?
- Milyen érintőképernyőn használatos gesztúrákat ismer? Ezeket hogyan lehet megvalósítani WP7-en?
- Mi az az Silverlight Toolkit? Milyen funkciókat nyújt?

Mobilszoftverek

J1 - Általános mobil alkalmazásfejlesztési elvek

Java ME

Ekler Péter

peter.ekler@aut.bme.hu

A Java ME előadások tartalma

- Alapelvek:
  - Átlátható felhasználói felület
  - Memóriatakarékos programozás
- Java ME alkalmazások életciklusa
- Java ME UI
- Szálkezelés és időzítők használata
- Perzisztens adattárolás és hálózatkezelés
- Mobiltelefon szolgáltatások
- Bluetooth kommunikáció
- Multimédia funkciók
- 3D grafika alapok

Tartalom

- Alapelvek
- Java ME helye a mobil platformok világában
- Java ME alkalmazások életciklusa
- Mobil alkalmazás felhasználói felülete
- Szálkezelés – miért fontos

Alapelvek

- Általános elvek mobil szoftverek készítése esetén:
  - Kis kijelző
  - Eltérő (!) felbontások
  - Beviteli eszközök korlátossága/eltérősége
- Mobil szoftvertervezési módszerek
- Az alkalmazások felépítése a legtöbb platformon hasonló
- Mielőbb próbáljuk ki valós készüléken!

Mobilos fejlesztés

- Fejlesztés PC-n
- Tesztelés Emulátoron
  - Mobiltelefon viselkedését emulálja
  - Figyelembe veszi-e a készülék erőforrás korlátait?
- Telepítés telefonra
- Tesztelés telefonon
- **Vagy**: On-Device debug

Mobil alkalmazás telepítése

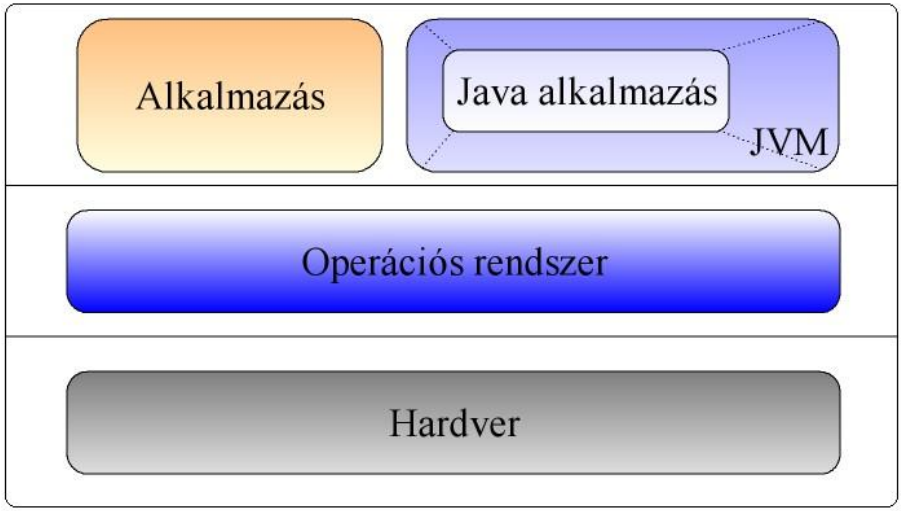
- Telepítés a készülékre
  - USB, Bluetooth
  - Böngészőn keresztül
  - Gyártó szoftvere, pl.: Nokia OVI Suite
- Fontos beállítások:
  - Alkalmazás neve
  - Ikon (ügyeljünk az ikonméretre)
  - Engedélyek

Miért Java ME?

- Platformfüggetlen
- Gyakorlatilag az összes mobil platform képességének a metszete
- Általános mobil alkalmazás fejlesztésére ideális
- A tervezési elvek ugyanúgy érvényesek
- Könnyebb az alapokat elsajátítani
- Ismert a nyelv, azonban néhány Java SE-ben megszokott osztály hiányozhat/eltérően viselkedhet

Platformfüggetlenség

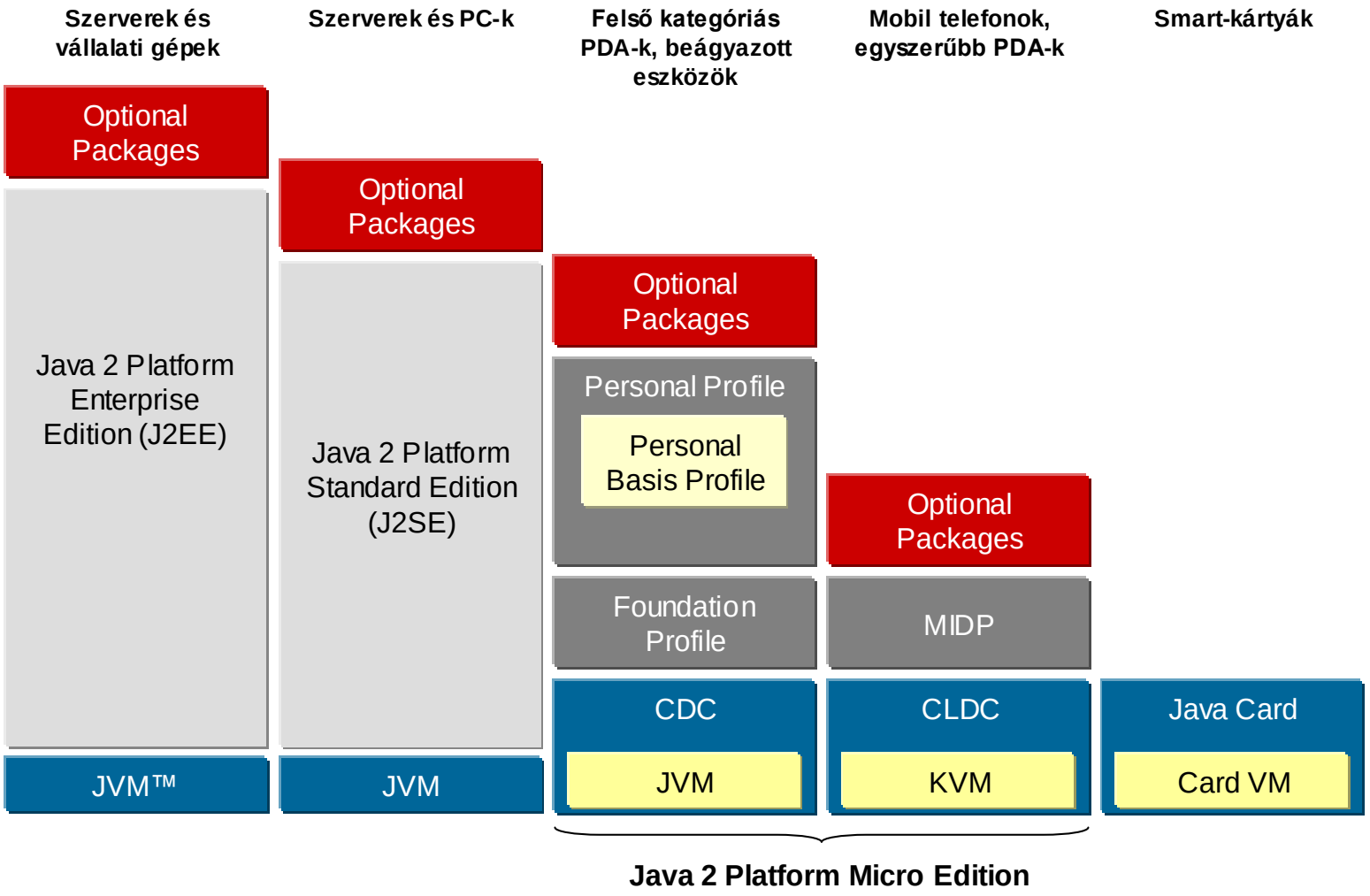
- Java ME szorosan kapcsolódik az operációs rendszerekhez
- Java alapú alkalmazások alapelve:
  - Az operációs rendszerre épül a Java virtuális gép
  - Az alkalmazások ezen virtuális gépen keresztül érik el az operációs rendszer szolgáltatásait
- A fentiekből következik a platformfüggetlenség



Java platformok

- J2EE: a legbővebb platform mely leginkább vállalati méretű alkalmazások fejlesztésére használatos
- J2SE: általános, desktop alkalmazások, valamint appletek fejlesztésére használják
- Java ME:
  - Legszűkebb API-val rendelkező platform
  - Lecsökkentette az API által biztosított funkciókat, hogy a mobil készülékek legszélesebb skáláján alkalmazható legyen

Java technológia főbb elemei



Java ME

- A technológia elemei:
  - Konfigurációk (CDC, CLDC)
  - Profilokat (MIDP)
  - Opcionális csomagok (JSRs)
- Ezen elemekből épül fel a teljes Java futtatókörnyezet (JRE, Java Runtime Edition) a mobil készülékeken

Konfigurációk

- Konfiguráció:
  - Virtuális gép
  - Meghatározott könyvtárhalmaz
- Az azonos konfigurációk biztosítják az egy kategóriába tatozó eszközök számára az alapfunkcionalitásokat: memória kezelés, hálózat kezelés, stb.
- Jelenleg két konfiguráció létezik:
  - CDC (Connected Device Configuration)
  - CLDC (Connected Limited Device Configuration)

Profilok

- A CLDC-vel együtt nevezzük a Java mobil eszközökre való futtatókörnyezetének (JRE)
- A MIDP és a CLDC együtt dinamikus és biztonságos platformot definiál
- Segítségével olyan alkalmazások írhatók, melyek a korszerű mobil készülékekre vannak optimalizálva
- A grafikus felhasználói felületet a mobil készülékekhez igazították

Opcionális csomagok I.

- A CLDC és MIDP fölött helyezkednek el a JSR-ek
- A készülékekben található különféle funkciókat teszik elérhetővé a Java virtuális gépen (bluetooth, filerendszer, stb.)
- Az opcionális csomagokkal kibővítve a Java ME platform jobban ki tudja használni az adott készülék képességeit

Opcionális csomagok II.

- JSR 82: Bluetooth
- JSR 135: Mobile Media
- JSR 172: Web service
- JSR 75: File/PIM
- JSR 226: SVG

Java ME jövője

- SUN-Oracle integráció
- Jelenleg az egyetlen megoldás platform független mobil alkalmazásfejlesztésre
- Korlátozott képességek, nem elégítik ki a modern igényeket
- Megújulás szükséges:
  - Java ME SDK 3.0 Mac OS-re (2010 január)
  - LWUIT (jó irány, de elavult technológia)
  - MIDP 3.0

MIDP 3.0 – JSR 271 1/3

- Fejlettebb MIDlet viselkedés:
  - Konkurens MIDlet futtatás
  - Tűzfal kezelés és fejlettebb életciklus
  - MIDletek háttérben futtatása (faceless)
  - MIDlet futtatása az Idle Screen-en
  - MIDlet auto indítás (akár boot során)
  - Fejlett MIDlet közi kommunikáci (direkt és esemény alapú kommunikáció)

MIDP 3.0 – JSR 271 2/3

- Osztott könyvtárak támogatása
- Készülékek közti interoperabilitás fejlesztése
- MIDlet telepítési módok bővítése: OMA (SyncML) DM/DS, Bluetooth, hordozható média, MMS és JSR 232
- Készülék tulajdonság lekérdezés fejlesztése
- Lokalizáció fejlett támogatása

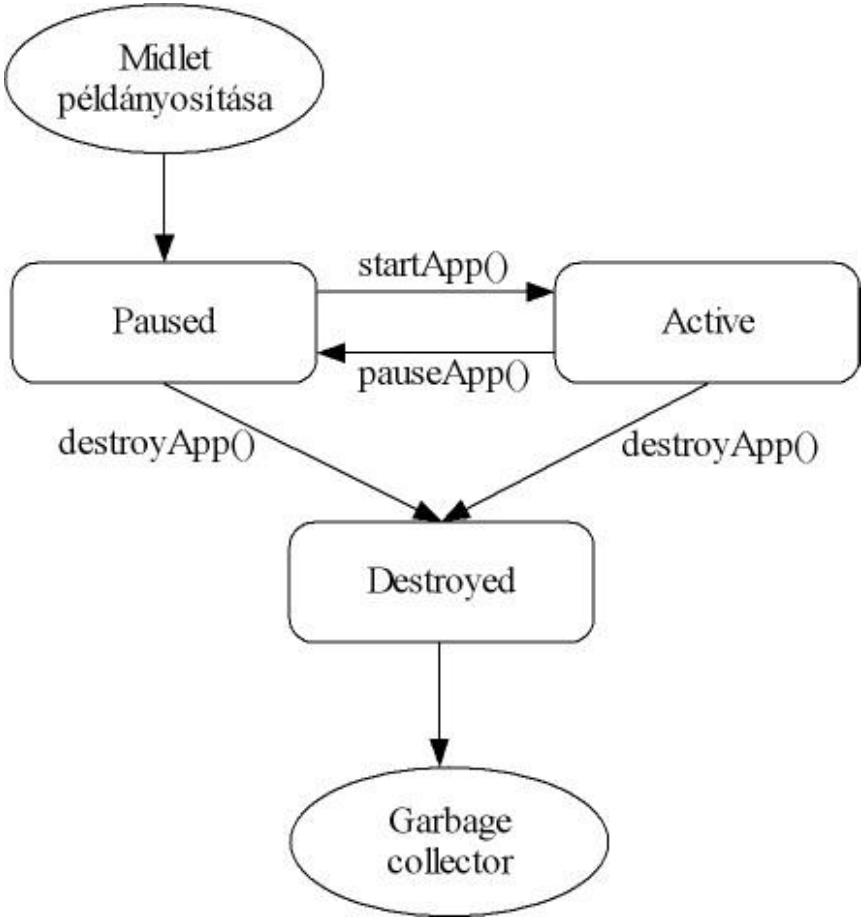
MIDP 3.0 – JSR 271 3/3

- Fejlettebb funkcionalitás minden területen:
  - Átláthatóbb és testre szabhatóbb UI elemek
  - Nagyfelbontású kijelzők támogatása
  - Másodlagos kijelző kezelése
  - Gyorsabb játékok
  - Biztonságos RMS
  - Távolról elérhető RMS
  - IPv6
  - Több hálózati interface támogatása

MIDlet

- A MIDP környezetben futó Java alkalmazásokat MIDleteknek nevezzük
- A MIDleteket egy alkalmazásmanager (AMS – Applicatio Management System) kezeli
- Egy MIDletnek három állapota van: Active, Paused, Destroyed

MIDlet életciklus



MIDlet váz - Példa

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloMidlet extends MIDlet
{
  public void startApp() {
  }
  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
  }
}
```

Felhasználói felület – cél

- Elérhetővé tegye az alkalmazás funkcióit
- Elfedje a felhasználó elől a program összetettségét
- Megkönnyítse az alkalmazás használatát
- Fontos információk átlátható formátumban való megjelenítése
- Barátságosabbá tegye a programot

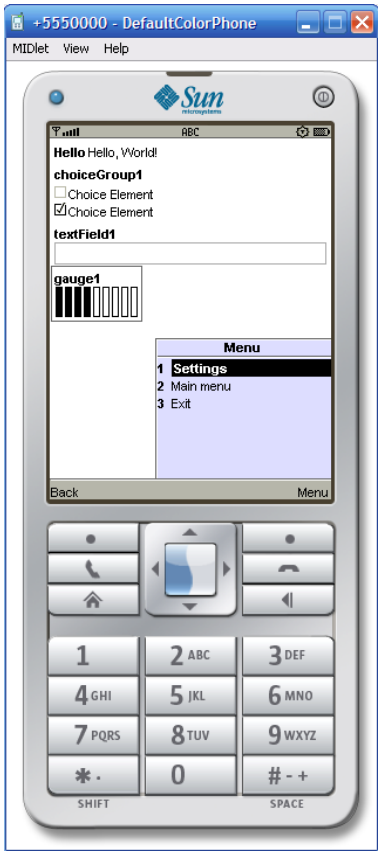
Példa hibás felhasználói felület kialakítására

- Linux ATM

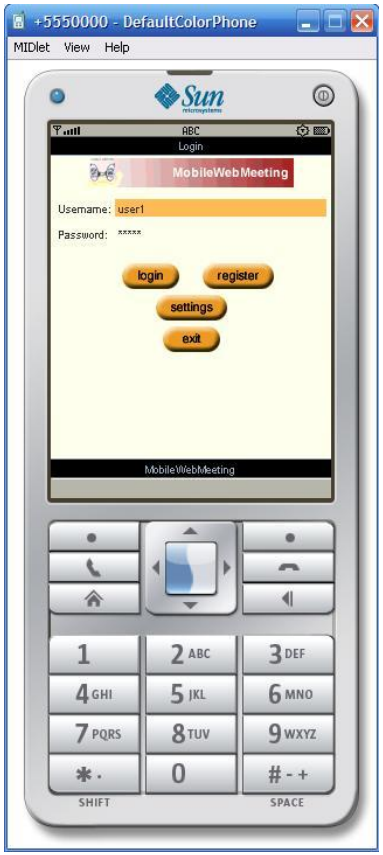
Tisztelt ügyfelünk!

Bankunk hálózatának fejlesztése keretében, valamint a költségek csökkentése érdekében ATM-jeinket Linux operációs rendszerre állítjuk át.
Készpénzfelvételhez használja a getcash parancsot.
Szintaxis:
getcash -ac számlaszám (hexában) -c kártyaszám (hexában) -p pinkód (MD5 hashként) -a összeg (hexában) -b bankazonosító (128 bites egyedi számlavezető bankazonosító)
Ha bizonylatot szeretne kapni a tranzakcióról, akkor irányítsa a **getcash parancs stdout-ját a /dev/lp0-ba, szükség esetén grep-ezze ki** az értékes sorokat.

Tipikus UI



Beépített GUI elemek



„Kézzel rajzolt” UI

Java ME alkalmazás felülete

- Java ME alkalmazás eltérő típusú készülékeken futtatható
- Csak olyan előre elkészített felületelem használható, amely a legtöbb készüléken megjeleníthető
- Az előre elkészített felületelemek skálája szűk

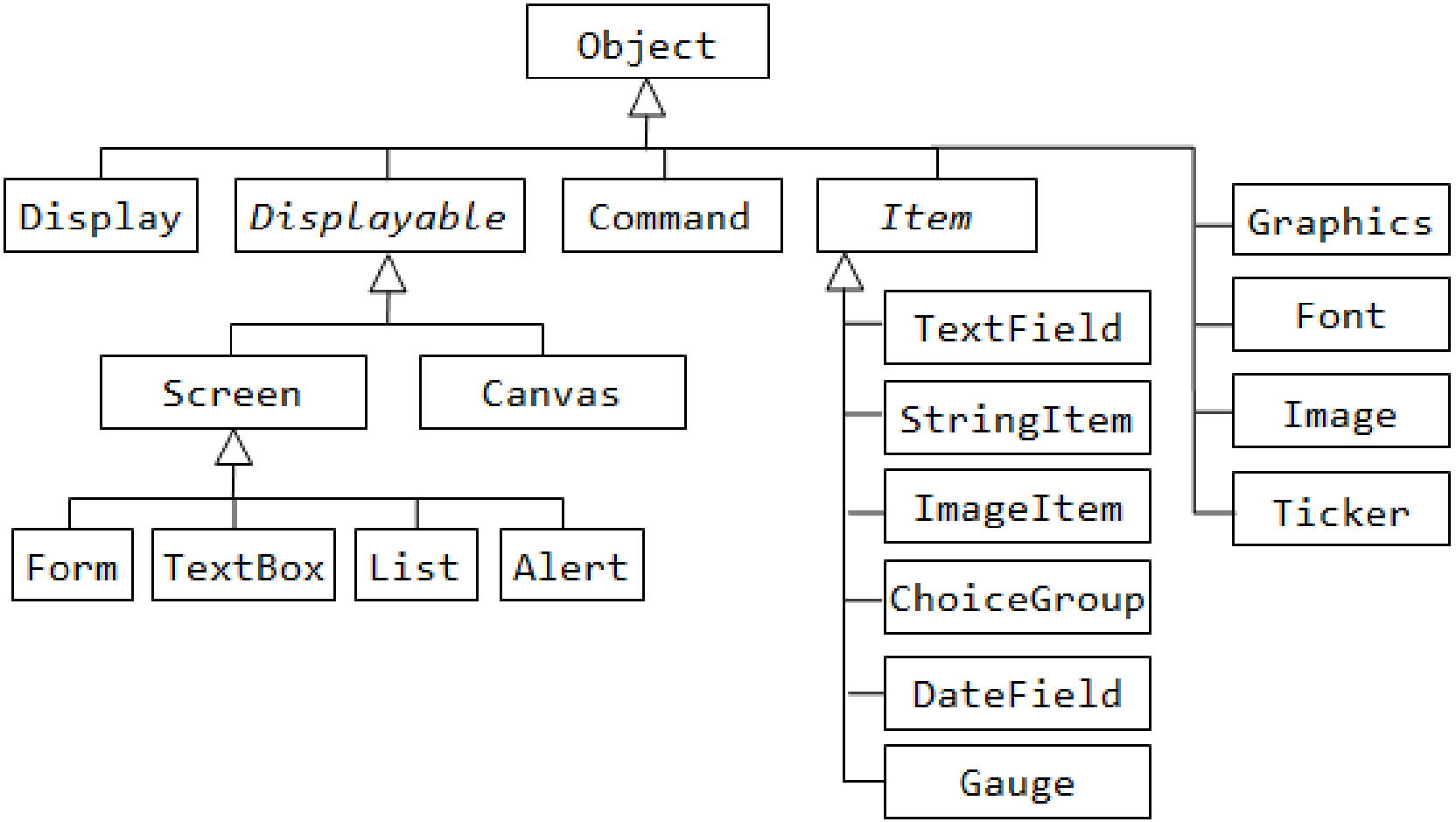
MIDP LCDUI

- A J2SE megszokott grafikai elemei (awt, swing) nem használhatók a mobiltelefonon
  - Túl nagyok
  - Átlapolódó ablakokat használnak
- Ezért hoztak létre egy teljesen új GUI-t: MIDP LCDUI
  - Optimalizált a kisméretű kijelzőkhöz
  - Tipikus felhasználói felületelemek kialakítása: lista, űrlap

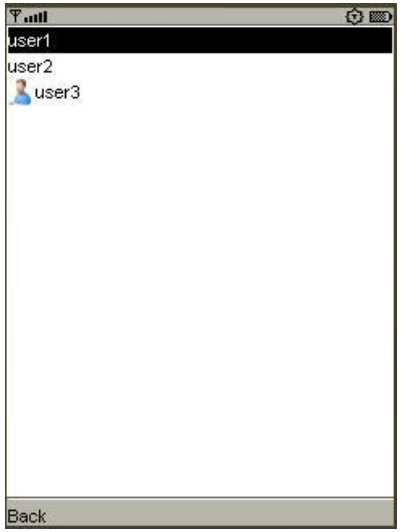
Kijelző tartalma

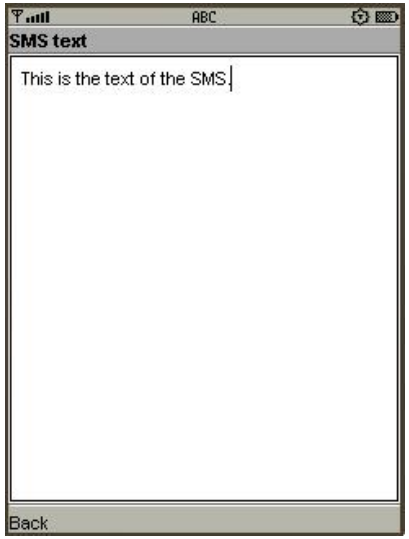
- Java ME platformon a Displayable egy kijelző tartalmat határoz meg, mint konténer típust
- Különféle típusú konténerek különféle elemek megjelenítésre használatosak
- 4+1 féle Displayable-t definiáltak:
  - List
  - TextBox
  - Alert
  - Form
  - *Canvas*
    - Nem a Screen osztályból származik
    - Erre bármit lehet rajzolni

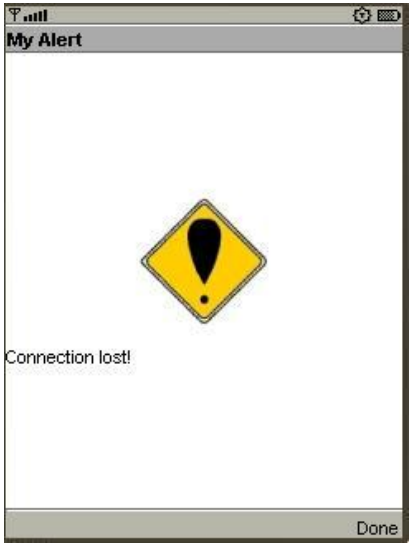
MIDP UI API

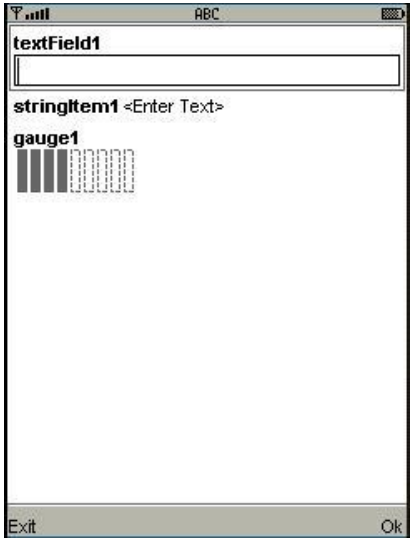

Object
Display
Displayable
Command
Item
Graphics
Font
Image
Ticker
Screen
Canvas
Form
TextBox
List
Alert
TextField
StringItem
ImageItem
ChoiceGroup
DateField
Gauge

Példák


user1
user2
user3
Back

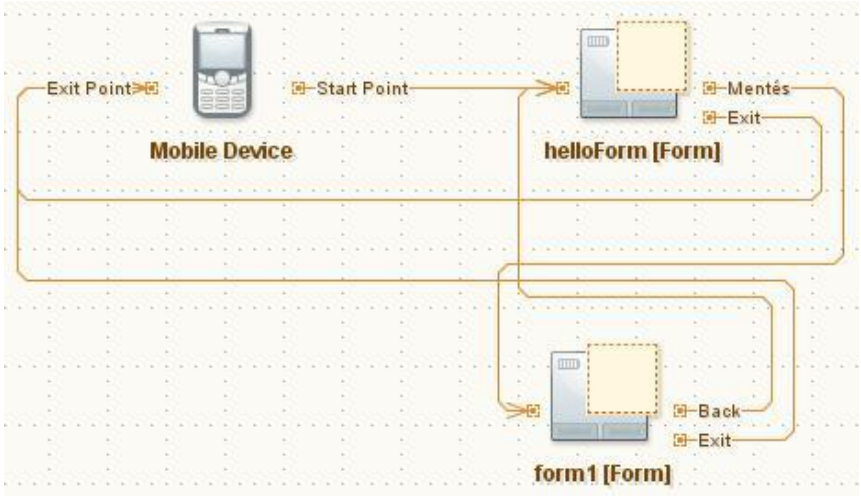

ABC
SMS text
This is the text of the SMS
Back


My Alert
Connection lost!
Done


ABC
textField1
stringItem1 <Enter Text>
gauge1
Exit
Ok

Navigáció screen-ek között

- Az alkalmazás általában több Displayable-ből áll
- Átlátható módon kell megvalósítani a navigálást, váltást a nézetek között
- NetBeans Flow Designer
- **Display.getDisplay([midlet]).setCurrent([displayable]);**



Beépített GUI elemek

- A mobil készülékeken találhatók előre elkészített GUI elemek
- Célszerű ezeket használni, mivel ezek a telefonra vannak optimalizálva (méret, megjelenés tekintetében)
- `Form`-ra helyezhetők
- `Item` ősosztályból származnak

- `StringItem`
- `TextField`
- `ImageItem`
- `Spacer`
- `Gauge`
- `DateField`
- `ChoiceGroup`
- `TableItem`
- `CustomItem`

Felhasználói parancsok beviteli módja

- A telefon parancsgombjaihoz (Softkey) lehet különféle eseményeket (akciókat) rendelni
- Lehetőség van kettőnél több akció definiálására is -> menü
- Ebben az esetben az akció prioritása határozza meg a menüben elhelyezkedő sorrendet
- `new Command("Options", Command.OK, 1);`

Példa - Form-ból származott osztály konstruktora

```
public UserForm1(String title) {
        super("");
        setTitle("Film");

        rblCinema = new ChoiceGroup("rblCinema", ChoiceGroup.EXCLUSIVE);
        rblCinema.append("Corvin", null);
        rblCinema.append("Idjos", null);
        append(rblCinema);

        addCommand(new Command("Options", Command.OK, 1));
        addCommand(new Command("Törlés", Command.OK, 2));

        ChoiceGroup rblDate = new ChoiceGroup("rblDate", ChoiceGroup.EXCLUSIVE);
        rblDate.append("06/06/2005 17:00", null);
        rblDate.append("06/06/2005 19:00", null);
        append(rblDate);

        Gauge tickets = new Gauge("Tickets", true, 200, 50);
        append(tickets);

        ChoiceGroup rblFilmTitle = new ChoiceGroup("rblDate", ChoiceGroup.EXCLUSIVE);
        rblFilmTitle.append("Film Title1", null);
        rblFilmTitle.append("Film Title2", null);
        append(rblFilmTitle);
    }
```

Az eredmény


Film
rblCinema
Corvin
Idjos
rblDate
06/06/2005 17:00
06/06/2005 19:00
Tickets
rblDate
Film Title1
Film Title2
Menu
SELECT
CLEAR
SHIFT
SPACE


Film
rblCinema
Corvin
Idjos
rblDate
06/06/2005 17:00
06/06/2005 19:00
Tickets
rblDate
Film Title1
Film Title2
Menu
1 Next
2 Exit
3 Options
4 Törlés
Menu
SELECT
CLEAR
SHIFT
SPACE

Canvas I.

- A nem űrlap alapú alkalmazások nagy része ezt használja
  - Bármit, bárhova lehet rajzolni
  - A képernyő pontosan úgy jelenik meg, ahogy azt lekódoltuk
- Játékok nagy része Canvas-ra épül

Canvas II.

- Saját Canvas osztály létrehozása származtatással
- Rajzolás, billentyűk kezelése

```
import javax.microedition.lcdui.*;

  public class MyCanvas extends Canvas {
    public void keyPressed( int keycode ) {
      // Billentyűnyomás eseménykezelő.
       ...
    }

    public void paint( Graphics g ) {
  // A Canvas újrafestésekor hívódik meg. A
  Canvas-ban kötelező implementálni.
    }
  }
```

Szálkezelés

- Soha ne blokkoljuk a UI szálat
- Elfogadhatatlan, ha „lefagy” egy mobil alkalmazás
- Java ME – Thread osztály használata (később példa!)

```
class MyTask extends Thread{
  public void run(){…}
}
… // használat:
new MyTask().start();
```

Időzítők használata

- *TimerTask*: ütemezhető feladat
- *Timer*: ütemező
- Másodpercenként egy „.” hozzáfűzése a kijelzőhöz (*mainForm* létezik):

```
private class MyTimerTask extends TimerTask
{
    public void run() {
        mainForm.append(".");
    }
}
...
Timer mainTimer = new Timer();
mainTimer.schedule(new MyTimerTask(), 0, 1000);
```

Ellenőrző kérdések / tippek

- Magyarázza el a Java ME platformfüggetlenség elvét!
- Rajzolja fel és magyarázza el a MIDlet életciklus modellt?
- Vázolja fel egy Java ME alkalmazás kódját, mely megjeleníti a „Hello Mobil Fejlesztők!” feliratot!
- Készítsen egy Canvas osztályt, mely fekete háttéren egy piros négyzetet jelenít meg!
- Vázolja fel egy Java ME alkalmazás kódját, mely másodpercenként egy ‚#’ jelet fűz a kijelző tartalmához!
- Tippek:
  - Mindig különítsük el az életciklus függvényeket a forráskódban!
  - Soha ne legyen túl nagy a MIDlet osztály!

Mobilszoftverek

J2-Gy Általános mobil alkalmazásfejlesztési elvek

Java ME

Ekler Péter

peter.ekler@aut.bme.hu

Tartalom

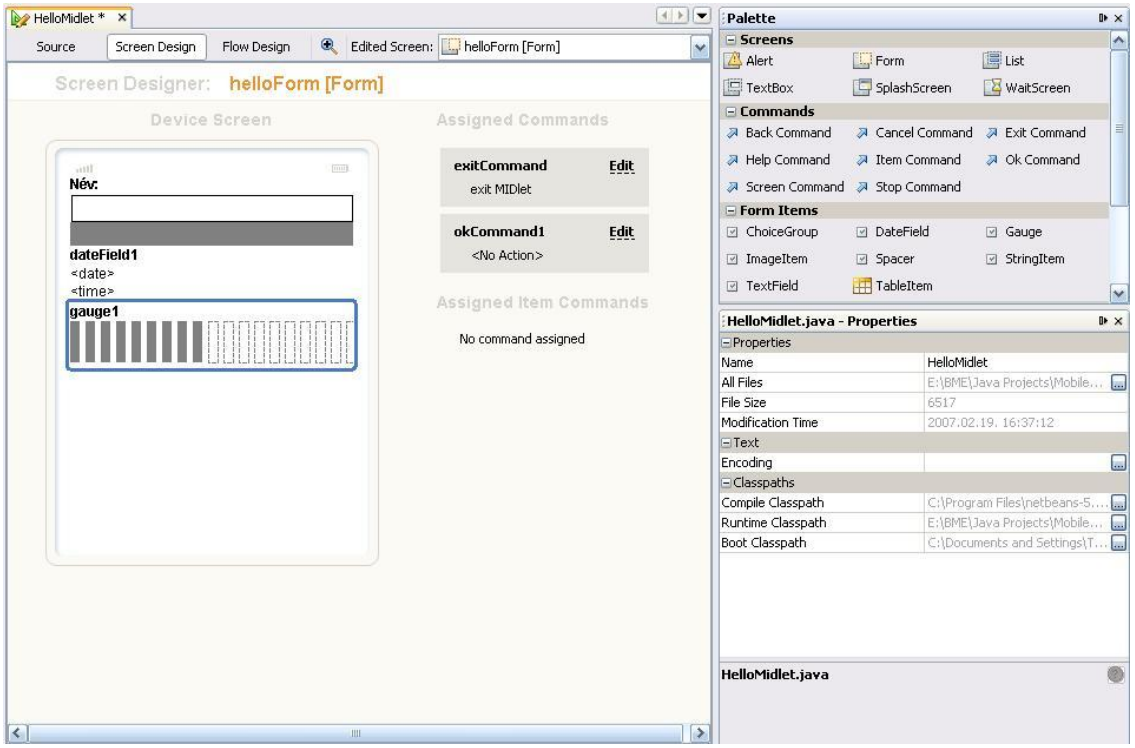
- Példa: UI tervezés/készítés
  - Beépített GUI elemek
  - Canvas alapú
  - Külső osztálykönyvtár használata: LWUIT
- Demo: MobTorrent
- Helymeghatározás Java ME platformon
- Elmélet: Gyakori tervezési minták mobil szoftvereknél
- Példa: Szálkezelés – blokkolódó műveletek
- Példa: Engine-UI szétválasztása

UI fejlesztés Java ME platformon

- Beépített elemek használata (tipikusan Form és rajta Item-ek)
- Canvas alapú felület
- Külső osztálykönyvtár használata
  - LWUIT
  - J2ME Polish

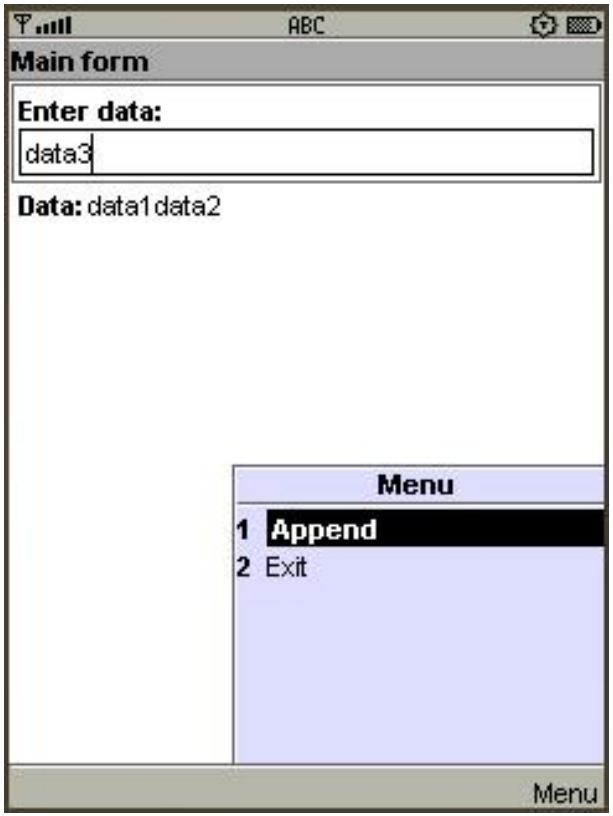
Felhasználói felület kialakítása

- A legtöbb fejlesztőeszközben megtalálható egy ScreenDesigner nevű eszköz (NetBeans Screen Designer):



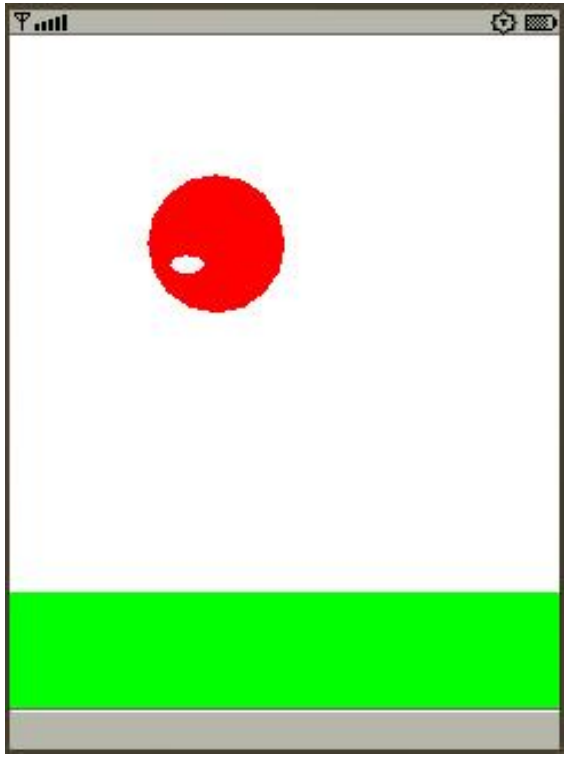
Beépített GUI elemek – a feladat

- Form, rajta egy TextField és egy Stringitem
- Append: TextField tartalmának hozzáfűzése a StringItem-hez
- Exit: kilépés



Canvas – a feladat

- Canvas osztály, mely az alábbi rajzot jeleníti meg



Canvas – a megoldás

```
public class FirstCanvas extends Canvas {

   private FirstMidlet midlet;

   public FirstCanvas(FirstMidlet aMidlet) {
      midlet = aMidlet;
   }

   public void keyPressed( int keycode ) {
      midlet.exitMIDlet();
   }
```

```
   public void paint( Graphics g ) {
         g.setColor(255,255,255);
         g.fillRect(0,0,getWidth(),getHeight());

         g.setColor(255,0,0);
         // teljes kört rajzolunk (0-360 fok)
         g.fillArc(60,60,60,60,0,360);

         g.setColor(255,255,255);
         // ellipszis rajzolása
         g.fillArc(70,95,15,8,0,360);

         g.setColor(0,255,0);
         g.fillRect(0,getHeight()-50,getWidth(),50);
      }
   }
```

Canvas duplabufferelés

- Általában támogatott, de nem minden készüléken
- boolean isDoubleBuffered()

```
public void paint(Graphics g) {
    if (bufferImage == null)
        initializeDraw();
    // image megjelenítése egy lépésben
    g.drawImage(bufferImage, 0, 0,
        Graphics.TOP | Graphics.LEFT);
}

// rajzolás elvégzése a memóriába
private void initializeDraw() {
    int w = getWidth();
    int h = getHeight();
    // kijelző méretével megegyező kép létrehozása
    bufferImage = Image.createImage(w, h);
    Graphics g = bufferImage.getGraphics();
    g.drawRect(0, 0, w - 1, h - 1);
    g.drawLine(0, 0, w - 1, h - 1);
    g.drawLine(w - 1, 0, 0, h - 1);
}
```

Mi is az az LWUIT?

- Lightweight User Interface Toolkit
  - SWING-szerű megközelítés
  - Erős hangsúly a hordozhatóságon kis erőforrás-igény mellett.
  - Eddigi Java ME UI Tapasztalatok figyelembe vétele. (Pl: hétköznapi, de LCDUIból hiányzó kontrollok implementálása)
  - A fejlesztés jelenleg is tart, de az eredmények már látványosak...

Legfőbb szolgáltatások I.

- SWING szerű MVC-patternes hozzáállás (például listáknál)
- Layoutok
- “PLAF” (Pluggable Look & Feel)
- Fontok
- TouchScreen támogatás
- Animációk és Transitionok
- Gazdag widget készlet

Legfőbb szolgáltatások II.

- Opcionális 3D Integráció (ha a készülék támogatja)
- Painterek
- Modális dialógusablakok
- Különböző segédprogramok (például témaszerkesztő)
- Lokalizációs lehetőségek
- SVG támogatás (ha a készülék támogatja)

Licensz

- “Early Access Binary”: Sun License Agreement
- Forráskód: GPL2v2 + classpath exception
- Magyarul:
  - A bináris gond nélkül használható kereskedelmi alkalmazásokban :)
  - Ha bele kell javítanunk a forráskódba, akkor már nem :(
    - Jelenleg is viták a fórumokon ezzel kapcsolatban

Hasznos LWUIT linkek

- LWUIT home: https://lwuit.dev.java.net/
  - LWUIT 1.4
  - Letöltés (Bináris)
  - SVN információk (Forráskód eléréshez)
  - Dokumentáció
  - Hírek
- LWUIT blog: http://lwuit.blogspot.com/
  - Nem feltétlenül triviális problémák megoldásai

LWUIT DEMO

- A bemutatott példaalkalmazás elérhető az LWUIT hivatalos honlapjáról
- A példaforráskód a hivatalos tutorialok alapján készült:
  - https://lwuit.dev.java.net/nonav/tutorial/index.html

LWUIT – a feladat

- Készítsünk egy teljes képernyős listát
- Adjunk néhány szöveges elemet a listához

LWUIT – a megoldás

```
public void initialize()
   {
      Display.init(this);
      Form f = new Form("lista");
      f.setLayout(new BorderLayout());
      f.setScrollable(false);

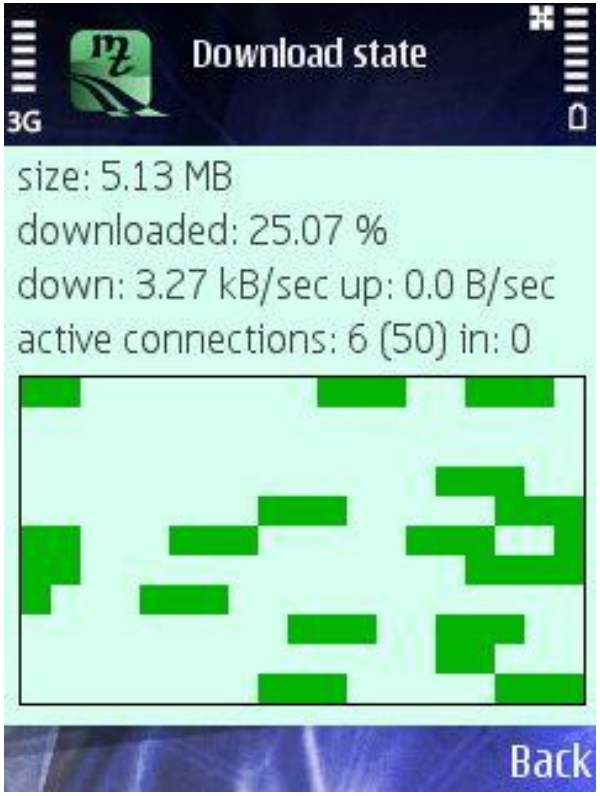
      List l = new List();
      l.setFixedSelection(List.FIXED_NONE_CYCLIC);
      l.addItem(new String("item1"));
      l.addItem(new String("item2"));
      l.addItem(new String("item3"));

      f.addComponent(BorderLayout.CENTER,l);

      f.show();
   }
```

DEMO - MobTorrent

- Tipikus példa UI és Engine szétválasztásra



Helymeghatározás példa

- Készítsünk egy alkalmazást, mely a 10 másodperces intervallummal ellenőrzi a koordinátákat és az új pozíciót megjeleníti a kijelzőn
- Főbb osztályok:
  - *Criteria: kért pozíció pontossága*
  - *LocationProvider: helymeghatározó manager*
  - *LocationListener: értesítési interface*
  - *QualifiedCoordinates: pozícióleíró objektum*

PositionManager osztály 1/3

```
public class PositionManager implements LocationListener {
    private LocationProvider locationProvider = null;
    private MidletLocationRefresher midlet;

    public PositionManager(MidletLocationRefresher aMidlet) {
        midlet = aMidlet;
    }

     public void startRequestLocation() {
        // if first request
        if (locationProvider == null) {
            Criteria cr= new Criteria();
            cr.setHorizontalAccuracy(500); cr.setVerticalAccuracy(500);
            try {
                locationProvider = LocationProvider.getInstance(cr);
            } catch (Exception ex) {ex.printStackTrace();}
        }
        try {
            locationProvider.setLocationListener(this, 10, -1, -1);
        } catch (Exception ex) {ex.printStackTrace();}
    }
```

PositionManager osztály 2/3

```
public void locationUpdated(LocationProvider aLocationProvider, Location aLocation) {
       if (aLocation != null && aLocation.isValid()) {
           QualifiedCoordinates coord = aLocation.getQualifiedCoordinates();
           if(coord != null ) {
               // new coord is considered when the horizontal and vertical accuracy is
  smaller than 1000.0
               if (coord.getHorizontalAccuracy() < 1000.0 && coord.getVerticalAccuracy() <
  1000.0) {
                   String lastCoords=coord.getLatitude()+","+coord.getLongitude();
                   String lastCoordsVel = ""+aLocation.getSpeed();
                   String lastCoordsHorAcc = ""+coord.getHorizontalAccuracy();
                   String lastCoordsVerAcc = ""+coord.getVerticalAccuracy();

                   midlet.refreshCoords(lastCoords);
               }
           }
       }
    }
```

PositionManager osztály 3/3

```
public void providerStateChanged(LocationProvider aLocationProvider, int aStatus) {
        switch (aStatus)
        {
            case LocationProvider.AVAILABLE:
                //
                break;
            case LocationProvider.OUT_OF_SERVICE:
                //
                break;
            case LocationProvider.TEMPORARILY_UNAVAILABLE:
                //
                break;
        }
    }
}
```

Position Midlet

```
public class MidletLocationRefresher extends MIDlet {
    Form f;
    PositionManager p;

    private void initialize() {
        f = new Form("Koordináták");
        Display.getDisplay(this).setCurrent(f);
        p=new PositionManager(this);
        p.startRequestLocation();
    }

    public void refreshCoords(String lastCoords) {
        f.append(lastCoords);
    }

    public void startApp() {
        initialize();
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
}
```

Tervezési minták

- Megkönnyítik és meggyorsítják az alkalmazásfejlesztést
- Fejlesztői „szleng”
- Többször felhasználható kódrészek
- Többnyire „platformfüggetlenek”☺
- Tipikus példa mobiloknál (is): Engine-UI szétválasztás (többször fogjuk használni példákban)

Singleton I.

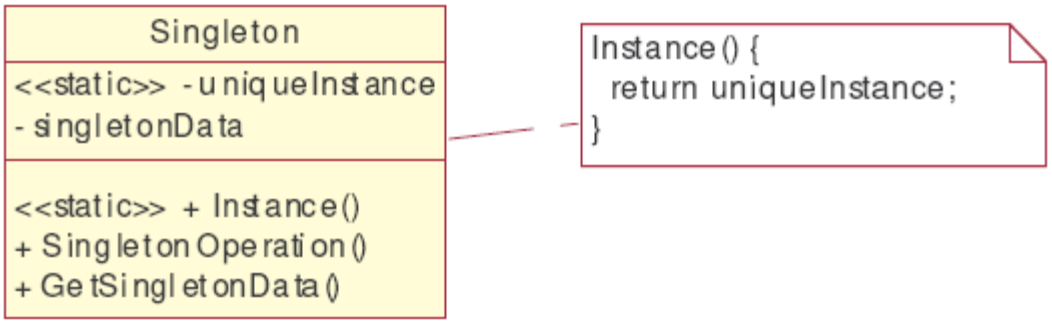
- **Célja**
  - Biztosítja, hogy egy osztályból csak egy példányt lehessen létrehozni, és ehhez az egy példányhoz globális hozzáférést biztosít
- **Elég gyakran van rá szükség**
  - egy ablakkezelő objektum
  - egy „engine” objektum
  - stb.

Singleton II.

```
public class MyEngine
{
    private static MyEngine instance = null;

    protected MyEngine(){}

    public static MyEngine getInstance()
    {
        if (instance==null)
        {
            instance = new MyEngine();
        }
        return instance;
    }
}
```



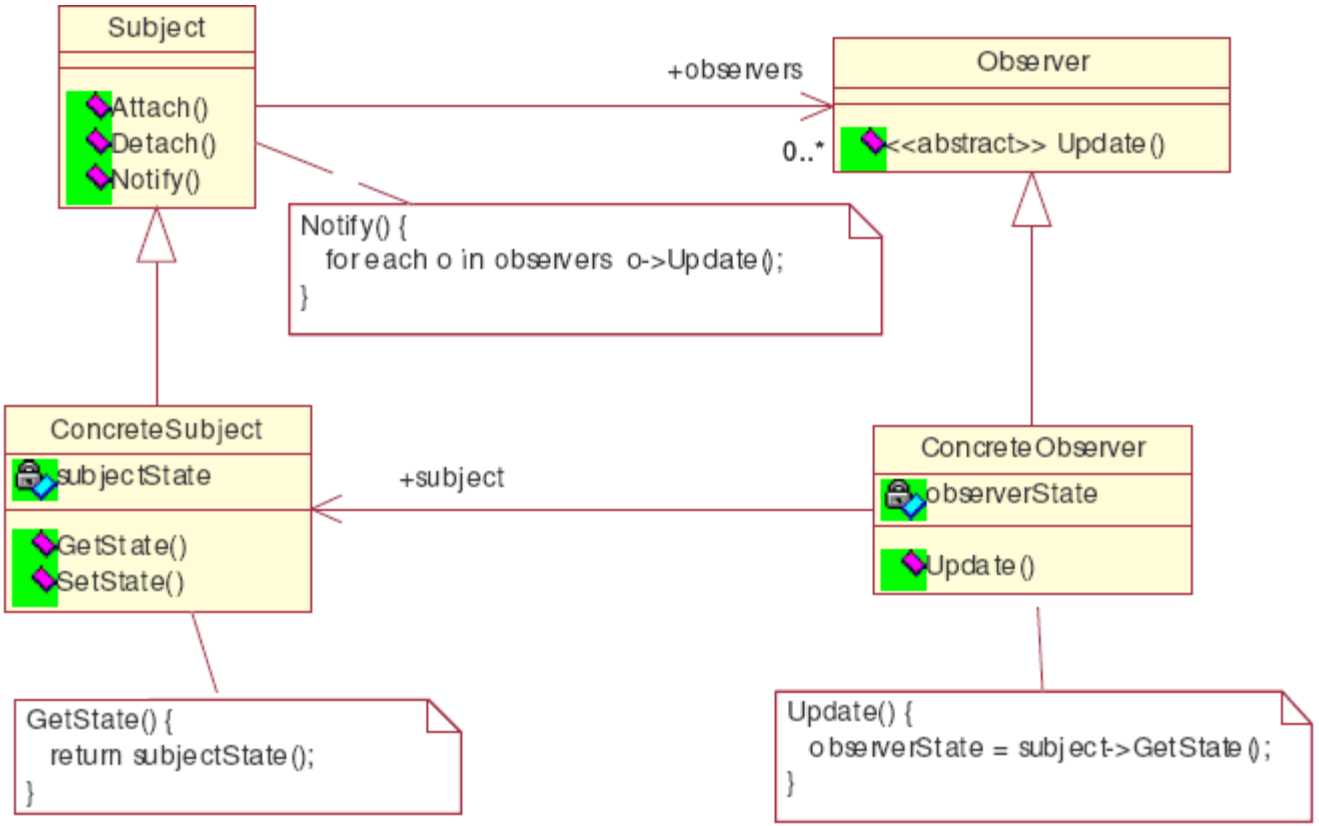
Observer I.

- **Cél**
  - Hogyan tudják az objektumok értesíteni egymást állapotuk megváltozásáról anélkül, hogy függség lenne a konkrét osztályaiktól
- Példa: MVC vagy Document-View architektúra
  - A felhasználó megváltoztatja az egyik nézeten az adatokat, hogyan frissítsük a többit? Közvetlen függvényhívással? – NEM!

Observer II.

- Emeljük ki az adatokat és az azon értelmezett műveleteket egy modell osztályba
- A modellhez különböző view-kat (observers) lehet beregisztrálni
- Ha valamelyik view megváltoztatja a modell adatait, a modell értesíti az összes beregisztrált view-t a változásról
- Az értesítés hatására a view lekérdezi a modell állapotát és frissíti magát
- A modell csak egy közös view (observer) interfészen keresztül tárolja a beregisztrált view-kat

Observer III.



Engine-UI szétválasztása - példa

MyObserver – nem egészen Observer

```
public interface MyObserver
{
    public abstract void fastEvent();

    public abstract void slowEvent();
}
```

MyEngine I.

```
public class MyEngine {
    private final int KFastEventTime = 1;
    private final int KSlowEventTime = 3;

    private static MyEngine instance = null;
    private MyObserver observer;
    private boolean active = false;
    private int counter = 0;
    private EngineThread engineThread;

    private MyEngine(MyObserver aObserver) {
        observer = aObserver;
    }
```

```
public static MyEngine
 getInstance(MyObserver aObserver) {
    if (instance==null)
        instance = new
 MyEngine(aObserver);
    return instance;
}

public void startEngine() {
    active = true;
    engineThread = new EngineThread();
    engineThread.start();
}
```

MyEngine II.

```
public void stopEngine() {
    active = false;

    try {
        engineThread.join();
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
}

public boolean isActive() {
    return active;
}
```

```
private class EngineThread extends Thread {
    public void run() {
        while(isActive()) {
            counter++;
            if (counter % KFastEventTime == 0)
                observer.fastEvent();
            if (counter % KSlowEventTime == 0)
                observer.slowEvent();

            // sleep 1 sec
            try {
                sleep(1000);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
    }
}

} // osztály vége
```

MyMidlet I.

```
public class MobMidlet extends MIDlet implements
    CommandListener, MyObserver {
    private int fastEventOccurs = 0;
    private int slowEventOccurs = 0;
    private StringItem fastEventStringItem = null;
    private StringItem slowEventStringItem = null;

    private Command startCommand = null;
    private Command stopCommand = null;
    private Command exitCommand = null;
```

```
private void initialize() {
    Form f = new Form("Observer example");

    fastEventStringItem = new StringItem("Fast
    event occurs: ", "0");
    slowEventStringItem = new StringItem("Slow
    event occurs: ", "0");

    f.append(fastEventStringItem);
    f.append(slowEventStringItem);

    startCommand = new Command("Start",
    Command.SCREEN, 0);
    stopCommand = new Command("Stop",
    Command.SCREEN, 1);
    exitCommand = new Command("Exit",
    Command.EXIT, 2);

    f.addCommand(startCommand);
    f.addCommand(stopCommand);
    f.addCommand(exitCommand);

    f.setCommandListener(this);

    Display.getDisplay(this).setCurrent(f);
}
```

MyMidlet II.

```
public void fastEvent() {
    fastEventOccurs++;

   fastEventStringItem.setText(""+fastEventOccu
   rs);
}

public void slowEvent() {
    slowEventOccurs++;

   slowEventStringItem.setText(""+slowEventOccu
   rs);
}
```

```
public void commandAction(Command c,
    Displayable d) {
    if (c.equals(startCommand)) {
        if
    (!MyEngine.getInstance(this).isActive())

    MyEngine.getInstance(this).startEngine();
    }
    else if (c.equals(stopCommand)) {

    MyEngine.getInstance(this).stopEngine();
    }
    else if (c.equals(exitCommand)) {

    MyEngine.getInstance(this).stopEngine();
        destroyApp(true);
        notifyDestroyed();
    }
}
```

MyMidlet III.

```
public void startApp() {
    initialize();
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

} // osztály vége
```

Az eredmény


Observer example
Fast event occurs: 16
Slow event occurs: 5
Menu
1 Start
2 Stop
Exit
Menu
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
SHIFT
SPACE

Gyakorló példák

- Készítsünk Barkóba alkalmazást
- Készítsünk Canvas alapú pattogó labdát a Thread osztály felhasználásáva
- Készítsünk űrlapot LWUIT segítségével személyes adatok beolvasására
- Alkalmazzuk az Observer designe patternt az előző Engine által szolgáltatott értékek többféle módú megjelenítésére (stringitem, gauge, külön Canvas, stb.)

Mobilszoftverek

J3 - Adat- és hálózatkezelés

Java ME

Ekler Péter

peter.ekler@aut.bme.hu

Tartalom

- Adattárolási elvek mobil eszközökön
- Személyes adatok elérése (Contacts, Calendar, ToDo)
- Példa: telefonkönyv listázása
- Fájlrendszer
- Beépített adatbázis
- Hálózatkezelés
  - TCP/IP kapcsolatok
  - HTTP kapcsolat
- Példaalkalmazás Térkép szolgáltatások eléréséhez

Adattárolási elvek mobil eszközökön

- Szokásos háttértár: beépített telefon memória és kiegészítő SD kártya (vagy merevlemez)
- Személyes adatok: telefonkönyv, naptár, tennivalók lista
- Alkalmazásonként saját adatbázis

Melyik módszert mikor alkalmazzuk?

- Háttértár:
  - Ha nagy adatokkal dolgozunk
  - Ha az alkalmazás jellege megkívánja (például letöltünk valamit)
- Személyes adatok:
  - Fontos: ezek az adatok kívülről, általános módon is elérhetők
  - Szinkron az alkalmazás és a telefon között (átlátszóvá tesszük az alkalmazás adatkezelését)
- Alkalmazás saját adatbázisa:
  - Beállítások elmentése
  - Kisebb adatok átmeneti tárolása

Jogosultság és Biztonság

- Elérheti-e egy alkalmazás a személyes adatokat?
- Elérheti-e egy alkalmazás a fájlrendszert?
- Elérhet-e egy alkalmazás saját adatbázist?
- Válasz: alkalmazás aláírása, ellenőrzése, biztonsági kérdések

Java ME – JSR 75

- JSR 75:
  - PIM (Personal Information Management) API
  - FileConnection API
- Napjainkban szinte mindegyik készülék támogatja
- Ökölszabály: amelyik készülék támogat külső háttértárat (SD kártya), abban megtalálható

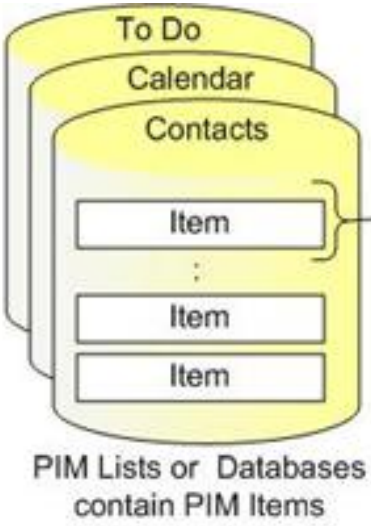
PIM (Personal Information Management)

- A kommunikáció után a legfontosabb funkció a telefonon
- ContactList, EventList, TodoList
- Emulátor is támogatja (pl. VCard formátumban a contactokat)
- Szorosabb kapcsolat az alkalmazásunk és az azt futtató készülék között

PIM API létezésének ellenőrzése

```
// Check that PIM Optional Package is available
String v = System.getProperty( "microedition.pim.version" );
if( v != null )
{
        // PIM API is available
}
else
{
        // PIM API is not available
}
```

PIM API felépítése



PIM Item

PIM Item

Each PIM Item has Fields. Each field has: ID, Label, Data Type, Zero or more Values, Zero or more Attributes

Field ID	*Label*	*Data Type*	*Values...*	*Attributes...*
106	Contact.NAME	PIMItem.STRING_ARRAY	First name | Middle name | Last name	PIMItem.ATTR_NONE
100	Contact.ADDR	PIMItem.STRING_ARRAY	Street | Locality | Region | Postal Code	Contact.ATTR_HOME
:	:	:	:	:
103	Contact.EMAIL	PIMItem.STRING	emailAddr@host.com	Contact.ATTR_HOME
:	:	:	:	:

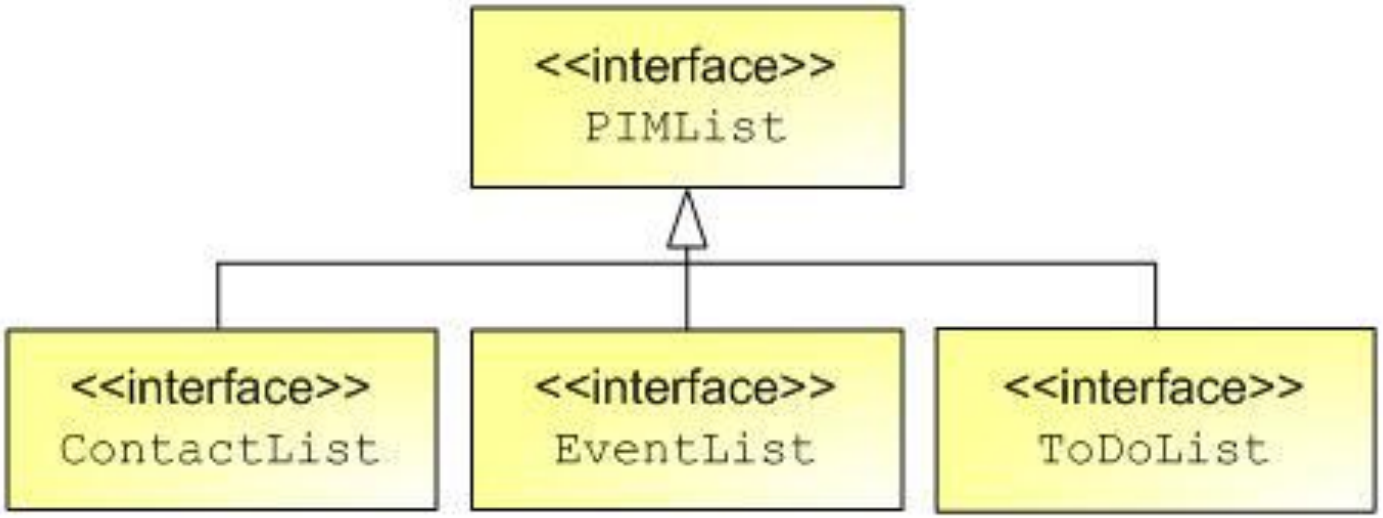
Forrás: http://developers.sun.com/mobility/apis/pim/pim1/

PIM osztály

- Absztrakt osztály, elérést biztosít a PIM implementációhoz
- Fontosabb függvényei:
  - getInstance(…) – statikus osztály a PIM eléréséhez (Singleton)
  - listPimList(…) - létező PIM adatbázisnevek lekérése
  - openPimList(…) - abstract method to open the default database, or open a specific database by name
  - fromSerialFormat(...) és toSerialFormat(...) – PIM adatok importálása és exportálása
  - supportedSerialFormats(...) – PIM adatcsere módok lekérdezése, pl: vCard version 2.1 és vCalendar version 1.0

PIMList

- PIM listák elérése

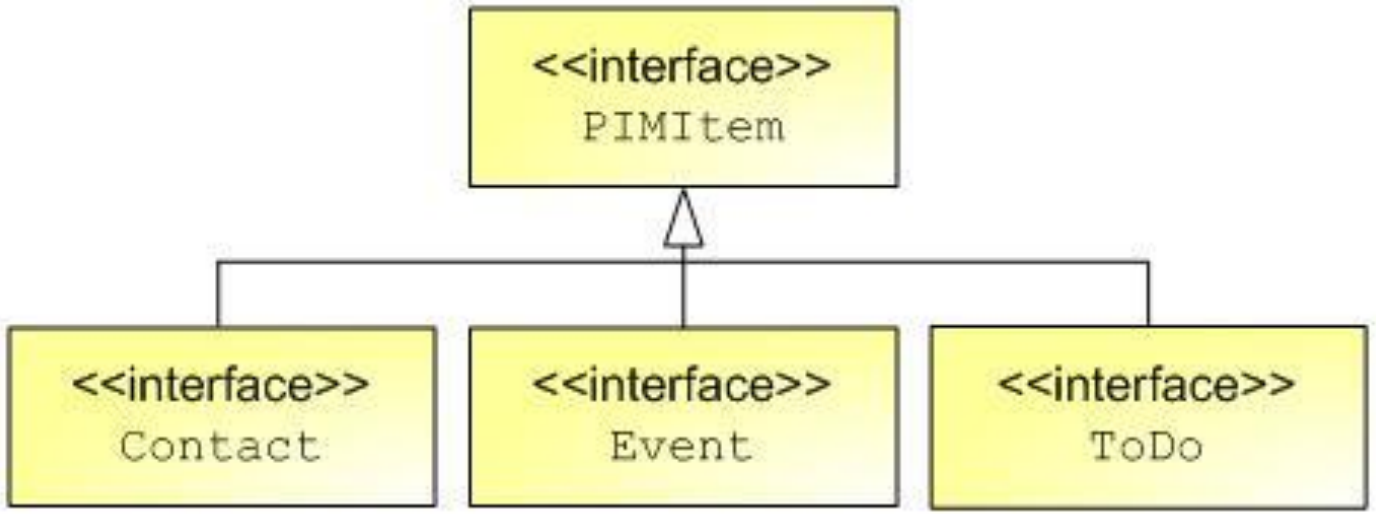


PIMList függvények

- getSupportedFields(…) és isSupportedField(…) – támogatott mezők (cím esetén: ország, város, utca, stb.)
- getSupportedAttributes(…) és isSupportedAttribute(…) – támogatott attribútumok (cím esetén: otthoni, munkahelyi, stb.)
- items(…) – A listában található PIMItem-eket adja vissza Enumeration-on keresztül

PIMItem

- PIM lista elemek elérése (személy, naptár és tennivaló bejegyzés)



PIMItem függvények

- getFields(…) – mezők lekérdezése
- getAttributes(…) – attribútumok lekérdezése
- getBinary(), getBoolean(), getDate(), getInt(), getString(), getStringArray() – érték lekérdezése egy mezőhöz (név mező – keresztnév, vezetéknév, stb.)
- setBinary(), setBoolean(), setDate(), setInt(), setString(), setStringArray() – érték beállítása
- commit() – PIMItem mentése

Példa: telefonkönyv listázása

```
private class MyContactLister extends Thread {
   public void run() {
      // Check that PIM Optional Package is available
      String v = System.getProperty( "microedition.pim.version" );
      if( v != null ) {
         PIM singleton = PIM.getInstance();
         ContactList cl = null;
         try {
            cl = (ContactList)singleton.openPIMList(PIM.CONTACT_LIST,
               PIM.READ_ONLY );
            Enumeration contactsEnum = cl.items();
            while( contactsEnum.hasMoreElements() ) {
               Contact contact = (Contact) contactsEnum.nextElement();
               get_listContacts().append(contact.getString( Contact.FORMATTED_NAME, 0 ),null);
            }
         }
         catch( Exception e){
            get_mainForm().append(e.getMessage());
         }
      }
   }
}
```

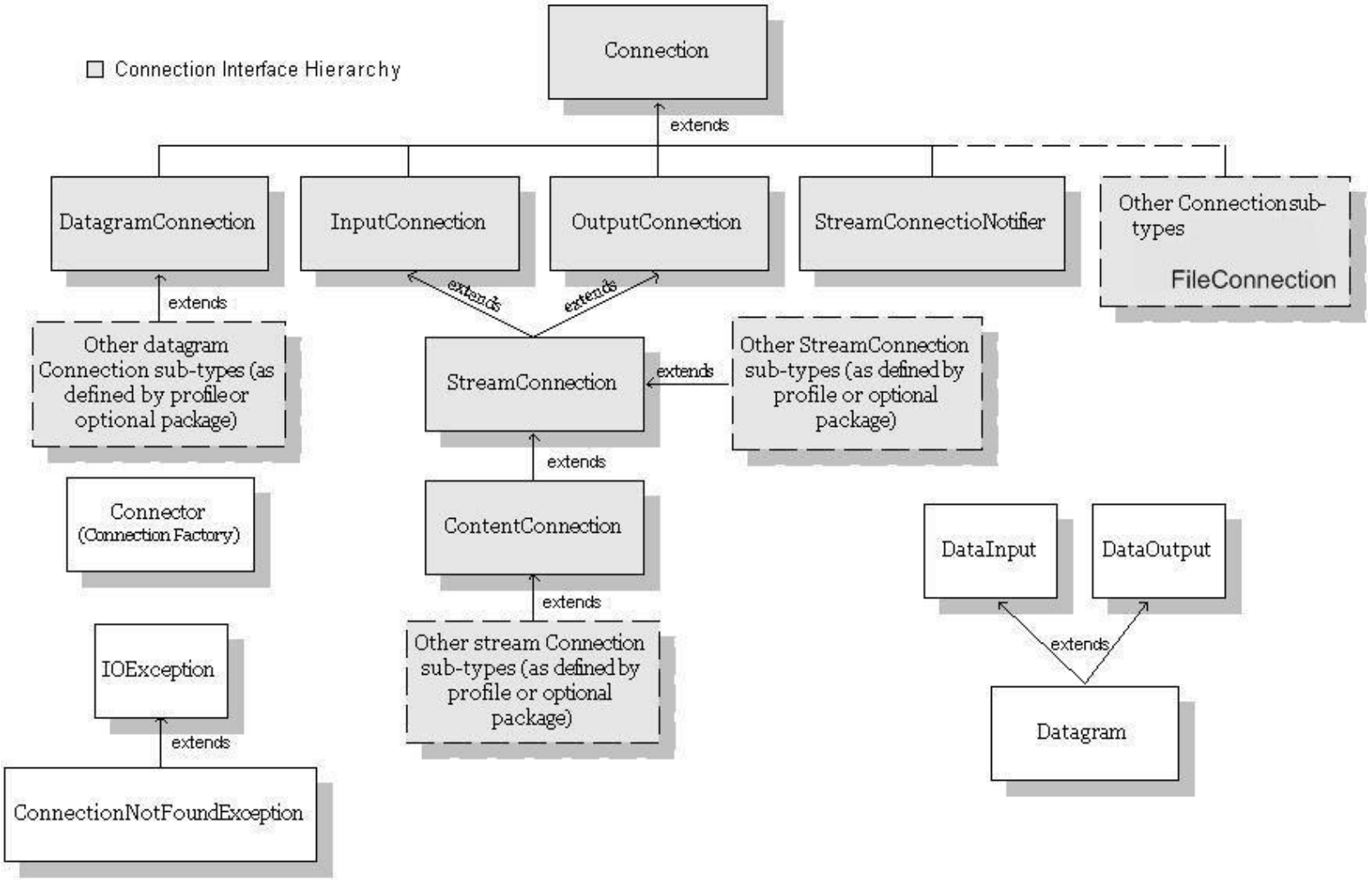
Általános kapcsolatkezelés

- Kapcsolattípusok széles skálájának kezelése
- A lehető legkevesebb megszorítás
- Eltérő hálózati technológiák azonos módon való kezelése, az adott technológiától függetlenül
- Hatékony erőforrás felhasználás
- Energiafelhasználás figyelembevétele

Generic Connection Framework I.

- `javax.microedition.io`
- Eredetileg hálózati kapcsolatok általános kezelése, de napjainkban egyre több újabb csomag felhasználja, például:
  - Fájlkezelés
  - Bluetooth alapú kommunikáció
- Többféle protokoll támogatása

Generic Connection Framework II.



Connector osztály

- `Connection c = Connector.open(String connectionString);`
- Factory jellegű osztály
- Többféle kapcsolatot nyithatunk meg vele, a megfelelő kapcsolódásleíró String megadásával: "protocol:address;parameters„
- A Connection helyén, konvertálás után a megfelelő kapcsolattípus állhat:
  - HttpConnection
  - FileConnection
  - StreamConnection
  - Stb.

```
FileConnection  conn = (FileConnection)
Connector.open("file:///E:/log.txt",
Connector.READ );
```

FileConnection API

- `javax.microedition.io.file` csomag
- Két interface:
  - `FileConnection`: Fájlok és könyvtárak eléréséhez
  - `FileSystemListener`: Listener, mely értesítést kaphat, ha változás történik a fájlrendszeren
- Három osztály:
  - `FileSystemRegistry`: Listener-ek központi tárolója
  - `ConnectionClosedException`: kapcsolat megszakadását jelzi
  - `IllegalModeException`: jelzi, ha például nincs meg a szükséges biztonsági szint (nincs írási jog)

Meghajtók listázása

- Emulátoron egy meghajtó: root1
  - …/toolkit/appdb/DefaultColorPhone/filesystem/root1
- A meghajtók listája:

```
private void getRootList() {
      Enumeration drives =
 FileSystemRegistry.listRoots();
      System.out.println(„Meghajtók listája:");
      while(drives.hasMoreElements()) {
         String root = (String)
 drives.nextElement();
         System.out.println("\t"+root);
      }
  }
```

Fájl/könyvtár létezésének vizsgálata

- Sok esetben szükség van megvizsgálni, hogy adott könyvtár/fájl létezik-e már

```
private boolean fileExist(String aPath)
{
    boolean result = false;
    FileConnection conn = null;
    try {
        conn = (FileConnection) Connector.open(
  "file:///"+aFilePath,Connector.READ );

        result = conn.exists();
        conn.close();
    } catch (IOException ex) {
        result = false;
    }
    return result;
}
```

Könyvtár létrehozása

- Könyvtár létrehozása az mkdir() függvényhívással történik

```
private boolean createFolder(String aFolderName)
{
    FileConnection conn = null;
    try {
        conn = (FileConnection) Connector.open(
  "file:///"+currentRoot+aFolderName+"/",
                Connector.WRITE );
        conn.mkdir();
        conn.close();
        return true;
    } catch (IOException ex) {
        ex.printStackTrace();
        return false;
    }
}
```

Fájl létrehozása tartalommal

- Fájl létrehozásához meg kell adnunk a fájl nevét a teljes elérési úttal

```
private void createFile(String aFileName, String aFileData)
{
    FileConnection conn = null;
    try {
        conn = (FileConnection) Connector.open(
  "file:///"+currentRoot+aFileName,Connector.WRITE );
        conn.create();

        OutputStream out = conn.openOutputStream();
        out.write(aFileData.getBytes());
        out.close();
        conn.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

Fájl tartalmának kiolvasása

```
private byte[] viewFile(String aFilePath)
{
    FileConnection conn = null;
    try {
        conn = (FileConnection) Connector.open( "file:///"+aFilePath,
                Connector.READ );

        InputStream in = conn.openInputStream();
        byte[] content = new byte[(int)conn.fileSize()];
        in.read(content);
        in.close();
        conn.close();
        return content;
    } catch (IOException ex) {
        return null;
    }
}
```

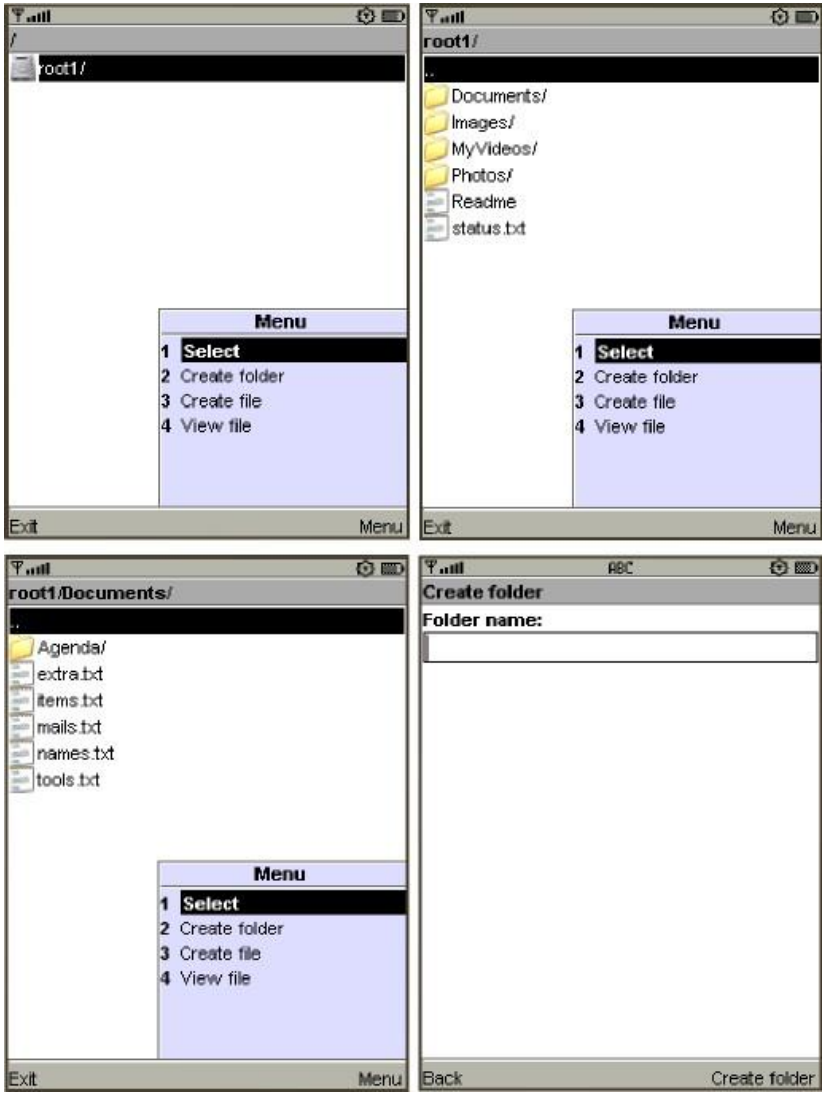
További hasznos függvények

- Fájl törlése: `delete()`
- Szabad terület: `availableSize()`
- Könyvtár vizsgálat: `isDirectory()`
- Rejtett-e: `isHidden()`
- Utolsó módosítás: `lastModified()`
- Könyvtár tartalmának listázása: `list()`
- Stb.

Gyakorló feladat

- Készítsünk egy alkalmazást, mely fájlböngészőként használható a mobil készüléken
- Az alkalmazás tudjon új fájlt és könyvtárat is létrehozni
- Az alkalmazás képes legyen a szöveges fájlok tartalmának megjelenítésére
- Kilépéskor tároljuk el RMS-ben, hogy melyik könyvtárban álltunk, betöltéskor pedig ezzel inicializáljuk az alkalmazást
- Hasonlóan a NetBeans FileSelect Dialog-hoz

A megoldás UI-a



Record Management System

- Rekord orientált adatbázis-kezelő rendszer
- Feladata, hogy egy mobil alkalmazás perzisztensen tárolhasson adatot
- Például az alkalmazás leállítása majd újraindítása után vissza lehessen tölteni az elmentett adatokat

Record orientáltság

- Az adatbázisban byte-ok tárolhatók
- Beépített típusok vagy objektumok közvetlen tárolása nem támogatott
- Ilyen adatokat előbb át kell alakítani byte tömbbé cast-olással, vagy szerializáció segítségével

Record orientáltság

- Az adatbázisban byte-ok tárolhatók
- Beépített típusok vagy objektumok közvetlen tárolása nem támogatott
- Ilyen adatokat előbb át kell alakítani byte tömbbé cast-olással, vagy szerializáció segítségével

RMS csomag

- Az RMS használatához a `javax.microedition.rms` package importálása szükséges
- A fő osztály a RecordStore osztály
- Méret lekérdezése: getSizeAvailable()
- Létrehozás:

```
RecordStore.openRecordStore(
  String recordStoreName,
  boolean createIfNecessary)
```

Példa: beállítások elmentése

```
public void storeDataToRMS()
{
    // open data streams
    ByteArrayOutputStream bout = new
    ByteArrayOutputStream();
    DataOutputStream dout = new DataOutputStream(bout);
    RecordStore rs = null;
    try {
        // save data to streams
        dout.writeUTF("hu");
        dout.writeUTF("192.168.1.0");
        dout.writeInt(10000);
        dout.flush();
                                                    }
        // store stream data to rms
        byte[] data = bout.toByteArray();
        rs = RecordStore.openRecordStore("mydatabase",
    true);
        try {
            rs.setRecord(1, data, 0,
    data.length);
        } catch (RecordStoreException ex) {
            rs.addRecord(data, 0, data.length);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    } finally {
        try {
            dout.close();
            bout.close();

    rs.closeRecordStore();
        } catch (Exception ex) {

    ex.printStackTrace();
        }
    }
```

Példa: beállítások betöltése

```
public void restoreDataFromRMS() {
    RecordStore rs = null;
    try {
        // open RecordStore
        rs = RecordStore.openRecordStore("mydatabase", true);
        if (rs != null)
        {
            byte[] data = rs.getRecord(1);
            // create streams
            ByteArrayInputStream bin = new
    ByteArrayInputStream(data);
            DataInputStream din = new
    DataInputStream(bin);                                    }

            // read data
            String language = din.readUTF();
            String ip = din.readUTF();
            int port = din.readInt();

            // close streams
            din.close();
            bin.close();
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    } finally {
        try {
            // close rms

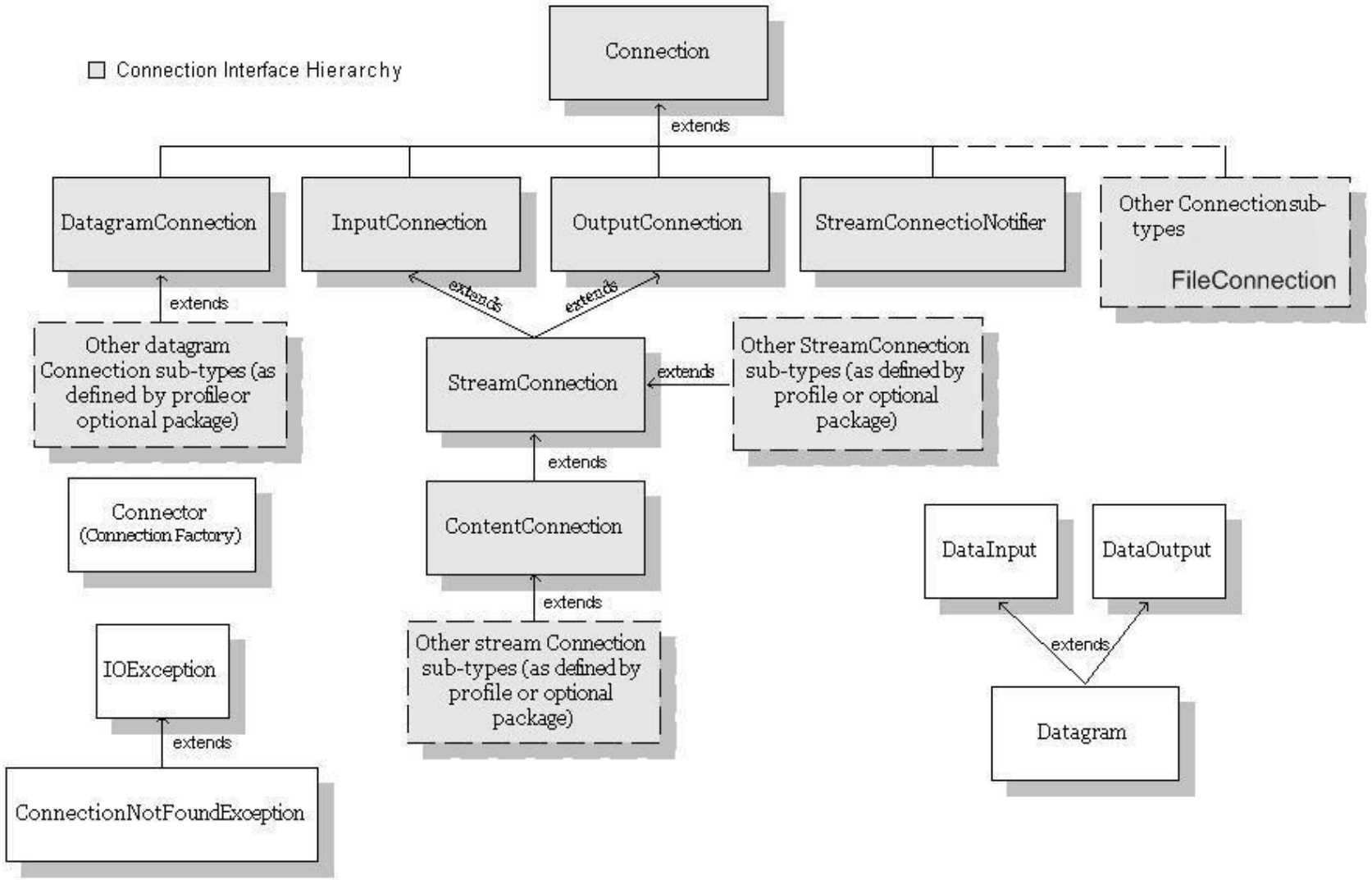
    rs.closeRecordStore();
        } catch (Exception ex) {

    ex.printStackTrace();
        }
    }
```

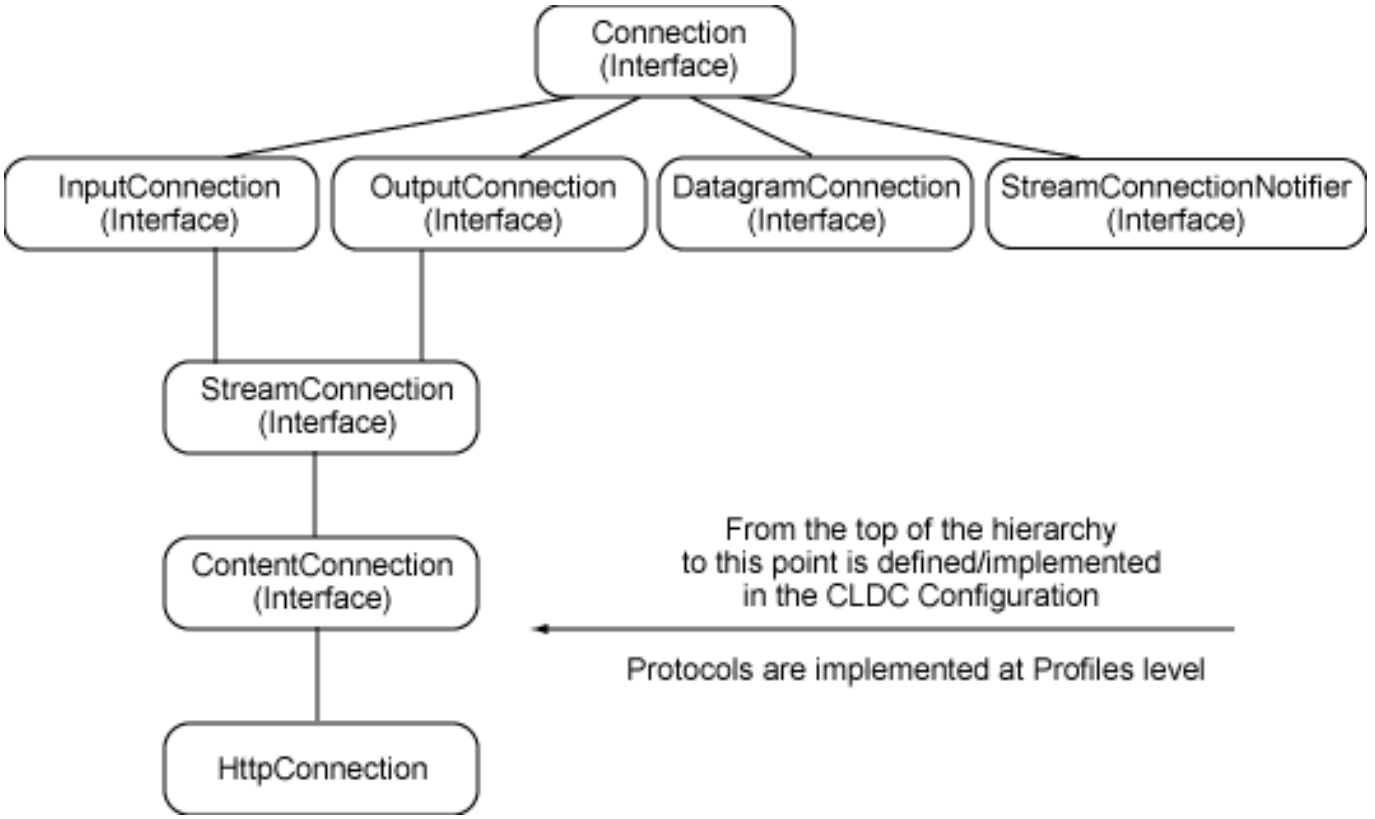
GCF – egyszerű bővíthetőség

- Magasszintű általánosítás
- Új kapcsolattípus támogatása egyszerűen:
  - Connection interface implementálása az új kapcsolattípust definiáló osztályban
  - Connector factory osztály biztosítása, mely támogatja az új kapcsolattípust
  - Kapcsolattípushoz tartozó URL definiálása
- Az új kapcsolattípus akár egy már meglévő Connection interface-t implementáló osztályból is származhat

Generic Connection Framework



Például: HttpConnection



Forrás:http://www.ibm.com/developerworks/java/library/j-j2me4/

GCF kapcsolat példák

URL scheme	Kapcsolat típus	GCF Connection osztály
datagram	Datagram	DatagramConnection
file	Fájl elérés	FileConnection, InputConnection
http	HTTP protokoll	HttpConnection
https	Biztonságos HTTP	HttpsConnection
comm	Soros I/O	CommConnection
sms, mms	Szöveges és multimédia üzenet	MessageConnection
socket, serversocket	Socket alapú kapcsolat	SocketConnection, ServerSocketConnection

GCF használatának főbb lépései

- Megnyitáshoz szükséges a Connector factory osztály és az URL
- A bezárás a létrehozott Connection feladata
- Az adott kapcsolat további használata a Connection osztályból leszármazott kapcsolatkezelő objektum feladata
- Az egyes Connection-ok használata egymástól jelentősen eltérhet

Példa: kapcsolat megnyitása

```
String url = "socket://www.tesztszerver.com:80";
...
SocketConnection socketConnection = null;
InputStream inputStream = null;
try {
    socketConnection = (SocketConnection)Connector.open(url);
    inputStream = c.openInputStream();
    ...
    // adatok beolvasása
    ...
}
catch (ConnectionNotFoundException cne) {}
catch (IllegalArgumentException iae) {}
catch (IOException ioe) {}
finally {
    try {
        if (inputStream != null) inputStream .close();
        if (socketConnection != null) socketConnection .close();
    } catch (Exception e) {}
}
```

Kapcsolat nyitás módjai

- A Connector osztály három `open()` függvényt definiál:
  - `open(String url)`
  - `open(String url, int mode)`
  - `open(String url, int mode, boolean timeouts)`
- A paraméterek jelentése:
  - Az URL a kapcsolat típusát írja le
  - A mód változó lehetséges értékei: READ, WRITE, vagy READ_WRITE (alapértelmezett)
  - A `timeouts` változóban adhatjuk meg, hogy kívánunk-e értesítést kapni timeout kivételről (`InterruptedIOException`), alapértelmezett értéke false

Példa: weboldal tartalmának beolvasása

```
public class MyHTTPReader extends Thread {
    public void run()
    {
        try {
            HttpConnection hc = (HttpConnection)
   Connector.open("http://127.0.0.1/index.html");
            InputStream is = hc.openInputStream();

            int ch;
            while ((ch = is.read()) != -1)
                contentStringItem.setText(
                  contentStringItem.getText()+(char)ch);
            is.close();
            hc.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

HttpConnection paraméterek

- Gyakorlatban fontos beállítani:
  - `setRequestMethod(HttpConnection.GET);`
  - `setRequestProperty("Accept","text/html, text/plain, *; q=.2, */*; q=.2");`

Google Map HTTP API

- Koordináták lekérdezése cím alapján:
  - http://maps.google.com/maps/geo?q=Budapest&output=csv&key=your_api_key
- Cím lekérdezése koordináták alapján (válasz: UTF-8):
  - http://maps.google.com/maps/geo?q=40.71,-73.96&output=json&sensor=true_or_false&key=your_api_key
- Térkép lekérése képként:
  - http://maps.google.com/staticmap?center=40.71,-73.96&format=jpeg&zoom=15&size=120x80&key=your_api_key
- Ne felejtsük a paraméterek URL encodolását, ha szükséges

URL encode

```
public static String URLEncode(String s) {
      StringBuffer tmp = new StringBuffer(); int i=0;
      try {
          while (true) {
              int b = (int)s.charAt(i++);
              if ((b>=0x30 && b<=0x39) || (b>=0x41 && b<=0x5A) || (b>=0x61 &&
  b<=0x7A)) {
                  tmp.append((char)b);
              }
              else {
                  tmp.append("%");
                  if (b <= 0xf) tmp.append("0");
                  tmp.append(Integer.toHexString(b));
              }
          }
      }
      catch (Exception e) {}
      return tmp.toString();
}
```

Reverse Geocode példa 1/2

```
new Thread() {
  public void run() {
        HttpConnection hc = null;
        InputStream is = null;
        try {
            // TODO: URL Encode the coordinates if needed
            hc = (HttpConnection) Connector.open(
                "http://maps.google.com/maps/geo?q="+
                coordsField.getString()+
                "&output=csv&sensor=false&key=your_api_key");
            hc.setRequestMethod(HttpConnection.GET);
            is = hc.openInputStream();
            int inputChar;
            ByteArrayOutputStream bf = new ByteArrayOutputStream();
            while ((inputChar = is.read()) != -1)
                bf.write(inputChar);}
```

Reverse Geocode példa 2/2

```
            // választ UTF-8-ban értelmezzük
            mainForm.append(new String(bf.toByteArray(),"UTF-8")+
            "\n");
            bf.close();
        } catch (Exception e) {e.printStackTrace();}
        finally {
            try {
                if(is != null)
                    is.close();
                if(hc != null)
                    hc.close();
            }
            catch(Exception e2) {e2.printStackTrace();}
        }
    }
}.start();
```

Gyakorló feladatok

- Milyen adattárolási lehetőségeket ismer mobil eszközökön?
- Hogyan oldja meg a Java ME a kapcsolatkezelés bővíthetőségét?
- Milyen adatok érhetők el a PIM-en keresztül?
- Készítsen egy szál osztályt, amely kilistázza a tennivalókat! (System.out.println(…) elég)
- Készítsen egy függvényt, mely az alkalmazás beállításait (egy string és egy int) elmenti perzisztens módon!
- Készítsen egy szál osztályt, mely a 192.168.0.100:10000-es címre socket-en keresztül elküldi a „hello world” stringet!

Mobilszoftverek

J4 – Gy Fejlett funkciók megvalósítása mobil eszközökre

Java ME

Tartalom

- Példa: Calendar bejegyzés elmentése
- Elmélet: WebService technológia mobil alkalmazásokban
- Példa: WebService példa
- Példa: Térkép megjelenítése képként koordináta változás esetén
- Példa: Fájl letöltő alkalmazás
- Példa: Bluetooth szerver
- Demo: MobSensor
- Példa: MP3 lejátszó alkalmazás

Adattárolási elvek mobil eszközökön

- Szokásos háttértár: beépített telefon memória és kiegészítő SD kártya (vagy merevlemez)
- Személyes adatok: telefonkönyv, **naptár**, tennivalók lista
- Alkalmazásonként saját adatbázis
- Alkalmazások által elérhető adatbázis – nem minden eszközön

Calendar – a feladat

- Készítsünk egy alkalmazást, amely egy adott bejegyzést ment a készülék naptárába.
- A bejegyzés paraméterezése céljából helyezzünk el egy form-on egy DateField-et és háromTextField-et.
- A DateField az esemény kezdési időpontját jelenti.
- Az első TextField az esemény időtartamát (például 30 perc) tárolja.
- A második TextField segítségével azt adhatjuk meg, mennyi idővel a kezdés előtt kérjük a riasztást.
- A harmadik TextField-ben az esemény tárgyát adhatjuk meg.

Calendar – a megoldás I.

```
public void storeEvent()
{
    String storeDuration, storeTopic, storeAlertBeforeInSec;
    Date storeDate;
    storeDuration = durationTextField.getString();
    storeAlertBeforeInSec = alertBeforeTextField.getString();
    storeTopic = topicTextField.getString();
    storeDate = dateTime.getDate();

    int length = 0;
    int alertBeforeInSec = 0;

    try {
        length = Integer.parseInt(storeDuration);
        alertBeforeInSec = Integer.parseInt(storeAlertBeforeInSec);
    } catch (NumberFormatException e) {
        return;
    }
```

Calendar – a megoldás II.

```
if (length <= 0) {
        showMessage("Duration needs to be positive");
    } else if (storeTopic == null || storeTopic.length() == 0) {
        showMessage("Subject not entered");
    } else {
        ByteArrayOutputStream out = new ByteArrayOutputStream(); // akkor kell, ha fájlba / netre ki akarjuk menteni
        EventList eventList = null;

        try {

            PIM pim = PIM.getInstance();
            String eventLists[] = pim.listPIMLists(PIM.EVENT_LIST);

            if (eventLists.length > 0) {
                eventList = (EventList) pim.openPIMList(
                        PIM.EVENT_LIST,
                        PIM.READ_WRITE, eventLists[0]);
                Event newEvent = eventList.createEvent();
```

Calendar – a megoldás III.

```
if (eventList.isSupportedField(Event.SUMMARY)) {
    newEvent.addString(Event.SUMMARY,
        PIMItem.ATTR_NONE,storeTopic);
}

if (eventList.isSupportedField(Event.START)) {
    newEvent.addDate(Event.START,PIMItem.ATTR_NONE,
        storeDate.getTime());
}

if (eventList.isSupportedField(Event.END)) {
    newEvent.addDate(Event.END,PIMItem.ATTR_NONE,
        storeDate.getTime() + 60 * 1000 * length);
}

if (eventList.isSupportedField(Event.ALARM)) {
    newEvent.addInt(Event.ALARM,PIMItem.ATTR_NONE,alertBeforeInSec);
}

// let's check that VCALENDAR/1.0 is supported
String supportedFormats[] =
      PIM.getInstance().supportedSerialFormats(
      PIM.EVENT_LIST);
```

Calendar – a megoldás IV.

```
for (int i=0;i<supportedFormats.length;i++) {
    if (supportedFormats[i].equals(
            "VCALENDAR/1.0")) { // támogatott-e a VCALENDAR/1.0

        PIM.getInstance().toSerialFormat(newEvent,
                out, "UTF-8","VCALENDAR/1.0");
        // out-ot használhatjuk nyers adatként szinkronizálásra
        break;
    }
}

if (out.size() == 0) {
    showMessage("VCALENDAR/1.0 not supported");
}

eventList.importEvent(newEvent);
newEvent.commit();
```

Calendar – a megoldás V.

```
            } else {
                showMessage("No Event list available");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {

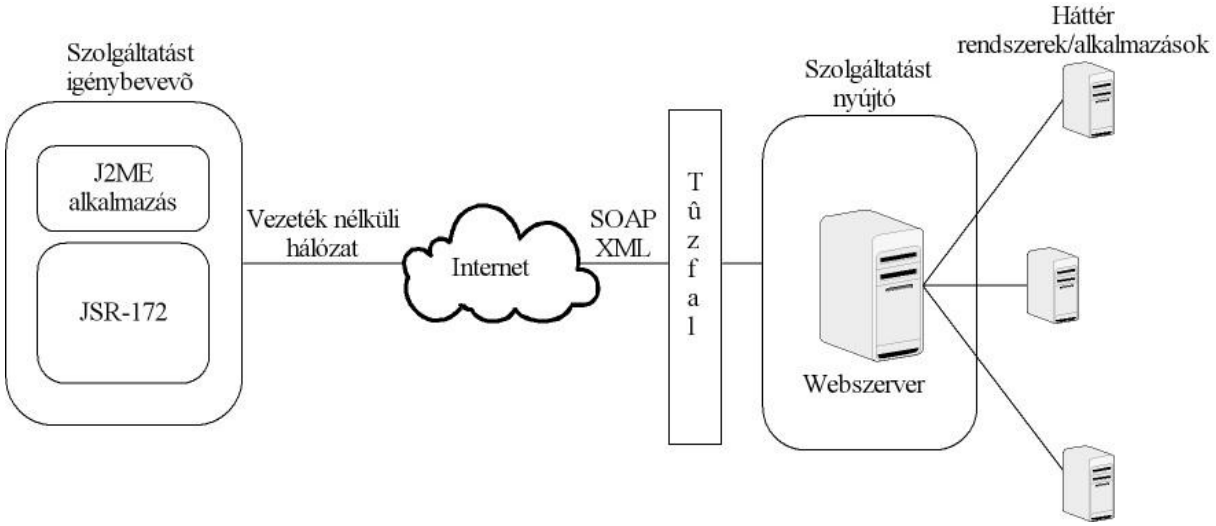
                if (eventList != null) {
                    eventList.close();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

WebService technológia Java ME platformon

- A JSR 172-es kiegészítés tartalmazza a Web Service (webszolgáltatás) API-t
- Az összes szükséges specifikációt tartalmazza (WS-I Basic Profile):
  - SOAP 1.1 (Simple Object Access Protocol): az adattovábbítást és kódolást specifikálja
  - WSDL 1.1 (Web Services Definition Language): specifikálja a távoli szolgáltatások leírási módját
  - XML 1.0: XML jelölőnyelv leírása
  - XML Schema leírás

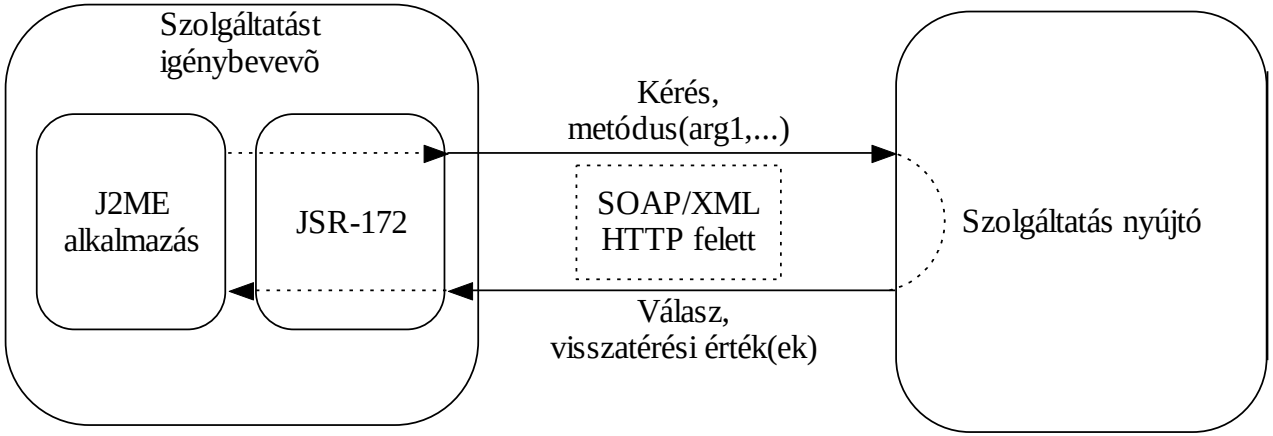
A technológia felépítése

- Kliens, Web Service felhasználó: J2ME MIDP alkalmazás JSR 172-es kiegészítést támogató mobil eszközön
- Hálózat: Az Internetkapcsolat és a hálózati protokollok összefoglalója a résztvevők között
- Szerver, Web Service kibocsátó: Tipikusan egy webszerver, amely a szolgáltatást nyújtja. Különféle háttér erőforrások csatlakozhatnak hozzá



A szolgáltatás elérése

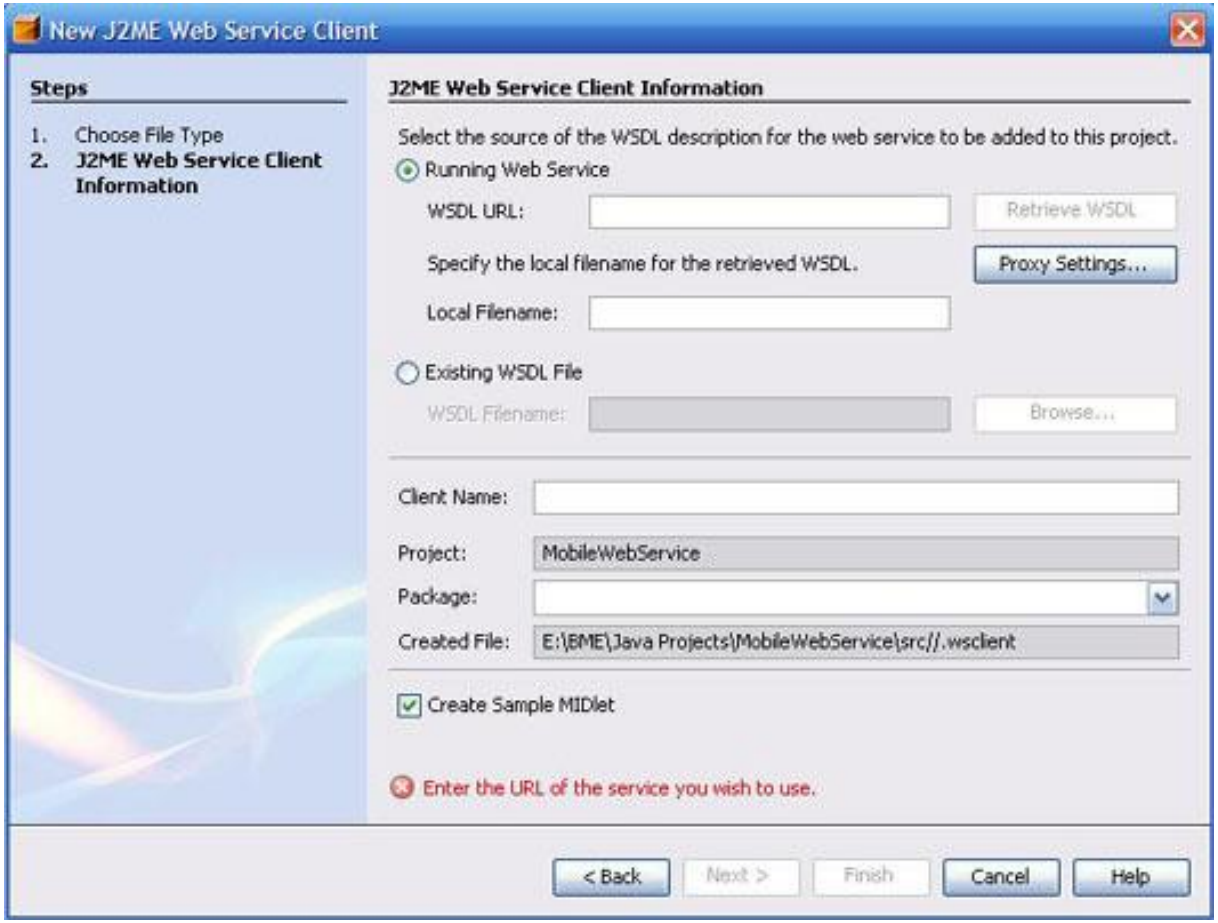
- Az alkalmazás meghívja a JSR 172 interfészét
- A JSR 172 elfedi a programozó elől a hálózati kommunikációt
- Szinkron jellegű hívásként valósul meg a távoli eljáráshívás



WebService hívás a NetBeans segítségével

- Fejlesztőkörnyezet által biztosított eszközök (Pl. NetBeans Mobility Pack)
- A Web Service WSDL alapján generálja a kódot
- Elfedi a hálózati kommunikációt
- Web Service hívás egyszerű függvényhívás formájában érhető el egy vezérlő objektumtól:
  - `MyProgramWebServiceSoap_Stub webService = new MyProgramWebServiceSoap_Stub();`
  - `String result = webService.exampleWebServiceCall(„hello”);`

NetBeans Mobility Pack – WebService stub generálás



WebService hívás – a feladat

- Készítsen egy webszolgáltatást tetszőleges platformon (Java EE, ASP.NET), mely a paraméterül kapott String-et megfordítja
- Készítsen egy mobil alkalmazást, mely igénybe veszi a szolgáltatást egy TextField-ben beolvasott String megfordítására

WebService hívás – a megoldás

- ASP.NET webservice:

```
public class WebServiceStringAdder : System.Web.Services.WebService
{
    [WebMethod]
    public string ReverseString(String origiString)
    {
        String result = "";

        for (int j = origiString.Length - 1; j > -1; j--)
        {
            result += origiString[j];
        }

        return result;
    }
}
```

WebService hívás – a megoldás II.

- Java ME oldalon a hívás, NetBeans stub generálás után:

```
public void reverseString()
{
    try {
            String res = webService.ReverseString(textField.getString());
            System.out.println(res);
    } catch (RemoteException ex) {
            ex.printStackTrace();
    } catch (Exception ex) {
            ex.printStackTrace();
    }
}
```

- Ahol a webService:

```
MobileMeetingWebServiceSoap_Stub  webService = new MobileMeetingWebServiceSoap_Stub();
```

Térkép frissítése új koordináta esetén

- Induljunk ki a 2. előadás példájából ahol a folyamatosan figyeltük a készülék koordinátáit és változáskor megjelenítettük a felületen az új értékeket
- Egészítsük ki az alkalmazást, hogy ne a koordinátákat, hanem a hozzájuk tartozó térképet jelenítsük meg képként folyamatosan
- Használjuk a Google Maps API-ját

MapManager osztály 1/3

```
public Image retrieveStaticImage(int width, int height,
    double lat, double lng, int zoom, String format,
    String apiKey) throws Exception {

    byte[] imageData = loadHttpFile(
        getMapUrl(width, height, lng,
        lat, zoom, format,apiKey));

    if (imageData != null)
        return Image.createImage(imageData,
          0, imageData.length);
    else
        return null;
}
```

MapManager osztály 2/3

```
private String getMapUrl(int width, int height, double
  lng, double lat, int zoom, String format,String apiKey)
{
    return "http://maps.google.com/staticmap?center=" +
        lat + "," + lng + "&format=" + format + "&zoom=" +
          zoom + "&size=" +
          width + "x" + height + "&key=" + apiKey;
}
```

MapManager osztály 2/3

```
private byte[] loadHttpFile(String url) throws
    Exception {
     HttpConnection hc = null;
     InputStream is = null;
     byte[] byteBuffer = null;
     try {
         hc = (HttpConnection)
           Connector.open(url);
         hc.setRequestMethod(
           HttpConnection.GET);
         is = hc.openInputStream();
         int inputChar;
         ByteArrayOutputStream bf = new
           ByteArrayOutputStream();
         while ((inputChar = is.read()) != -1) {
             bf.write(inputChar);
         }

         byteBuffer = bf.toByteArray();
         bf.close();
     }
     catch(Exception e) {
         e.printStackTrace();
     }
     finally {
         try {
             if(is != null)
                 is.close();
             if(hc != null)
                 hc.close();
         }
         catch(Exception e2) {
             e2.printStackTrace();
         }
     }
     return byteBuffer;
}
```

MapManager használata

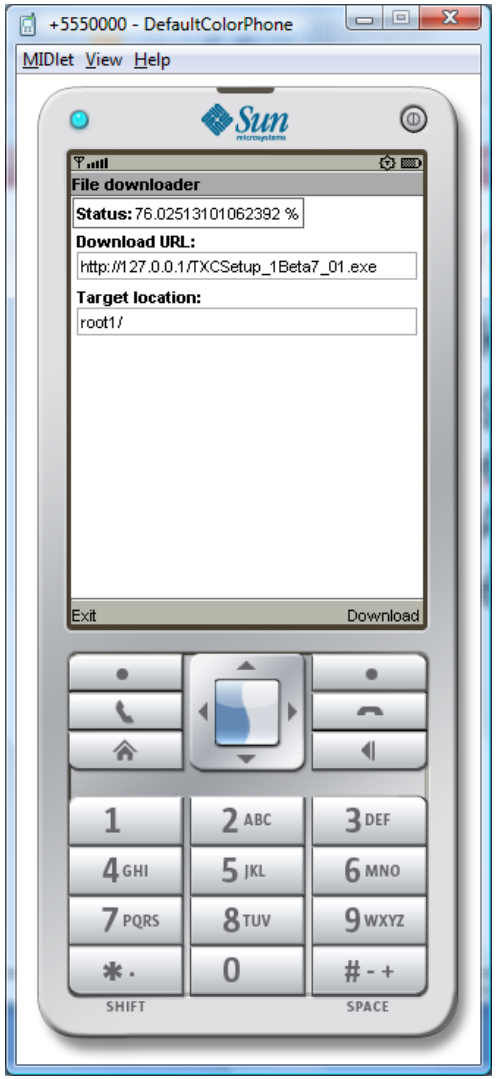
```
public void refreshCoords(final double aLatitude,final double aLongitude)
    {
        new Thread() {
            public void run()
            {
                try {
                    Image map = mapManager.retrieveStaticImage(
                      f.getWidth(), f.getHeight()-30,
                      aLatitude, aLongitude, 15, "jpeg", "your_api_key");
                    if (map != null)
                        imgItem.setImage(map);
                }
                catch (Exception e)
                {
                    f.append(e.getMessage());
                }
            }
        }.start();
    }
```

Fájlletöltő alkalmazás készítése

- Készítsünk egy alkalmazást, mely adott címről letölt egy fájlt a készülékre.
- Nehézségek:
  - Hova mentsük el a készüléken?
    - Nincs file/folder kiválasztó dialógus.
  - Hogyan valósítsuk meg a letöltést memóriatakarékosan?
    - Soha ne töltsük be a teljes fájlt a memóriába!
    - Használjuk a Runtime.getRuntime().freeMemory() függvényt, hogy megtudjuk mennyi memóriával gazdálkodhatunk épp

Fájlletöltő alkalmazás – a feladat

- Készítsünk egy alkalmazást, mely egy TextField-ben megadott fájlt letölt egy másik TextField-ben megadott könyvtárba.
- A letöltés közben jelezzük ki a letöltés állapotát!



A megoldás – MIDlet I.

```
private void initialize()
{
    formMain = new Form("File downloader");
    stringItemStatus = new StringItem("Status:", "-");
    formMain.append(stringItemStatus);
    downloadURLTextField = new TextField("Download URL:",
        "http://127.0.0.1/TXCSetup_1Beta7_01.exe",100,TextField.ANY);
    formMain.append(downloadURLTextField);
    downloadLocationTextField = new TextField("Target location:",
        "root1/",100,TextField.ANY);
    formMain.append(downloadLocationTextField);

    exitCommand = new Command("Exit", Command.EXIT, 0);
    downloadCommand = new Command("Download", Command.SCREEN, 0);
    formMain.addCommand(exitCommand);
    formMain.addCommand(downloadCommand);
    formMain.setCommandListener(this);
    Display.getDisplay(this).setCurrent(formMain);
    downloaderManager = new DownloaderManager(this);
}
```

A megoldás – MIDlet II.

```
public void commandAction(Command c, Displayable d)
{
    if (c == exitCommand)
    {
        destroyApp(true);
        notifyDestroyed();
    }
    else if (c == downloadCommand)
    {
        String fileName = downloadURLTextField.getString().substring(
            downloadURLTextField.getString().lastIndexOf('/')+1,
            downloadURLTextField.getString().length());

        String locationURL = downloadLocationTextField.getString() + fileName;

        downloaderManager.downloadFile(locationURL,
            downloadURLTextField.getString());
        stringItemStatus.setText("Download started...");
    }
}
```

A megoldás – interface az állapot kijelzésre

```
public interface DownloaderManagerListener
{
    public abstract void fileDownloadFinished(int aResultCode);

    public abstract void downloadStatusUpdate(String aDownloadStatus);
}
```

- MIDlet-ben:

```
public void fileDownloadFinished(int aResultCode) {
    if (aResultCode==1) // siker
            stringItemStatus.setText("Download finished!");
    else // kudarc
            stringItemStatus.setText("Download failed!");
}

public void downloadStatusUpdate(String aDownloadStatus) {
    stringItemStatus.setText(aDownloadStatus);
}
```

A megoldás – fájl letöltő függvény szállal

- A letöltést anonim szálban valósítjuk meg:

```
public void downloadFile(final String aToFilePath, final String
   aDownloadURL)
     {
         new Thread(){
             public void run()
             {
                 try {
//-- Fájl létrehozása
                     FileConnection conn = null;
                     int perIndex = aToFilePath.indexOf("/");
                     String tempPath =
   aToFilePath.substring(0,perIndex+1);
                     String remindPath =
   aToFilePath.substring(perIndex+1);
```

A megoldás – könyvtárszerkezet létrehozása

- A ‘/’ jelek mentén daraboljuk a könyvtárat és egyesével létrehozzuk a megfelelő mélységben:

```
while (perIndex>0)
{
    conn = (FileConnection) Connector.open( "file:///"+tempPath,
        Connector.READ_WRITE );

    if (!conn.exists())
        conn.mkdir();

    perIndex = remindPath.indexOf("/");
    tempPath += remindPath.substring(0,perIndex+1);
    remindPath = remindPath.substring(perIndex+1);
}
```

A megoldás – az üres fájl létrehozása

- Hozzuk létre az üres fájlt (ha esetleg már létezne, töröljük) és nyissunk egy OutputStream-et az íráshoz:

```
conn = (FileConnection) Connector.open(
  "file:///"+aToFilePath,
        Connector.READ_WRITE );
if (conn.exists())
{
    conn.delete();
}
conn.create();
OutputStream fileOS = conn.openOutputStream();
```

A megoldás – HTTP kapcsolat megnyitása

- Nyissuk meg a HTTP kapcsolatot, kérdezzük le a fájl méretét, majd nyissunk egy InputStream-et a beolvasáshoz (letöltéshez):

```
HttpConnection httpConn =
    (HttpConnection)
  Connector.open(aDownloadURL,Connector.READ);
long contentLength = httpConn.getLength();
InputStream httpIn = httpConn.openInputStream();
boolean saved = false;
long currentReaded = 0;
```

A megoldás – Fájl letöltése kis darabokban

```
while (!saved) { // amíg nem végeztünk
    int available = httpIn.available(); // ennyit olvashatunk be blokkolás nélkül; ha az available nem támogatott 8192 byte[] jó
    int freeMemory = (int) (Runtime.getRuntime().freeMemory() / 2); // ennyi mem. szabadon, ennek max csak a felét foglaljuk le
    byte[] readPiece;
    if (available < 1) { // ha letöltöttük a teljes fájlt (nincs már mit beolvasni)
        saved = true;
    }
    else {
        if (available < freeMemory) { // ne foglaljunk le nagyobb méretet, mint a freeMemory
            readPiece = new byte[available];
        }
        else {
            readPiece = new byte[freeMemory];
        }
        int readed = httpIn.read(readPiece); // visszatérési érték, hogy tulajdonképpen mennyit is sikerült beolvasni, -1 ha vége
        currentReaded += readed;
        fileOS.write(readPiece,0,readed); // tartalom elmentése 0-tól addig amennyit sikerült beolvasni
        fileOS.flush();
        downloaderManagerListener.downloadStatusUpdate(
            ((double)currentReaded/(double)contentLength)*100+" %"); // UI tájékoztatása az állapotról
    }
}
```

A megoldás – megnyitott kapcsolatok bezárása

```
            fileOS.close();
            httpIn.close();
            conn.close();
            httpConn.close();
        } catch (Exception ex) {
            ex.printStackTrace();
            downloaderManagerListener.fileDownloadFinished(0); // hiba
        }

        downloaderManagerListener.fileDownloadFinished(1); // siker
    }
  }.start();
}
```

Bluetooth server – a feladat

- Készítsen egy Bluetooth szervert bejövő kapcsolat fogadására!
- A kapcsolat fogadása után valósítson meg egy szálat, melyben megjeleníti a beérkezett üzeneteket!

A megoldás – szerver indítása

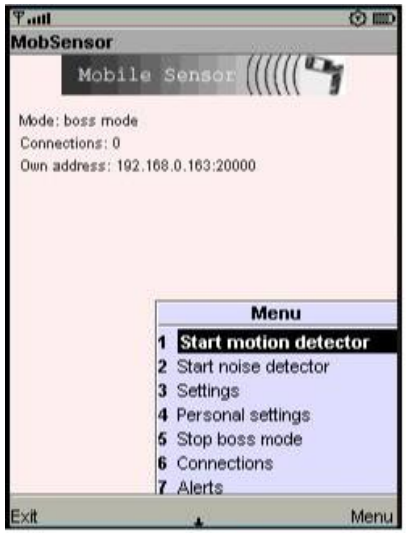
```
public void startServer()  {
   try {
      LocalDevice localDevice = LocalDevice.getLocalDevice();
      localDevice.setDiscoverable(DiscoveryAgent.GIAC);
      String serverUrl = "btspp://localhost:F0E0D0C0B0A000908070605040302011"; // egyedi azonosító
      serverUrl +=  ";name=ExampleService;authorize=false"; // szolgáltatásnév
      StreamConnectionNotifier notifier = (StreamConnectionNotifier)Connector.open(serverUrl);
   }
   catch (Exception e) {
      e.printStackTrace();
   }
   // kapcsolat fogadása
   try {
      // szünetel a szál amíg nem érkezik kapcsolat
      StreamConnection conn = notifier.acceptAndOpen();
      btDataInputStream = conn.openDataInputStream(); // a btDataInputStream egy DataInputStream
      new BluetoothReader().start(); // az osztály a következő dián van kifejtve
   }
   catch (Exception ex) {
      midlet.get_StringItemServerStatus().setText("Bluetooth Server Running Error:" + ex);
   }
}
```

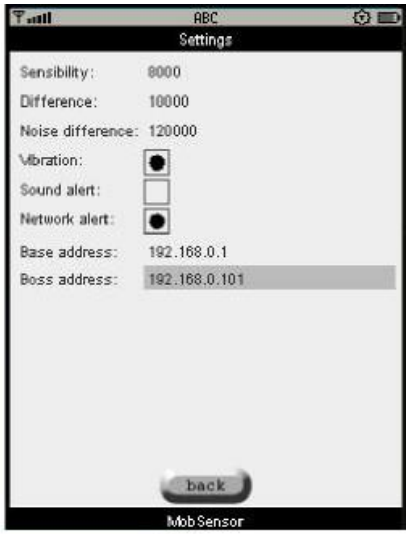
A megoldás – üzenetek olvasása

```
class BluetoothReader extends Thread {
    public void run() {
        while(readEnabled)
        {
            try
            {
                String msg = btDataInputStream.readUTF();
                midlet.appendMessage(msg);
                sleep(50);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

Demo - MobSensor

- Mozgás és hangérzékelés Java ME platformon
- Kamera és mikrofon elérhető
- Folyamatos mintavétel és különbségszámítás





Mp3 lejátszó – a feladat

- Készítsünk Mp3 lejátszó alkalmazást!
- Próbáljuk ki a VolumeControl vezérlő használatát!
- Használjuk a PlayerListener interface-t a lejátszás állapotának kijelzésére (start-stop-pause), illetve a player által nyújtott vezérlők (Control) lekérdezésére!

A megoldás – mp3 lejátszás

- Mp3 megnyitása és a lejátszás indítása:

```
private void playMP3() {
   try {
      // első módszer
      //mp3File = (FileConnection) Connector.open("file:///"+midlet.getTextFieldSelectedMusic().getString(),Connector.READ);
      //mp3FileInputStream = mp3File.openInputStream();
      //mp3Player = Manager.createPlayer(mp3FileInputStream,"audio/mp3");
      // --

      //-- második módszer
      mp3Player = Manager.createPlayer("file:///"+midlet.getTextFieldSelectedMusic().getString());
      //--
      if(mp3Player!=null) {
         mp3Player.addPlayerListener(this);
         mp3Player.realize();
         volumeControl=(VolumeControl)(mp3Player.getControl("VolumeControl"));
         volumeControl.setLevel(90);

         mp3Player.prefetch();
         mp3Player.start();
      }
   }
   catch(Exception e) { e.printStackTrace(); }
```

A megoldás – PlayerListener interface

```
public void playerUpdate(Player aPlayer, String aString, Object aObject) {
    if (aPlayer.equals(mp3Player)) { // vizsgáljuk, hogy a saját mp3Player eseménye érkezett-e
        if (aString.equals(STARTED)) {
            midlet.getStringItemMusicStatus().setText("started");
            // Elérhető Controllok megjelenítése
            Control[] controls = mp3Player.getControls(); // Control-ok lekérdezése
            midlet.getStringItemControls().setText("");
            for (int i=0; i<controls.length; i++) { // Control-ok listázása
                midlet.getStringItemControls().setText(
                    midlet.getStringItemControls().getText()+controls[i].toString()+"\n");
            }
        }
        else if (aString.equals(STOPPED)) {
            midlet.getStringItemMusicStatus().setText("stopped");
        }
        else if (aString.equals(END_OF_MEDIA)) { // média végére értünk
            midlet.getStringItemMusicStatus().setText("end of media");
        }
    }
}
```

Gyakorló példák

- Készítsünk egy teendőket karbantartó alkalmazást!
- Készítsünk egy példa WebService-t friss termékek listázására és hívjuk meg mobil oldalról a szolgáltatást!
- Készítsünk Java ME fájlböngészőt alapfunkciókkal (könyvtár létrehozása, szöveges fájl megnyitása, másolás, stb.)!
- Készítsünk egy Bluetooth alapú csevegő alkalmazást, mely alkalmas párhuzamos beszélgetések karbantartására!
- Készítsünk videó lejátszó alkalmazást! (nehéz)

Mobilszoftverek

J5 - Hálózatkezelés, Mobil3D és Multimédia

Java ME

Nokia Fejlesztői Klub

A Nokia Fejlesztői Klub rendezvényei

- Március 30., 17.00 óra
- QMF 14-15
  - Az új platformstratégia
  - Csilli-villi User Interface tervezés gyorsan
  - QML és QT Quick részletesen
  - Meglepetés!
- Regisztráció:
  - http://www.developer.mobil-academy.hu/

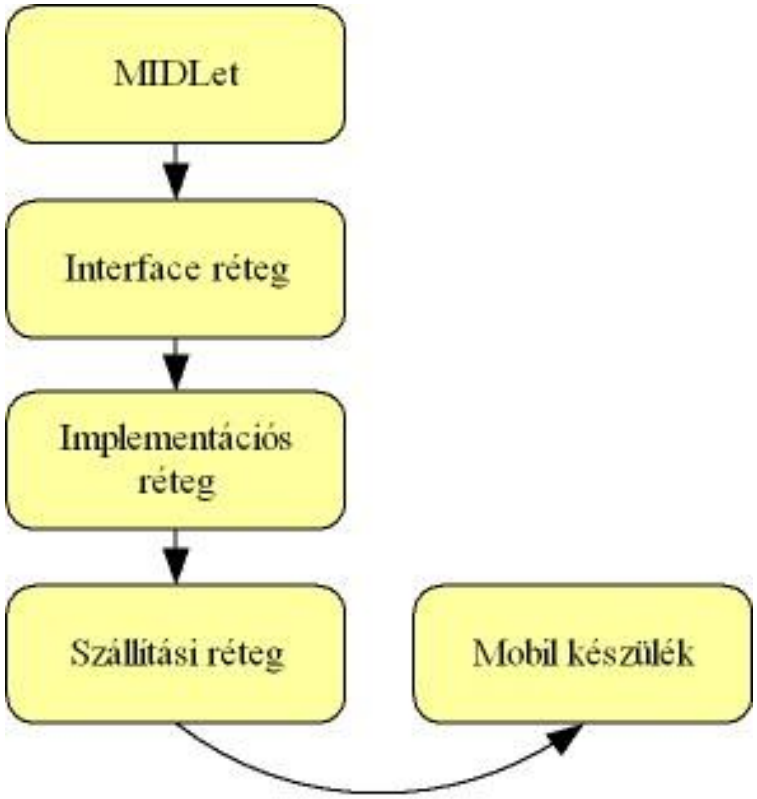
Tartalom

- Wireless Messaging API (SMS, MMS) – általános szolgáltatás
- Rövidtávú kommunikációs minták: Bluetooth
- Érdekesség: Mobile 3D
- Általános multimédia funkciók
  - Rezgés
  - Képernyő fényerőssége
  - Hang felvétele és lejátszása
  - Kamerakezelés

Wireless Messaging API (WMA) I.

- A WMA segítségével lehetővé válik GSM üzenetek küldése és fogadása, valamint GSM Cell Broadcast Service(CBS) üzenetek fogadása.
- A megfelelő osztályok a `javax.wireless.messaging` csomagban találhatók

Wireless Messaging API II.



3 rétegből áll:

- **Interface**: üzenet-protokoll független réteg, az üzenet küldés/fogadás egységes reprezentációja a feladata
- **Implementáció**: itt válnak külön az üzenetfajták, részekre bontást és összefűzést is végez
- **Szállítási**: a tényleges protokoll megvalósítás

Az interface réteg

- A Generic Messaging API foglalja össze az interface-ket, ami a `javax.wireless.messaging` csomagban található

Osztályok:

- `Message`: az üzenetet reprezentálja, 3 részből áll: cím, üzenet, flag-ek. Ősosztálya a `TextMessage` és a `BinaryMessage` osztályoknak, amik a szöveges illetve bináris (multimédia) üzeneteket jelképezik
- `MessageConnection`: Üzenetek fogadása és küldése a feladata. Többek között `Message` objektum létrehozására használhatók a függvényei
- `MessageListener`: Bejövő üzenetek detektálása

Az implementációs réteg

Short Message Service(SMS) API

- A `com.sun.midp.io.j2me.sms` csomag teszi lehetővé SMS üzenetek küldését és fogadását
- `MessageObject`: A `MessageObject` az SMS üzenet megtestesítője. Ez az osztály kezeli az üzenetpuffer létrehozását és az input/output műveletek végrehajtását. Itt is megjelenik a két alosztály : `TextObject` és `BinaryObject`
- `Protocol`: Ez köti a szállítási réteghez az SMS küldést. Ellenőrzi a paramétereket, kezeli a következő kivételeket: url-hiba, biztonsági ütközések, I/O ütközések

SMS - kapcsolat létrehozás

- A kapcsolat létrehozása a `Connector.open()` függvénnyel lehet, ahol a szükséges paraméter formátuma:
  - `"sms:// +18643630999:5000"` (port nem kötelező)
- SMS fogadása is támogatott:
  - `"sms://:5000"`
- Az SMS kapcsolatot egy `MessageConnection` objektum jelképezi

MessageConnection függvényei

- `public Message newMessage(loadType)`: előállítja az üzenetet, ahol a `loadType` `MessageConnection.TEXT_MESSAGE` vagy `MessageConnection.BINARY_MESSAGE` értékű lehet. A visszatérési érték ez alapján egy `TextMessage` vagy egy `BinaryMessage` objektum
- `public void send(Message)` : elküldi az adott üzenetet
- `public Message receive()` : üzenet fogadása
- `public setMessageListener(MessageListener ml)`: beállítja a MessageListener osztályt, amely kezeli az üzenet érkezését

TextMessage és BinaryMessage

- `TextMessage:`
  - `setPayloadText(text):` Beállítja a szöveges üzenet tartalmát
  - `String getPayloadText():` Kiolvassa a szöveges üzenet tartalmát
- `BinaryMessage:`
  - `setPayloadData(data):` Beállítja a bináris üzenet tartalmát
  - `byte[] getPayloadData():` Kiolvassa a bináris üzenet tartalmát

SMS küldése

```
private void sendSMS(String aText) {
  try
  {
    String url = "sms://+36202445695";
    MessageConnection conn =
      (MessageConnection)Connector.open(url);
    TextMessage msg = (TextMessage)conn.newMessage(
      MessageConnection.TEXT_MESSAGE);
    msg.setPayloadText(aText);
    conn.send(msg);
  } catch(Exception e) {
    e.printStackTrace();
  }
}
```

Bluetooth technológia

- Rádiófrekvenciás technológia, a 2.4 GHz-es Ipari-Tudományos-Orvosi tartományt használja
- Készülékek automatikus összekapcsolása (párosítás után, de azt elég egyszer)
- Számos új szolgáltatás is megvalósítható általa, mint például készülékek szinkronizálása
- Gyorsan és automatikusan működik

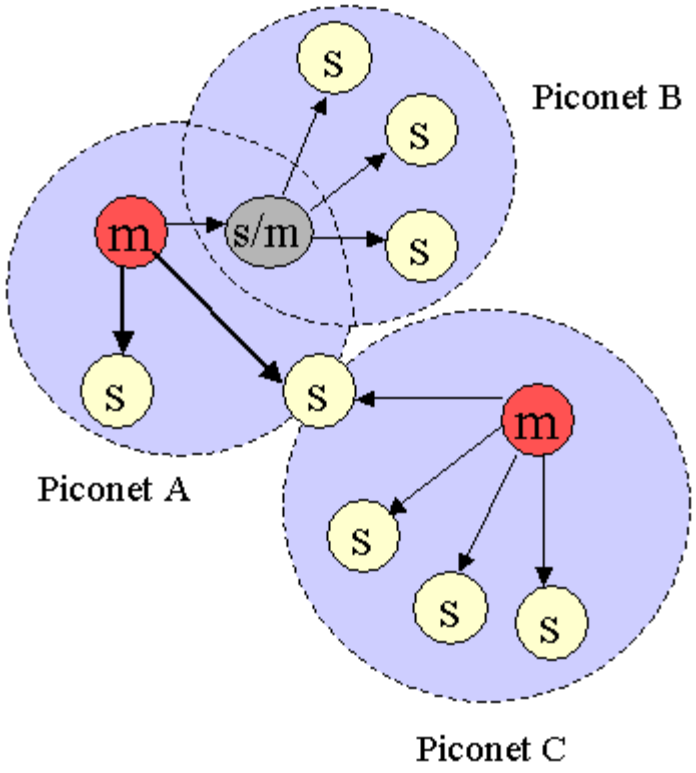
A Bluetooth tulajdonságai

- Vezeték nélküli és automatikus
- Olcsó (5 dolláros hardware)
- Adat és hang továbbítást is támogat (például headset)
- Többirányú jel, a falakon is képes áthatolni
- Frekvenciaugrást használ

Bluetooth hálózati topológia I.

- A készülékek úgynevezett piconet-be szerveződnek
- A piconet egy master-t és maximum 7 aktív slave-t tartalmaz
- Egy-egy és egy-több kapcsolatok
- A master kezdeményezi a kommunikációt
- Egy piconetben lévő készülék kommunikálhat egy másik piconet-ben lévővel: scatternet

Bluetooth hálózati topológia II.



Bluetooth energia módok

- A Bluetooth három energiamódot definiál az akkumlátor használat csökkentése érdekében:
  - Sniff mód: Alacsony energiaszinten való figyelés
  - Hold mód: Nincs adat továbbítás, az óra azonban működik
  - Park mód: A slave-k szinkronizálva vannak a master-hez, azonban nem részei a forgalomnak

Bluetooth alkalmazás anatómiája

- Stack inicializálás: Bluetooth egység felkészítése
- Készülék management: Bluetooth egység indítása
- Készülék felderítés: elérhető Bluetooth eszközök felderítése
- Szolgáltatás felderítés: az eszközök által nyújtott szolgáltatások lekérdezése
- Kommunikáció

Bluetooth technológia Java ME platformon

- A Bluetooth API-t a JSR 82-es kiegészítés tartalmazza
- `javax.bluetooth` és `javax.obex` csomagok
- A Bluetooth eszköz a `javax.bluetooth.LocalDevice` osztály statikus `getLocalDevice()` függvényével kérhető le
- Az API által biztosított szolgáltatások:
  - Szolgáltatások regisztrálása
  - Készülékek és szolgáltatások keresése
  - Kapcsolat létrehozása
  - Adatok küldése és fogadása
  - Kapcsolatok felügyelete
  - Biztonság szavatolása a fent felsorolt szolgáltatásokra
  - Bluetooth Control Center (BCC) – elkülöníti az alkalmazások csatorna-használatát

Készülékek keresése

- `DiscoveryAgent`: a keresés lebonyolítására használható osztály
  - A `public boolean startInquiry(int accessCode, DiscoveryListener listener)` függvénye indítja a keresést
  - A `public RemoteDevice[] retrieveDevices(int option)` függvénye pedig paramétertől függően a már ismert készülékeket vagy a legutóbbi keresés eredményét adja vissza
- `DiscoveryListener`: kereséskor ezt az interface-t kell implementálni, ha értesülni szeretnénk egy készülék megtalálásáról
  - Egy új készülék felderítése esetén a `public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)` függvény hívódik meg
  - Keresés befejezésekor pedig a `public void inquiryCompleted(int discoveryType)` függvény hívódik meg

Szolgáltatás keresése

- A `DiscoveryAgent` osztály `int searchServices(…)` függvényével lehet szolgáltatásokat keresni, visszatérési értéke a keresés tranzakciós azonosítója
- A `DiscoveryListener servicesDiscovered(…)` függvényét implementálva értesítést kapunk a talált szolgáltatásokról, paraméterül a keresés indításának tranzakciós azonosítóját természetesen megkapjuk
- A `serviceSearchCompleted()` függvény a keresés végét jelzi

Példa: listázzuk a látható Bluetooth készülékeket I.

```
public class BluetoothManager implements DiscoveryListener
{
    // variables
    private DiscoveryAgent discoveryAgent;
    private NearbyBluetoothDevicesMidlet midlet;
    private Vector foundDevices; // founded devices

    /** Creates a new instance of BluetoothManager */
    public BluetoothManager(NearbyBluetoothDevicesMidlet aMidlet)
    {
        midlet = aMidlet;
        foundDevices = new Vector();
    }
```

Példa: listázzuk a látható Bluetooth készülékeket II.

```
// start searching devices
 public void startSearchingDevices()
 {
        midlet.getDevicesList().setTitle("Searching devices...");
       foundDevices.removeAllElements();
       midlet.getDevicesList().deleteAll();
       try {
           LocalDevice localDevice = LocalDevice.getLocalDevice();
           discoveryAgent = localDevice.getDiscoveryAgent();
           discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
       } catch (Exception e) {
           midlet.getDevicesList().setTitle("Error during search");
       }
}
```

Példa: listázzuk a látható Bluetooth készülékeket III.

```
// a device is discovered
 public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)
 {
     foundDevices.addElement(btDevice);
     midlet.getDevicesList().setTitle("Found: "+foundDevices.size()+" searching...");
 }

public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
     throw new UnsupportedOperationException("Not supported yet.");
 }

 public void serviceSearchCompleted(int transID, int respCode) {
     throw new UnsupportedOperationException("Not supported yet.");
 }
```

Példa: listázzuk a látható Bluetooth készülékeket IV.

```
// finished searching devices
 public void inquiryCompleted(int i) {
    midlet.getDevicesList().setTitle("Reading devices("+foundDevices.size()+") data...");
    int ok = 0;
    for (int j=0; j<foundDevices.size(); j++) {
       RemoteDevice btDevice = (RemoteDevice)foundDevices.elementAt(j);
       try {
          String name = btDevice.getFriendlyName(true);
          midlet.getDevicesList().append(name,null);
          ok++;
       } catch (Exception ex) {
          midlet.getDevicesList().setTitle("Error during name retrieving");
       }
    }
    midlet.getDevicesList().setTitle("Completed:: "+ok);
}
}
```

Mobile 3D Graphics API

- JSR 184: első Java specifikus szabvány mobil 3D megjelenítésre
- Főbb cégek támogatásával készült (Sun Microsystems, Sony Ericsson, Symbian, Motorola, ARM, Cingular Wireless), specifikációvezető: Nokia
- Eredetileg alacsony számításkapacitást és memóriahasználatot feltételezve tervezték, de az API skálázható jobb készülékekhez, illetve akár beépített 3D gyorsító támogatásához

JSR 184 követelmények I.

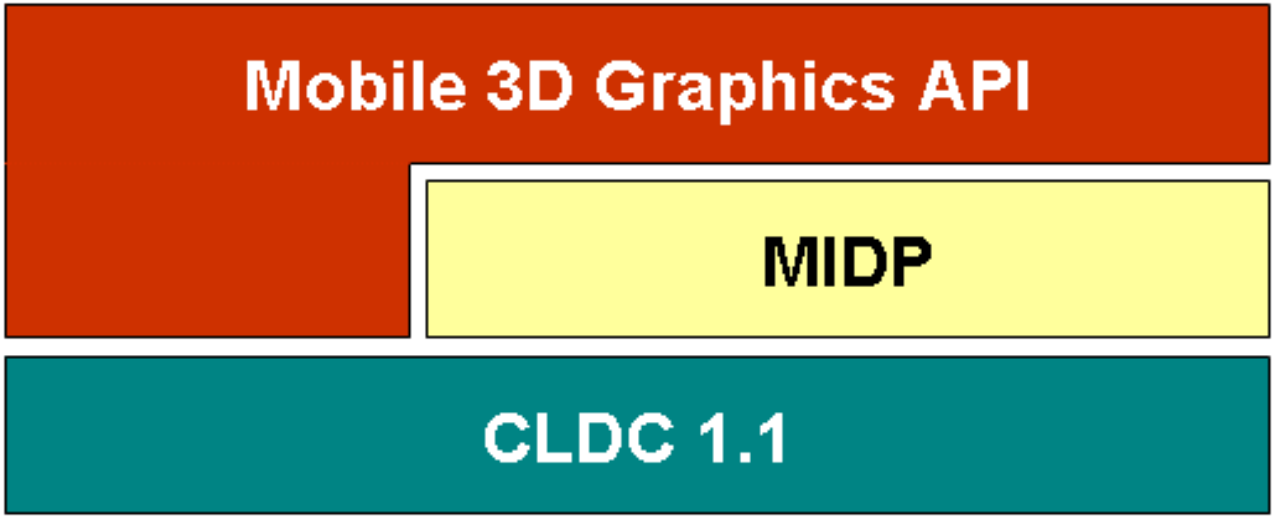
- Retain (előre elkészített m3g objektumok) mód és immediate (kód alapján rajzolás) mód támogatása, akár keverve is
- Programozás megkönnyítése céljából az API implementálja a fontos, általános osztályokat, és metódusokat
- Adatok bináris módban tárolása helytakarékosság céljából
- Hatékony megvalósítás akár hardware-s lebegőpontos támogatás nélkül is

JSR 184 követelmények II.

- A beépített Java float típus használata, új típus bevezetése nélkül
- Alacsony ROM és RAM követelmény, az API-nak már 150KB is elegendő kell, hogy legyen
- Legalább alapszintű Garba Collection támogatás
- Együttműködés más Java ME API-kal (pl MIDP)

Mobile 3D API software stack

- CLDC 1.1 szükséges a float támogatás miatt (akkor is, ha nincs hardware-s támogatás)
- Ügyeljünk a számításkapacitásra!



JSR 184 csomag

- javax.microedition.m3g
- Főbb osztályok:
  - Graphics3D: renderelés
  - Object3D: file-ból betölthető elemek kezelése
  - Camera: rálátás az implementált jelenetre
  - VertexArray: az objektum pontjait és azok tulajdonságait tárolja (pozíció, normális, szín, textúra koordináta)

JSR 184 renderelés fő lépései

- Graphics3D objektum elkészítése
- Graphics3D objektum hozzárendelése a 2D-s graphics objektumhoz
- Világ, objektumok és egyéb elemek renderelése
- Objektum felengedése (release), hogy a 3D-s képet a 2D-s bufferre renderelje a rendszer

m3g tartalom renderelése

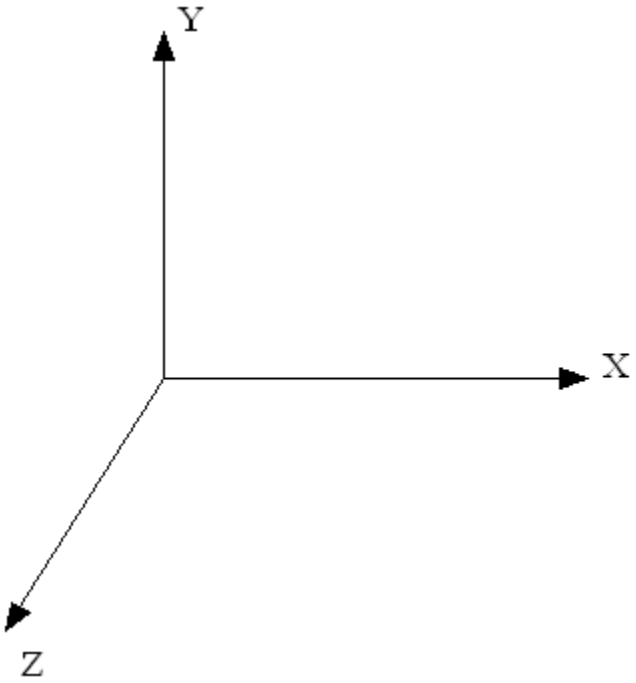
```
public class MyCanvas extends Canvas
    Graphics3D g3d;
    World world;
    int currentTime = 0;

    public MyCanvas() {
        g3d = Graphics3D.create();
        Object root[] = Loader.load("world.m3g");
        world = root[0];
    }

    protected void paint(Graphics g) {
        g3d.bindTarget(g);
        world.animate(currentTime);
        currentTime += 50;
        g3d.render(world);
        g3d.releaseTarget();
    }
}
```

Példa: háromszög körüli forgás

A koordinátarendszer



Tagváltozók és konstruktor

```
public class ShowTriangleCanvas extends Canvas{

    private ThreeDimensionTestMidlet midlet;

    /** A háromszög pontjainak koordinátái (x, y, z). */
    private static final byte[] VERTEX_POSITIONS = {
        -1, -1,  1,    1, -1,  1,   -1,  1,  1
    };

    /** Indexek, hogy melyik pont melyikkel legyen összekötve */
    private static int[] TRIANGLE_INDICES = {
        0, 1, 2, 0
    };

    /** Háromszög vertex adatok */
    private VertexBuffer _triangleVertexData;

    /** Háromszög vonalai */
    private TriangleStripArray _triangleTriangles;

    /** Graphics singleton a rendereléshez */
    private Graphics3D _graphics3d;

    private Camera camera;
    private float cameraX = 0.0f;
    private float cameraY = 0.0f;
    private float cameraZ = 10.0f;
    private float camRotDegreeY = 0.0f;
    private float cameraRadius = 10.0f;

    public ShowTriangleCanvas(ThreeDimensionTestMidlet
        aMidlet)
    {
        midlet = aMidlet;

        camera = new Camera();
    }
```

Renderelés

```
protected void init()
{
   // Singleton elkérése a 3D rendereléshez
   _graphics3d = Graphics3D.getInstance();

   // Vertex adatok létrehozása
   _triangleVertexData = new VertexBuffer();

   VertexArray vertexPositions =
       new VertexArray(VERTEX_POSITIONS.length/3, 3, 1);
   vertexPositions.set(0, VERTEX_POSITIONS.length/3,
  VERTEX_POSITIONS);
   _triangleVertexData.setPositions(vertexPositions, 1.0f, null);

   // Háromszögek létrehozása; az indexek VERTEX_POSITION-ra
  // hivatkoznak
   _triangleTriangles = new
  TriangleStripArray(TRIANGLE_INDICES,
       new int[] {TRIANGLE_INDICES.length});

   // Camera létrehozása perspekciós vetítéssel
   float aspect = (float) getWidth() / (float) getHeight();
   camera.setPerspective(30.0f, aspect, 1.0f, 1000.0f);
   Transform cameraTransform = new Transform();
   cameraTransform.postTranslate(cameraX,
  cameraY, cameraZ);

  cameraTransform.postRotate(camRotDegreeY,0,1,0
  );
   _graphics3d.setCamera(camera,
  cameraTransform);
}
```

Canvas paint()

```
protected void paint(Graphics graphics)
  {
     init();

     _graphics3d.bindTarget(graphics);
     _graphics3d.clear(null);
     _graphics3d.render(_triangleVertexData, _triangleTriangles,
        new Appearance(), null);
     _graphics3d.releaseTarget();
  }
```

Kamera forgatása

```
    protected void keyPressed(int keyCode)
    {
        if(keyCode==-3)
        {
            camRotDegreeY+=5.0f;
            cameraX = (float) (Math.sin(camRotDegreeY*Math.PI/180.0f) * cameraRadius);
            cameraZ = (float) (Math.cos(camRotDegreeY*Math.PI/180.0f) * cameraRadius);
            repaint();
        }
        else if(keyCode==-4)
        {
            camRotDegreeY-=5.0f;
            cameraX = (float) (Math.sin(camRotDegreeY*Math.PI/180.0f) * cameraRadius);
            cameraZ = (float) (Math.cos(camRotDegreeY*Math.PI/180.0f) * cameraRadius);
            repaint();
        }
    }
} // osztály bezáró
```

Telefonspecifikus figyelmeztetések

- Rezgés: tipikusan hívás/sms, de más alkalmazásokban is hasznos lehet:
  - Játék
  - Alkalmazás specifikus figyelmeztetés
- Háttérvilágítás: figyelemfelkeltés céljából, de szintén alkalmazhatjuk saját alkalmazásban:
  - Villogás hatás játékoknál
  - Felvillanás valamilyen esemény jelzésére

Villogás effekt

```
try
{
    Display.getDisplay(this).flashBacklight(50); // 50: felvillanás időtartama
    Thread.sleep(2500);
    Display.getDisplay(this). flashBacklight(50);
    Thread.sleep(2500);
    Display.getDisplay(this). flashBacklight(50);
    Thread.sleep(2500);
    Display.getDisplay(this). flashBacklight(50);
    Thread.sleep(2500);
}
catch (Exception e) // sleep miatt
{
    e.printStackTrace();
}
```

Rezgés megvalósítása

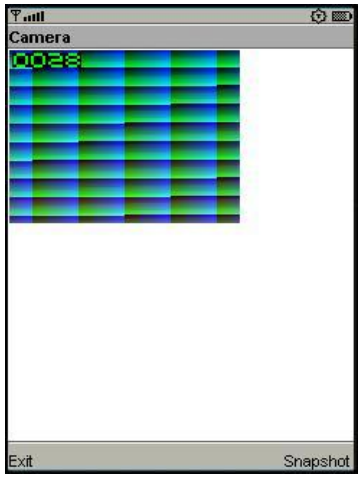
```
try
{
    Display.getDisplay(this). vibrate(500); // 500: rezgés időtartama
    Thread.sleep(1000);
    Display.getDisplay(this). vibrate(500);
    Thread.sleep(1000);
    Display.getDisplay(this). vibrate(500);
    Thread.sleep(1000);
}
catch (Exception e)
{
    e.printStackTrace();
}
```

Multimedia API – JSR 135

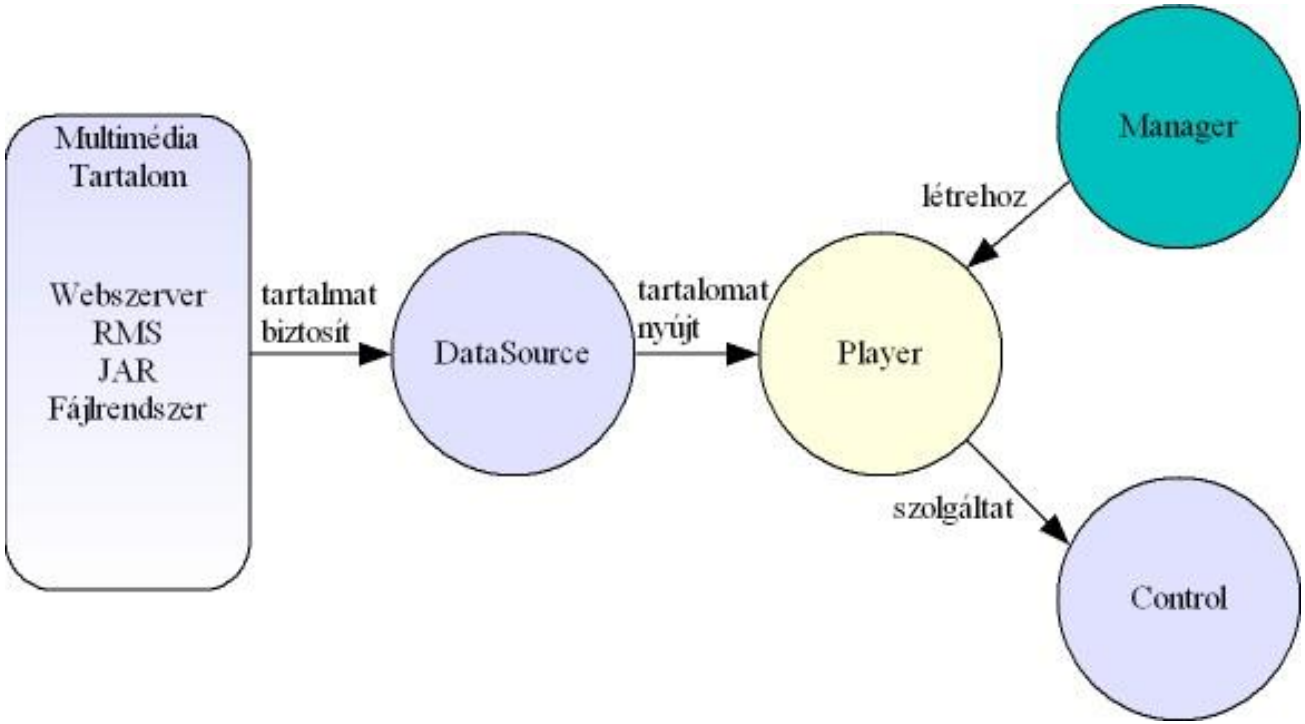
- Támogatja hangok generálását, felvételét és visszajátszását (szigorú memórialimittel dolgozik) - OutOfMemoryError
- Széleskörű protokoll és tartalom támogatás: az API nem szorítkozik semmilyen protokoll, vagy formátum megkötésre
- Bővíthetőség

JSR 135 csomag

- `javax.microedition.media`
- Cél: erőforrás takarékos módon a multimédia funkciók elérése J2ME alkalmazásból
- Az emulátor a kamerát is képes emulálni



Multimédia feldolgozása



Főbb osztályok

- **Manager**: legfelsőbb szintű factory jellegű vezérlőosztály
  - Feladata a Player objektumok létrehozása, információt biztosít az rendszer képességeiről (formátum, protokoll)
  - Egyszerű felületet nyújt tone-ok (hangok) lejátszására
  - Content-type alapján történik a formátum azonosítása
- **Player**: a multimédia tartalmat lejátszó interface
  - Lekérdezhető, hogy mely Control-okkal rendelkezik az adott Player
- **Control**: a Player beállításait kezelő interface
  - Hangerő: VolumeControl

Egyszerű hangok lejátszása

- Manager osztály statikus függvénye:
  - `static void playTone(int note, int duration, int volume)`
- Példa:

```
try {
  // hang lejátszása 4000 ms hosszan 100-as hangerővel
  Manager.playTone(ToneControl.C4, 4000, 100);
}
catch(MediaException me) { }
```

- ToneControl vezérlő:
  - Hang szekvenciák lejátszása
  - Az osztály tartalmazza a szükséges konstansokat

Manager osztály I.

- `javax.microedition.media.Manager`
- Player létrehozása Stream, vagy hely megadásával:
  - `static Player createPlayer(java.io.InputStream stream, String type)`
  - `static Player createPlayer(String locator)`

Manager osztály II.

- A Manager osztályban található a createPlayer() függvényt, amellyel a player objektumok létrehozhatók:

```
Player player = Manager.createPlayer(String locator);
```

- locator: a protokoll és a tartalom meghatározására használható a következő formátumban: <protocol>:<content location>
- A visszakapott Player objektum függvényeivel vezérelhető az adott média

Manager osztály III.

- Szintetizált hang lejátszása:
  - `static void playTone(int note, int duration, int volume)`
- Kezelt formátumok és protokollok lekérdezése:
  - `static String[] getSupportedContentTypes(String protocol)`
  - `static String[] getSupportedProtocols(String content_type)`

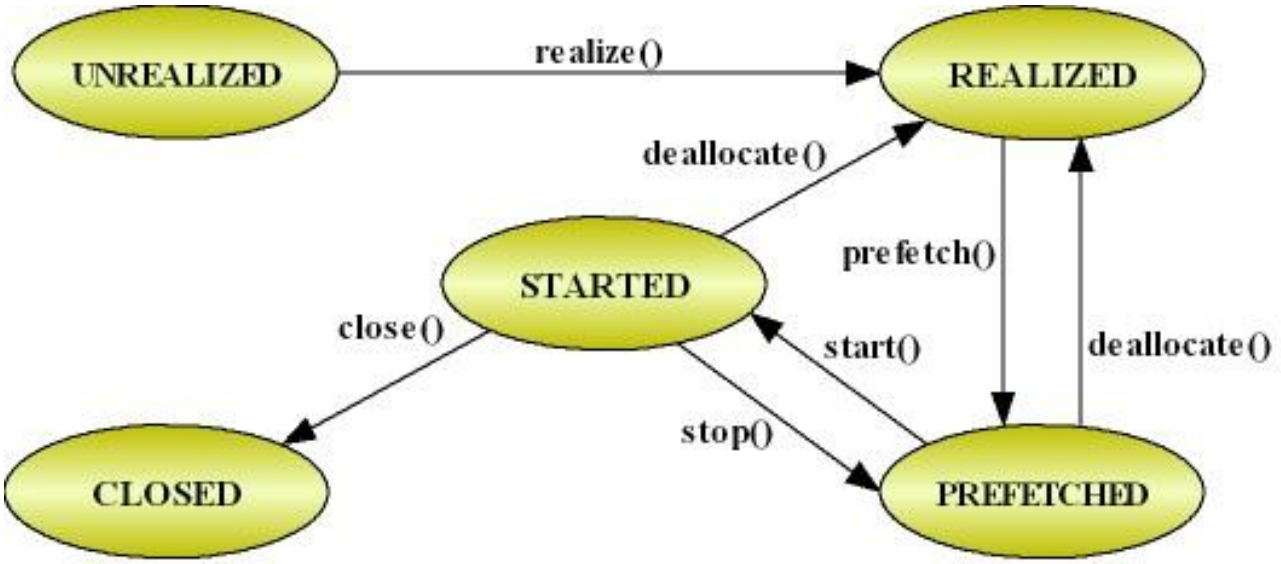
Multimédia információk lekérdezése - példa

```
public void writeInfos()
{
    mainForm.append("Midi device locator:
  "+Manager.MIDI_DEVICE_LOCATOR+"\n");
    mainForm.append("Tone device locator:
  "+Manager.TONE_DEVICE_LOCATOR+"\n");

    // Kezelt tartalmak és protokollok lekérdezése
    String[] tempContentTypes=
       Manager.getSupportedContentTypes(null);
    for (int i=0; i<tempContentTypes.length; i++)
    {
        String[] tempProtocols=
  Manager.getSupportedProtocols(tempContentTypes[i]);
        mainForm.append("PROTOCOLS to: "+tempContentTypes[i]+"\n");
        for (int j=0; i<tempProtocols.length; i++)
        {
            mainForm.append(tempProtocols[j].toString()+"\n");
        }
    }
}
```

Player életciklus

- `javax.microedition.media.Player`



Megjegyzés: A close() meghívása minden állapotból a CLOSED állapotba vezet

Player állapotok

- **Unrealized**: kezdeti állapot
- **Realized**: a média lejátszásához minden információa rendelkezésünkre áll (időigényes lehet az állapotba lépés!)
- **Prefetched**: a média lejátszásának indításához minden elővan készítve
- **Started**: a média lejátszás alatt van
- **Closed**: az összes erőforrást felszabadítja a rendszer

Player állapotváltás

- Amikor a player létrejött, UNREALIZED állapotba kerül
- A **realize()** függvényt meghívva kerül REALIZED állapotba és inicializálásra kerülnek azok az információk, amelyek a player-nek szükségesek az adott média eléréséhez
- A **prefetch()** függvényt meghívva a player a PREFETCHED állapotba kerül, létrehozza a hálózati kapcsolatokat, valamint további inicializálásokat hajt végre. Meghívásával a player tényleges indításának ideje minimalizálható
- A **start()** metódus segítségével a player a STARTED állapotba kerül, ahol a tényleges média feldolgozására kerül sor
- A PREFETCHED, vagy STARTED állapotban a **deallocate**() függvény hívásakor a rendszer az erőforrásokat felszabadítja és a player újra REALIZED állapotba kerül
- STARTED állapotban a **stop**() függvény hívásakor a player ismét visszakerül a start előtti PREFETCHED állapotba
- Amikor a feldolgozás befejeződött, a player automatikusan visszakerül a PREFETCHED állapotba
- A **close()** függvényt meghívva kerül a player CLOSED állapotba

Player létrehozása – példák I.

- Audio:
  - `capture://audio`: hang beolvasása az alapértelmezett beolvasó eszközről
  - `capture://devmic0?encoding=pcm`: hang beolvasása a devmic0 eszközről
- Video:
  - `capture://video`: videó, vagy kép beolvasása a készülék alapértelmezett videó eszközéről
  - `capture://devcam0?encoding=rgb888&width=100&height=50`: videó, vagy kép beolvasása a devcam0 készülékről (például másodlagos kamera), rgb888-as felbontással, megadott szélességben és magasságban

Player létrehozása – példák II.

- Rádió:
  - `capture://radio?f=105.1&st=stereo`: rádió az 105.1-es FM frekvencián
- Streaming:
  - `rtp://host:port/type`: streaming, ahol a típus „video”, „audio”, vagy „text” lehet
- Multimédia adat a Web-en:
  - `http://webserver/music.wav`: Interneten elérhető hangfájl lejátszása

Player függvények I.

- Player eseménykezelő hozzárendelése és eltávolítása:
  - void addPlayerListener(PlayerListener playerListener)
  - void removePlayerListener(PlayerListener playerListener)
- A média lejátszás ismétléseinek száma (-1 paraméter: amíg le nem állítjuk):
  - void setLoopCount(int count)

Player függvények II.

- A média hossza mikroszekundumban, a lejátszás aktuális idejének lekérdezése, valamint beállítása:
  - long getDuration()
  - long getMediaTime()
  - long setMediaTime(long aLength)

PlayerListener interface

- `javax.microedition.media.PlayerListener`
- `public void playerUpdate(Player player, String event, Object eventData)`
  - player: A lejátszó, amely az eseményt generálta
  - event: A generálódott esemény String típusú azonosítója (STARTED, STOPPED, END_OF_MEDIA, DURATION_UPDATED, DEVICE_UNAVAILABLE, DEVICE_AVAILABLE, VOLUME_CHANGED, ERROR, CLOSED)
  - eventData: Az eseményhez kapcsolódó egyéb adatok

EventData adatok

- Mikor indult a lejátszás
- Mikor állt le a lejátszás
- Mikor fejeződött be a lejátszás
- Mekkorára módosult a lejátszás időtartama
- VolumeControl objektum, melytől lekérdezhetők az adatok
- A keletkezett hiba leírása

Multimédia vezérlők

- A vezérlők a `player` objektumoktól kérhetők el, használhatók a multimédia eszközök irányítására:
  - CameraControl
  - VideoControl
  - ZoomControl
  - FocusControl
  - SnapshotControl
  - VolumeControl
- Például:
  - ```
    VideoControl videoControl =
    (VideoControl)(player.getControl("VideoControl"));
    ```
    ```

Alapértelmezett kamera eszköz létrehozása

- Player létrehozása a `capture://video` paraméterrel:
  - `Player player= Manager.createPlayer("capture://video");`
- VideoControl létrehozása:
  - `VideoControl videoControl= (VideoControl)(player.getControl("VideoControl"));`

VideoControl megjelenítése

- `Object initDisplayMode(int mode, java.lang.Object arg)`
- `int mode`:
  - USE_DIRECT_VIDEO: ha például Canvas-on szeretnénk megjeleníteni a kamera képét
  - USE_GUI_PRIMITIVE: ha például egy GUI elemként szeretnénk hozzáfűzni egy Form-hoz
- `Object arg`:
  - A Canvas objektum, ha a USE_DIRECT_VIDEO módot választottuk

VideoControl elhelyezése Form-on

```
Player player=
  Manager.createPlayer(
  "capture://video");

player.realize();

VideoControl videoControl=
  (VideoControl)(player.getControl("Vi
  deoControl"));

if (videoControl != null){
  mainForm.append((Item)(videoControl.
  initDisplayMode(
  VideoControl.USE_GUI_PRIMITIVE,null)
  ));

}
```

VideoControl elhelyezése Canvas-on I.

```
public class MyCameraCanvas extends Canvas {

    private HelloMidlet midlet;

    /** Creates a new instance of MyCameraCanvas */
    public MyCameraCanvas(HelloMidlet aMidlet) {
        midlet = aMidlet;
    }

    protected void paint(Graphics graphics) {
        graphics.setColor(0,0,0);
        graphics.fillRect(0,0,getWidth(),getHeight());
    }

    protected void keyPressed(int keyCode) {
        midlet.getDisplay().setCurrent(midlet.get_mainForm());
    }
}
```

VideoControl elhelyezése Canvas-on II.

```
public void putCameratoCanvas(){
    try{
        player=Manager.createPlayer("capture://video");
        player.realize();
        videoControl=(VideoControl)(player.getControl("VideoControl"));
        if (videoControl != null){
            videoControl.initDisplayMode(VideoControl.USE_DIRECT_VIDEO,canvas);
            videoControl.setDisplaySize(100,220);
            videoControl.setDisplayLocation(30,40);
        }
    }
    catch(MediaException me){ me.printStackTrace(); }
    catch (IOException ex){ ex.printStackTrace(); }
    try{
        if(player!=null)
            player.start();
        if(videoControl!=null)
            videoControl.setVisible(true);
    }
    catch(MediaException me){ me.printStackTrace(); }
    catch(SecurityException se){ se.printStackTrace(); }
}
```

VideoControl elhelyezése Canvas-on II.

Gyakorló feladatok

- Készítsünk egy saját SMS (MMS) küldő alkalmazást
- Készítsünk egy egyszerű Bluetooth-os csevegő alkalmazást
- Rajzoljunk ki egy 3D-s kocka objektumot
- Valósítsuk meg egy kameráról beolvasott kép átküldését Bluetoothon (nehéz)

AMORG APPLIED MOBILE RESEARCH GROUP

Mobilszoftverek

iPhone

Blázovics László
Blazovics.laszlo@aut.bme.hu

Tartalom

- Történelem
- Lehetőségek és korlátok
- A fejlesztőkörnyezet
- Az Objective-C

Történelem

iPhone

iPhone 3G

iPhone 3GS

iPhone 4

Történelem

iPad

iPad 2

Fontos tulajdonságok

- Egyszerre csak egy third-party program futhat
  - Programváltáskor, híváskor állapotmentés
  - Sok kritika
  - iPhone OS 4.0: Multitasking*
- A programok “Sandbox”-ba kerülnek
- Nincs “swap” a memóriához
  - Low Memory Warning
  - Program kényszerített kiléptetése

A programozó lehetőségei

- Nincs Java
- Nincs Flash*
- Vannak
  - Webes alkalmazások
  - Hibrid alkalmazások
  - **Natív alkalmazások**

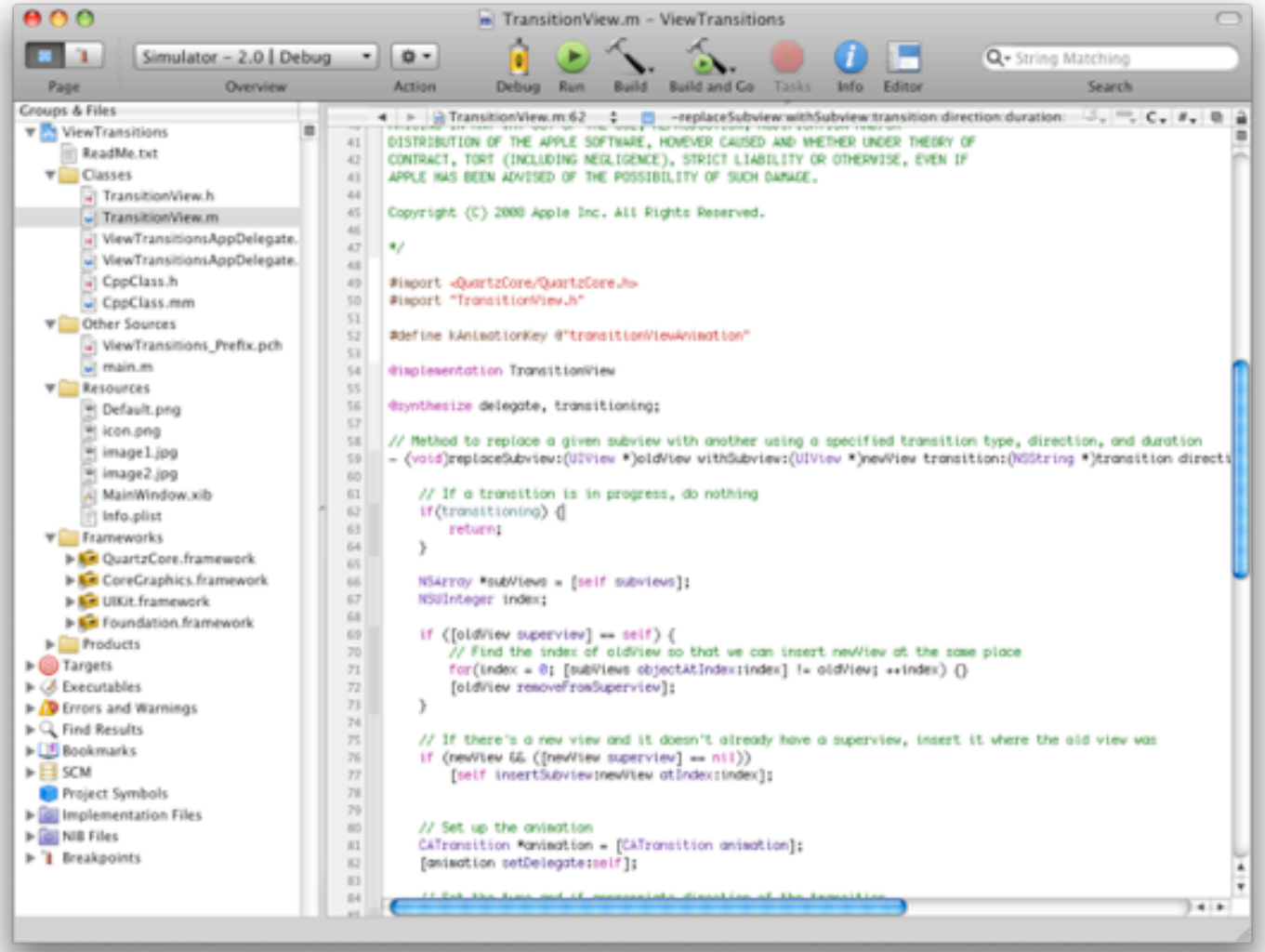
A fejlesztőkörnyezet

- iPhone Developer Program
  - AppStore
    - iTunes-on keresztül elérhető egységes eladási felület
    - A fejlesztő szabja meg az árat
    - 70% bevétel a fejlesztőé, havonta fizetve
    - Az egyes programok AppStore-ba feltöltése ingyenes
    - Az Apple visszadobhatja az alkalmazást
      - privát API-k használata, kifogásolt funkcionalitás, de nem kötelesek indokoni
  - Ad-Hoc megosztás
    - 100 eszközre tesztelés céljából

A fejlesztőkörnyezet

- A fejlesztés költségei
  - Számítógép
    - Intel processzoros Macintosh
      - vagy: OSX86
  - A fejlesztőkörnyezet és az iPhone SDK ingyen jár a Mac OS X-hez*
    - (az iPhone SDK-hoz regisztrálni kell az ADC-re, ingyen)
    - Az Xcode 4-hez ADC Membership kell vagy 5$.
  - iPhone, iPod Touch és / vagy iPad készülékek
  - iPhone Developer Program – 100$

Xcode


TransitionView.m – ViewTransitions
Simulator – 2.0 | Debug
Page
Overview
Action
Debug
Run
Build
Build and Go
Tasks
Info
Editor
Search
Groups & Files
ViewTransitions
ReadMe.txt
Classes
TransitionView.h
TransitionView.m
ViewTransitionsAppDelegate.
ViewTransitionsAppDelegate.
CppClass.h
CppClass.mm
Other Sources
ViewTransitions_Prefix.pch
main.m
Resources
Default.png
icon.png
image1.jpg
image2.jpg
MainWindow.xib
Info.plist
Frameworks
QuartzCore.framework
CoreGraphics.framework
UIKit.framework
Foundation.framework
Products
Targets
Executables
Errors and Warnings
Find Results
Bookmarks
SCM
Project Symbols
Implementation Files
NIB Files
Breakpoints
DISTRIBUTION OF THE APPLE SOFTWARE, HOWEVER CAUSED AND WHETHER UNDER THEORY OF
CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF
APPLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Copyright (C) 2008 Apple Inc. All Rights Reserved.
#import <QuartzCore/QuartzCore.h>
#import "TransitionView.h"
#define kAnimationKey @"transitionViewAnimation"
@implementation TransitionView
@synthesize delegate, transitioning;
// Method to replace a given subview with another using a specified transition type, direction, and duration
// If a transition is in progress, do nothing
if(transitioning) {
return;
}
NSArray *subViews = [self subviews];
NSUInteger index;
if ([oldView superview] == self) {
// Find the index of oldView so that we can insert newView at the same place
for(index = 0; [subViews objectAtIndex:index] != oldView; ++index) {}
[oldView removeFromSuperview];
}
// If there's a new view and it doesn't already have a superview, insert it where the old view was
if (newView && ([newView superview] == nil))
[self insertSubview:newView atIndex:index];
// Set up the animation
CATransition *animation = [CATransition animation];
[animation setDelegate:self];

Xcode 

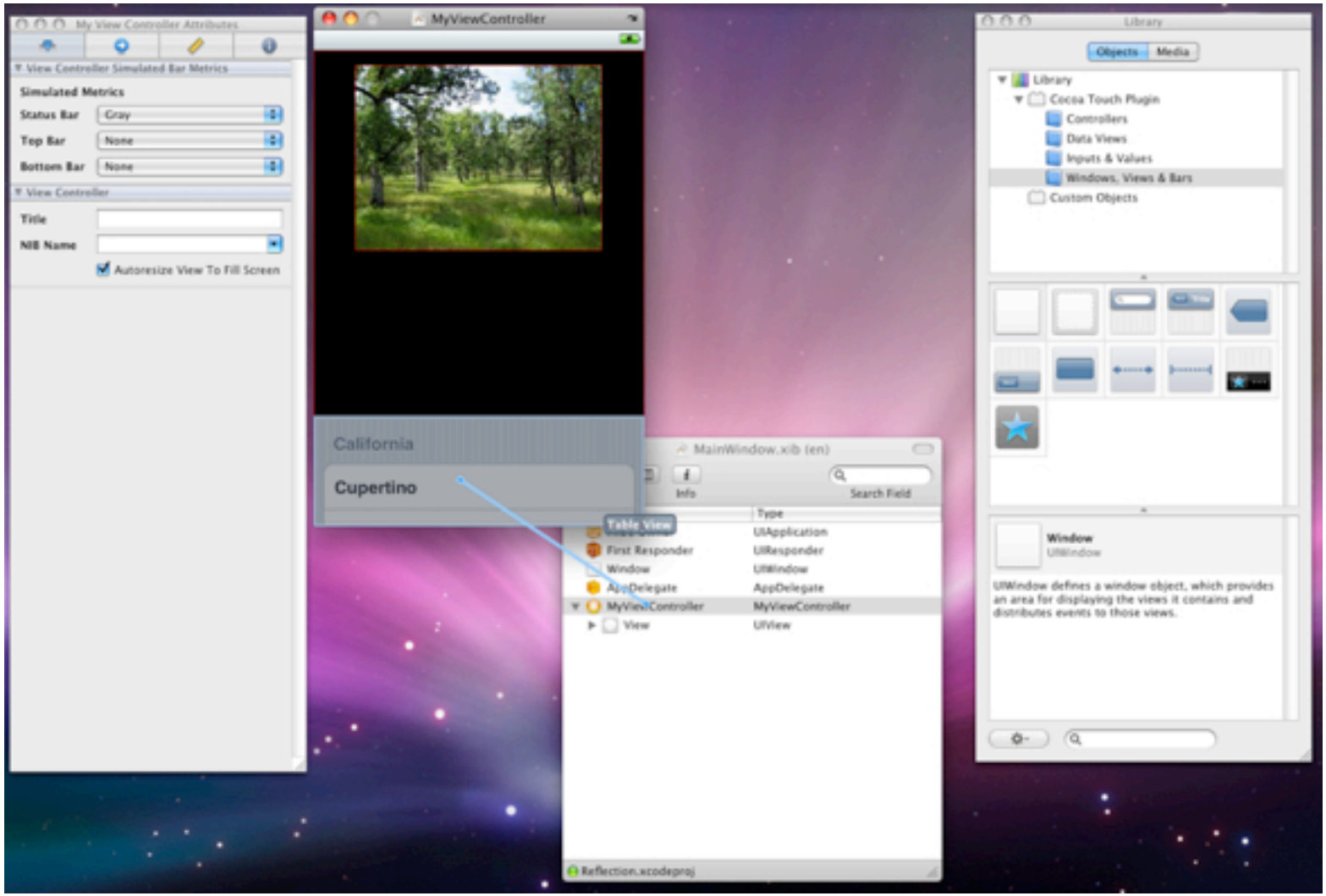
- IDE
- Jól kereshető dokumentáció
- GCC, GDB (grafikus felület hozzá)
- Version Control (CVS, Subversion)
- 3.2 óta beépített kódelemzés (CLang)

Xcode

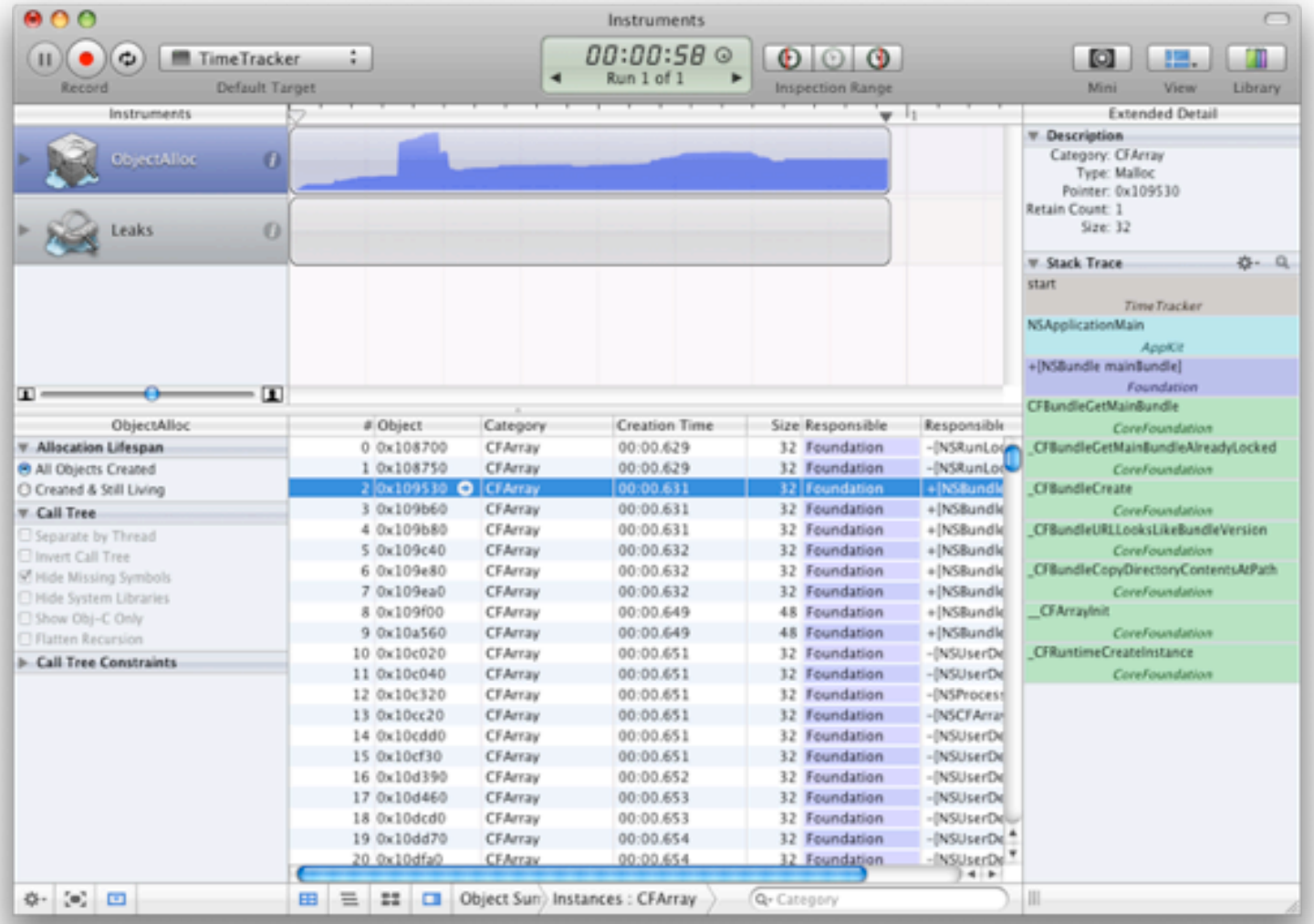
- Kódelemzés

```
174 - (void)receiveNotification:(NSNotification*)info {
175     if ([[info name] isEqualToString:DataControllerDidFinishFetchingChannelListNotification]) {
176     if (![DataController dataController].purgingOldItems) {
177         shouldDisplayWaitForUpdateView = NO;
178         if (waitForUpdateView) {
179             [waitForUpdateView removeFromSuperview];
180             [waitForUpdateView release];
181             waitForUpdateView = nil;
182         }
183     } else {
184         waitForUpdateView.text = @"Régi hírek törlése";
185     }
186     [[NSNotificationCenter defaultCenter] removeObserver:self
187                                                     name:DataControllerDidFinishFetchingChannelListNotification
188                                                   object:nil];
189
190     NSArray* array = [[NSArray alloc] initWithObjects:@"1",@"2",nil];   Method returns an Objective-C object with a +1 retain count (owning referer
191 }
192
193 if ([[info name] isEqualToString:DataControllerDidFinishPurgingOldItems]) {
194     if (![DataController dataController].updatingChannelList) {   Object allocated on line 190 is no longer referenced after this point and has a retain coun
```

Interface Builder



Instruments



iPhone Simulator

- Nem hardware emulátor!
  - a Mac OS X framework-jei futnak alatta
  - csak az iPhone API-jait szimulálja
  - feltételes fordítás
- Jó kezdés a teszteléshez, optimalizáláshoz, de nem helyettesíti az eszközön végzett tesztelést.

Objective-C

- Jellemzők
  - Objektumorientált
  - Az ANSI-C nyelvet teljesen tartalmazza
    - A hardverközelibb modulokat C-ben írtják (UNIX alapok)
  - C++-val könnyen összehozható
    - “Objective-C++”, .mm fájl XCode-ban
  - De mégis más egy kicsit

Objective-C

- Objektumorientált nyelv
  - Objektum
    - az adatok és a hozzájuk tartozó funkcionalitás összefogása
    - az objektumok felépítését, tervét osztályok definiálják
- Osztályok az Objective-C-ben
  - Konvenció szerint [Osztálynév].h és [Osztálynév].m fájlokba struktúrálva
  - .h header file az osztályt deklaráló interfésznek
  - .m implementációs file az üzenetek (metódusok) implementációinak

Objective-C: Osztályok

Interfész Objective-C-ben

- #import
  - egyszer tölti csak be az adott header-t
- @... direktívák
  - @interface ...
  - @class
  - @synthesize
- tagváltozók
  - nincs osztály változó
- metódusok
  - ezeknek nem szabályozható a láthatósága
  - + : osztálymetódus
  - - : példánymetódus
- property-k

```
#import <UIKit/UIKit.h>

@class MyViewController;

@interface AppDelegate : NSObject {
    BOOL windowDisplayed;
    IBOutlet UIWindow *window;
    IBOutlet MyViewController *viewController;
}
+ (AppDelegate*)appDelegate;

@property (nonatomic, retain) UIWindow *window;
@property (nonatomic, retain) MyViewController
*viewController;

- (void)applicationDidFinishLaunching:(UIApplication *)
application;

@end
```

Objective-C: Osztályok

- Interfész Objective-C-ben és C++-ban

```
#import <UIKit/UIKit.h>

@class MyViewController;

@interface AppDelegate : NSObject {

    BOOL windowDisplayed;
    UIWindow *window;
    MyViewController *viewController;
}

+ (AppDelegate*)appDelegate;
- (void)applicationDidFinishLaunching:
  (UIApplication *)application;

@end
```

```
class MyViewController;

class AppDelegate : public Object {
public:
    bool windowDisplayed;
    UIWindow* window;
    MyViewController* viewController;

public:

    static AppDelegate* appDelegate();

    void applicationDidFinishLaunching
(UIApplication* application);

};
```

Objective-C: Osztályok

- Property-k
  - Objective-C 2.0 óta
  - dot Syntax
  - getter/setter metódusok
    - `readonly / readwrite (ez a default)`
    - `assign / retain / copy`
      - `assign:` Egyszerű értékadás (ez a default)
      - `retain:` Átadáskor az alany retain üzenetet kap
      - `copy:` NSCopying protocol-t megvalósító objektumok másolódnak a copy üzenethívással
    - `nonatomic / atomic` (nincs ilyen de ez a default)
      - Teljesítmény vs. threadsafe
  - Implementáció részben @synthesize direktívával le kell generáltatni!

Objective-C: Osztályok

- Property-k
  - Objective-C 2.0 óta
  - dot Syntax
  - getter/setter metódusok

```
@interface MyLabel : UILabel{
...
}
@property (nonatomic, readwrite) UIColor *bacgroundColor;
...
@end

Mylabel* label;
//Objective-c standard syntax
[label setBackgroundColor:[UIColor whiteColor]];

//dot syntax Csak property-kre!
label.backgroundColor = [UIColor whiteColor];
```

Objective-C: Osztályok

- Van már definíciónk, sőt automatikusan generálhatunk metódusokat, de azért csak kell kódot is írni nem?
- Implementációs fájl
  - Az osztály számára deklarált üzenetek implementációit gyűjti
    - történelmileg innen a ‘.m’ kiterjeszés - messages

Objective-C: Osztályok

- Implementáció
  - metódus implementációk az @implementation és @end direktívák között

```
#import "AppDelegate.h"
#import "MyViewController.h"

@implementation AppDelegate

@synthesize window, viewController;

- (void)applicationDidFinishLaunching:(UIApplication *)application{
    [window addSubview:viewController.view];
    [window makeKeyAndVisible];
}

@end
```

Objective-C: Osztályok

- Implementáció Objective-C-ben és C++-ban

```
#import "AppDelegate.h"
#import "MyViewController.h"

@implementation AppDelegate

@synthesize window, viewController;

- (void)applicationDidFinishLaunching:
  (UIApplication *)application
{
    [window addSubview:viewController.view];
    [window makeKeyAndVisible];
}

@end
```

```
void AppDelegate::applicationDidFinishLaunching:
(UIApplication * application)
{
    window->addSubview(viewController->view);
    window->makeKeyAndVisible();
}
```

Objective-C: Osztályok

- Hogyan érhetjük el egy osztály tagváltozóit és tagfüggvényeit saját metódusainak törzséből?
- A tagváltozókat szabadon használhatjuk, mint a C++-ban
- A tagmetódusokat és property-ket a self kulcsszón keresztül érhetjük el (self = this a C++-ban)

Objective-C: Osztályok

- Hogyan érhetjük el egy osztály tagváltozóit és tagfüggvényeit saját metódusainak törzséből?

```
@implementation RootViewController
...
- (void)increment {
    member++;
}
...
- (void)viewDidLoad {
    [super viewDidLoad];
    member = 2;
    [self increment];
}
@end
```

Objective-C: Objektumok

- Most már tudjuk, hogyan definiálhatunk osztályokat és ezek működését
- Hogyan lesznek azonban belőlük használható példányok, objektumok futás közben?

Objective-C: Objektumok

- Példányosítás
  - 2 lépcsős
    - memória lefoglalása

```
+ (id)alloc;
```

    - objektum inicializálása

```
- (id)init...;
- (id)initWithArray:(NSArray *)array copyItems:(BOOL)flag
```

  - factory metódusok

```
+ (id)arrayWithObject:(id)anObject;
```

  - egy lépésbe is összevonható, ha nem akarunk paramétereket adni az init metódusnak

```
id array = [NSMutableArray new];
//VS.
id array = [[NSMutableArray alloc] init];
```

Objective-C: Protocol-ok

- Protocol-ok (máshol: interface)
  - Metódus lista
  - Direktívák
    - `@optional`
    - `@required (ez a default)`
  - Protocol adoptációja
    - `@interface AppController : NSObject <TCPServerDelegate, ...>`
  - Egymásba ágyazhatók (maga a protocol is adoptál egy másik protocolt)

```
@protocol processedInfo

- (BOOL)processed;
- (float)processedRate;

@end
```

Objective-C: Objektumok

- Objective-C 2.0 garbage collection
  - de teljesítmény okokból nem megy az iPhone-on
  - Helyette a fejlesztő dolgozik (előre)
- “Kézi” memóriakezelés
  - referencia számlálás, retainCount
    - retain (növeli a referencia számlálót, lásd propertyk)
    - release (csökkenti a referenciaszámlálót)
    - dealloc (“destruktor”)
      - Ha a referenciaszámláló 0, meghívódik
      - Explicit nem hívjuk meg!

Objective-C: Hasznos osztályok

- Utólag nem módosítható osztályok:
  - NSString
  - NSArray
  - NSSet
  - NSDictionary
- Ugyanezek módosítható verzióban is:
  - NSMutableString
  - NSMutableArray
  - NSMutableSet
  - NSMutableDictionary

Köszönöm a figyelmet!

AMORG APPLIED MOBILE RESEARCH GROUP

Mobilszoftverek

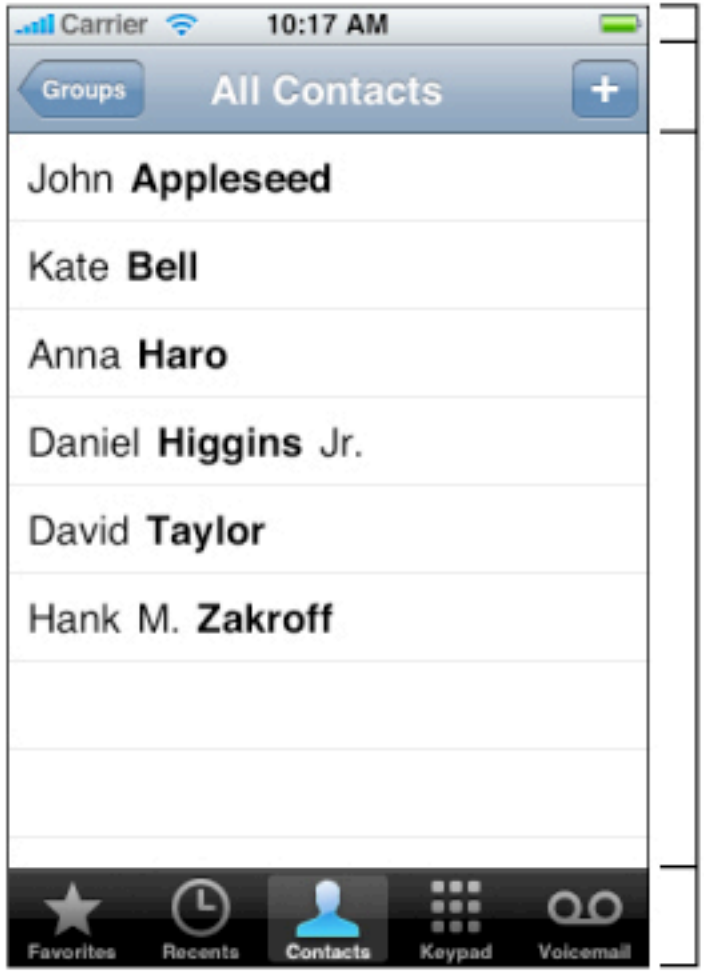
iPhone

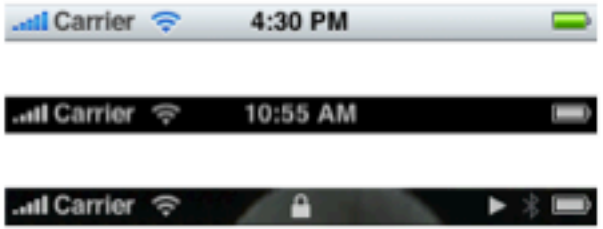
Blázovics László
Blazovics.laszlo@aut.bme.hu

Tartalom

- Az iPhone felhasználói felülete
- I am Rich (Demo)

Az iPhone felhasználói felülete





View-k és ViewController-ek

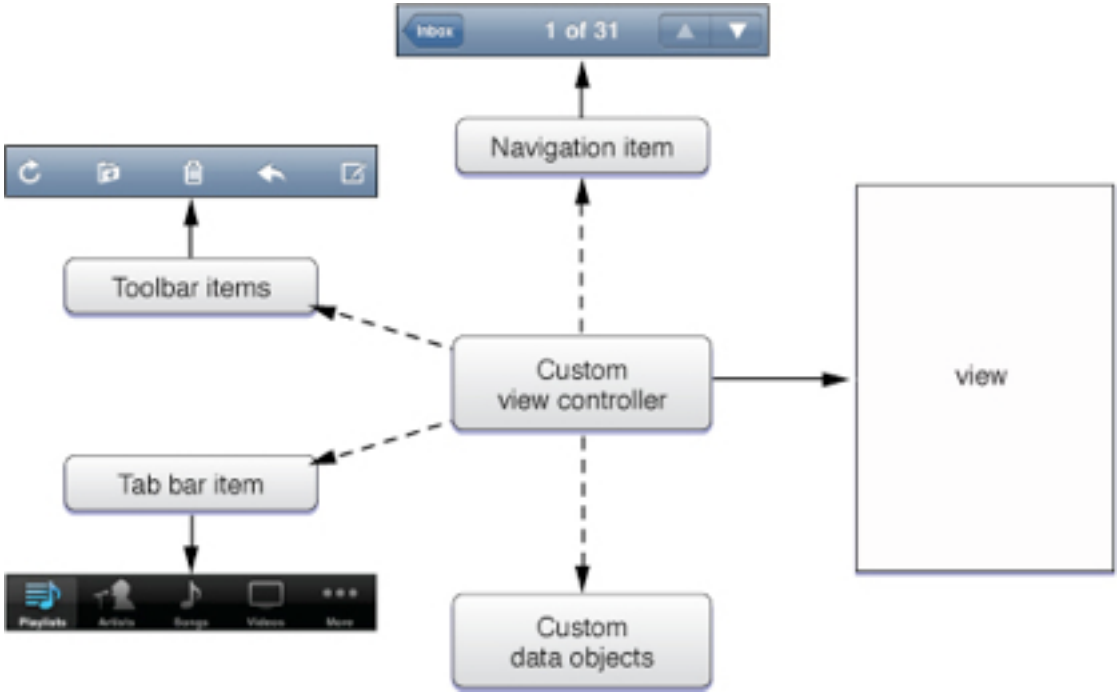
- MVC design pattern
- UIView
  - Felelős
    - A tartalom rajzolásáért
    - A nézet hierarchia kezeléséért
    - A megjelenített nézet animálásáért
    - A nézetet ért események kezeléséért
- UIWindow
  - UIView leszármazott
  - Jellemzően egy van egy alkalmazásban

View-k és ViewController-ek

- UIViewController
  - Felelős
    - A nézete állapotának kezeléséért (betöltés, felszabadítás ...)
    - A megjelenítendő adatok és a nézet kapcsolatának megteremtéséért
    - A nézet “tarozékaiért”
      - Toolbar elemek
      - TabBar elem
      - NavigationBar elemek
    - Az elforgatás kezeléséért
    - A nézet eseményeinek kezeléséért
    - Kritikus memóriahelyzetek kezeléséért
      - Ha nagyon fogy a memória a rendszer értesítést küld neki és felszabadíthatja a nézetét, ha épp nincs megjelenítve

View-k és ViewController-ek

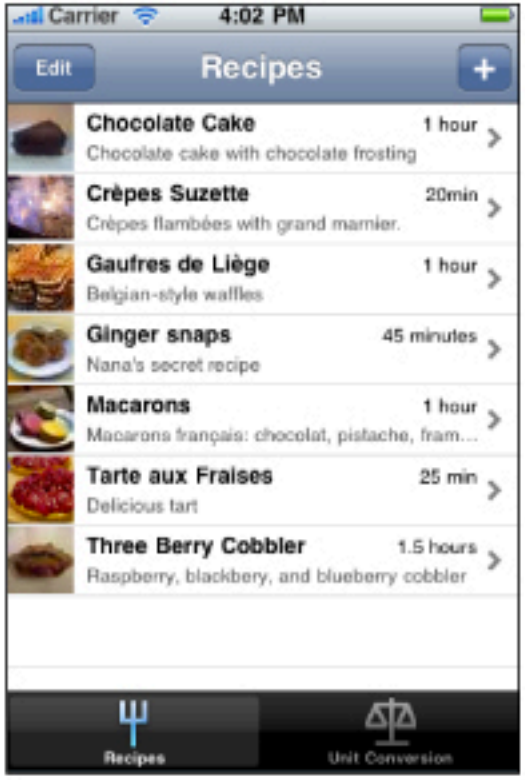
- UIViewController
  - A nézet tartozékai:

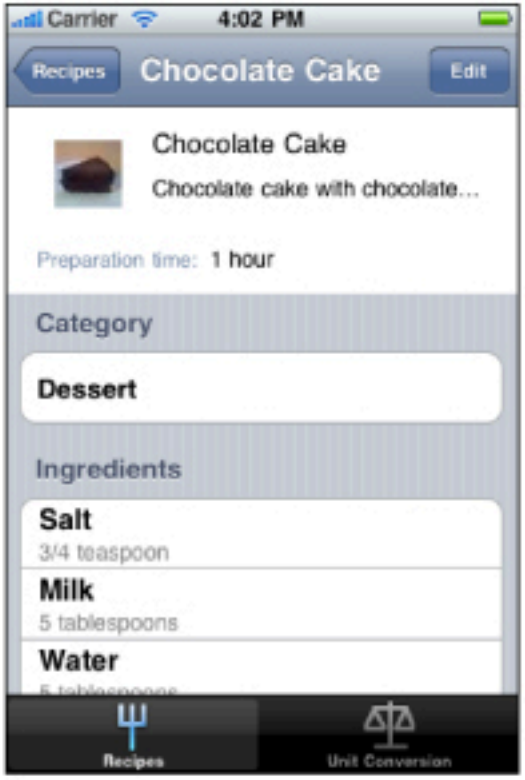


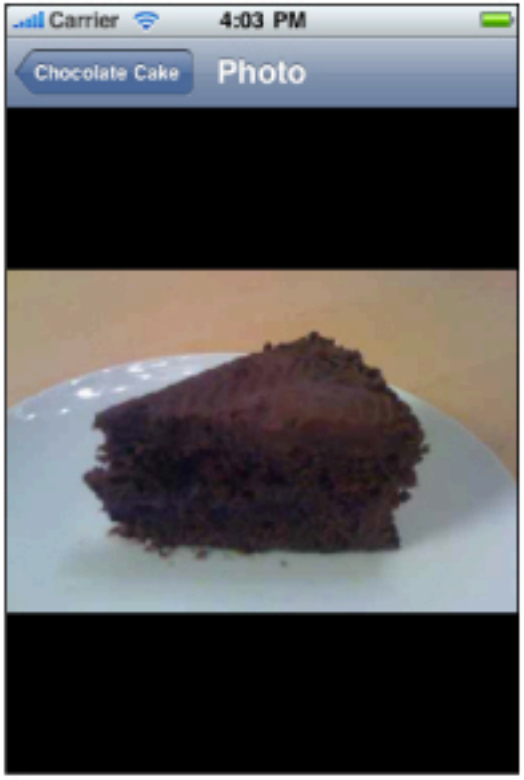
View-k és ViewController-ek

- A legfontosabb beépített ViewController-ek
  - UINavigationController
  - UITabBarController
  - UITableViewController

View-k és ViewController-ek





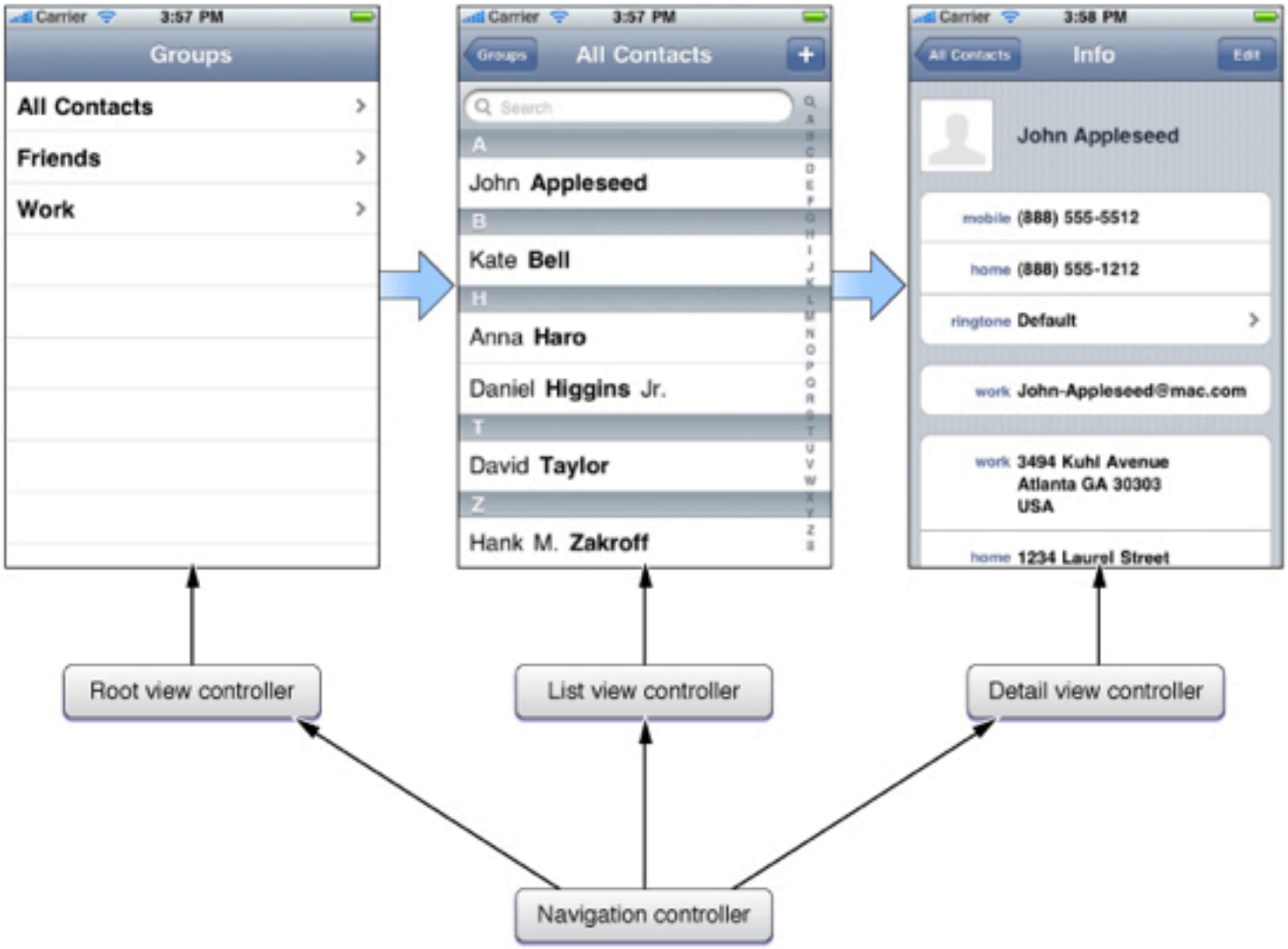


List controller

Detail controller

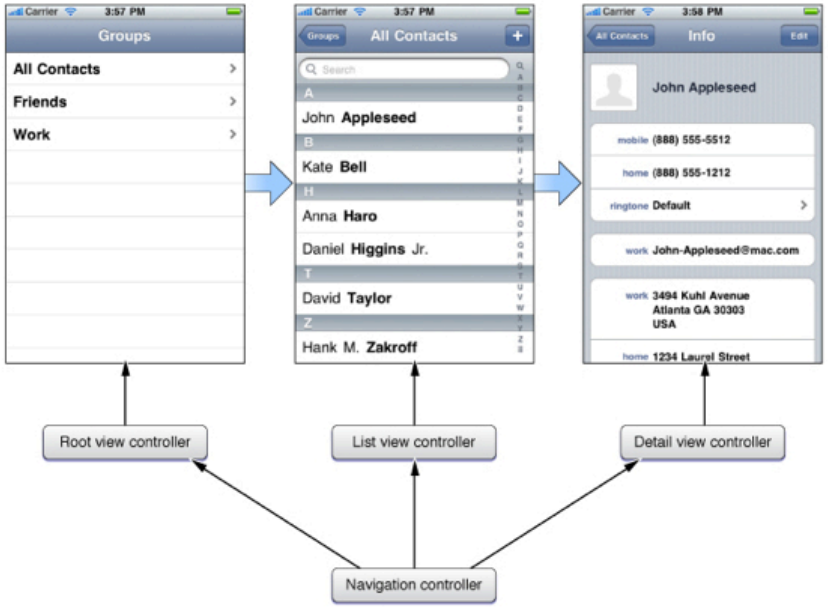
Photo controller

UINavigationController



UINavigationController

- Hierarchia bejárására
- Master / Detail architektúra
- pop és push műveletek view controller-ekkel

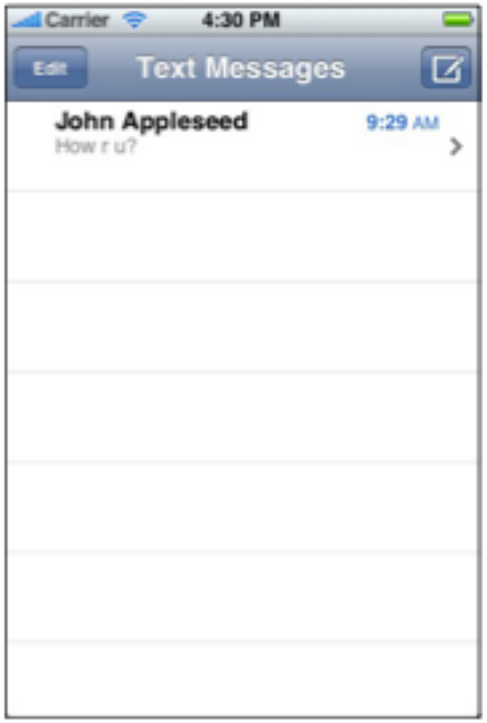


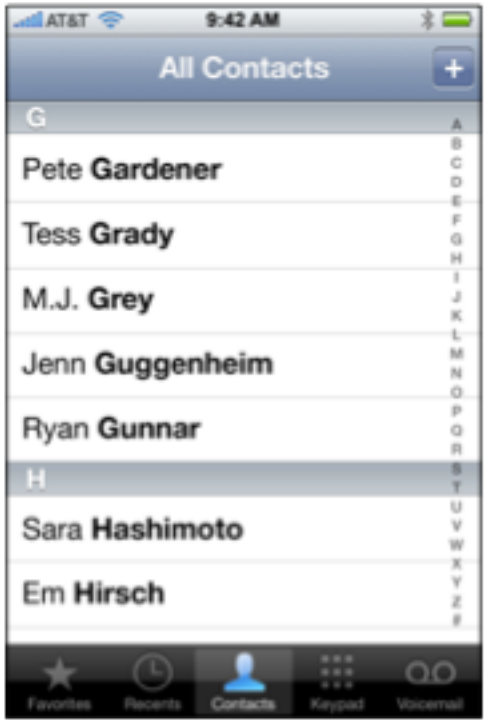
UITabBarController

UITabBarController

- Egymástól független funkciójú nézetek közötti váltásra szolgál

UITableView és TableViewController



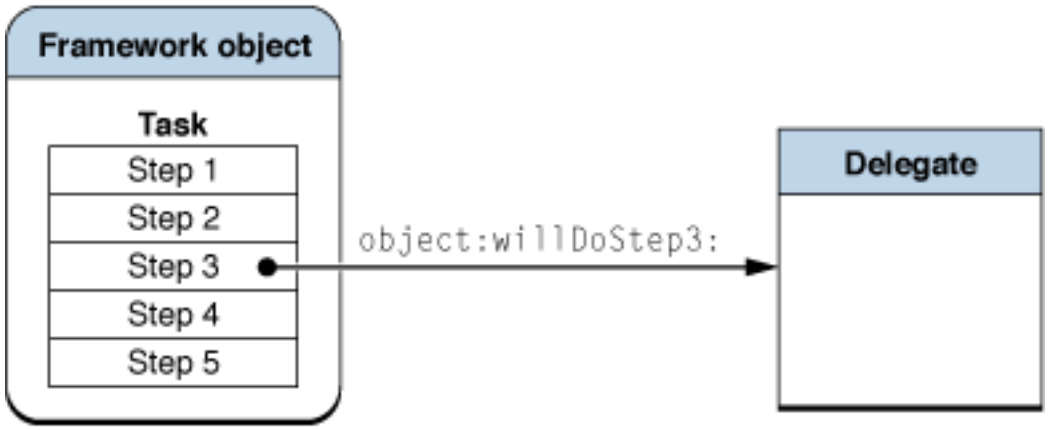


UITableView és TableViewController

- Összetett architektúra
- A UITableView a Decorator design pattern-t követve kérdezi le az adatokat (dataSource) és a megjelenítéshez szükséges információkat (delegate)
  - Jellemzően a vezérlőtől

Decorator design pattern

- Decorator
  - Delegation
    - Öszetett objektumok munkafolyamatába való beleszólás öröklés nélkül
    - Az Objective-C dinamikus implementáció ellenőrzése



Demo

I am Rich

I am Rich (999$) Alkalmazás

- Kirajzol egy gyémántot
- Egy modális nézeten kiírja, hogy mennyire gazdagok vagyunk.

Köszönöm a figyelmet!

AMORG APPLIED MOBILE RESEARCH GROUP

Mobilszoftverek

iPhone

Blázovics László
Blazovics.laszlo@aut.bme.hu

Tartalom

- Memóriakezelés
- Adattárolás

Memóriakezelés az iPhone-on

- A korlátos memória miatt fontos
- Garbage Collection (nincs iPhone-on)
  - Objective-C 2.0
  - Összetett algoritmus figyeli az objektumok egymásra fenntartott referenciáit
    - fogyasztja a szintén korlátos CPU időt és az elemet
- “Kézi” memóriakezelés
  - A programozó rendelkezik a memóriába betöltött objektumok életciklusa felett
  - Több emberi hibalehetőség
  - Sokkal hatékonyabb

Memóriakezelés az iPhone-on

- C
  - malloc és változatai
    - void* malloc(size_t size);
  - free
    - void free(void *pointer);
  - Probléma: összetett struktúrákra mutató pointer pointer tagváltozóit egyenként, rekurzívan kell felszabadítani
- C++
  - new
  - delete

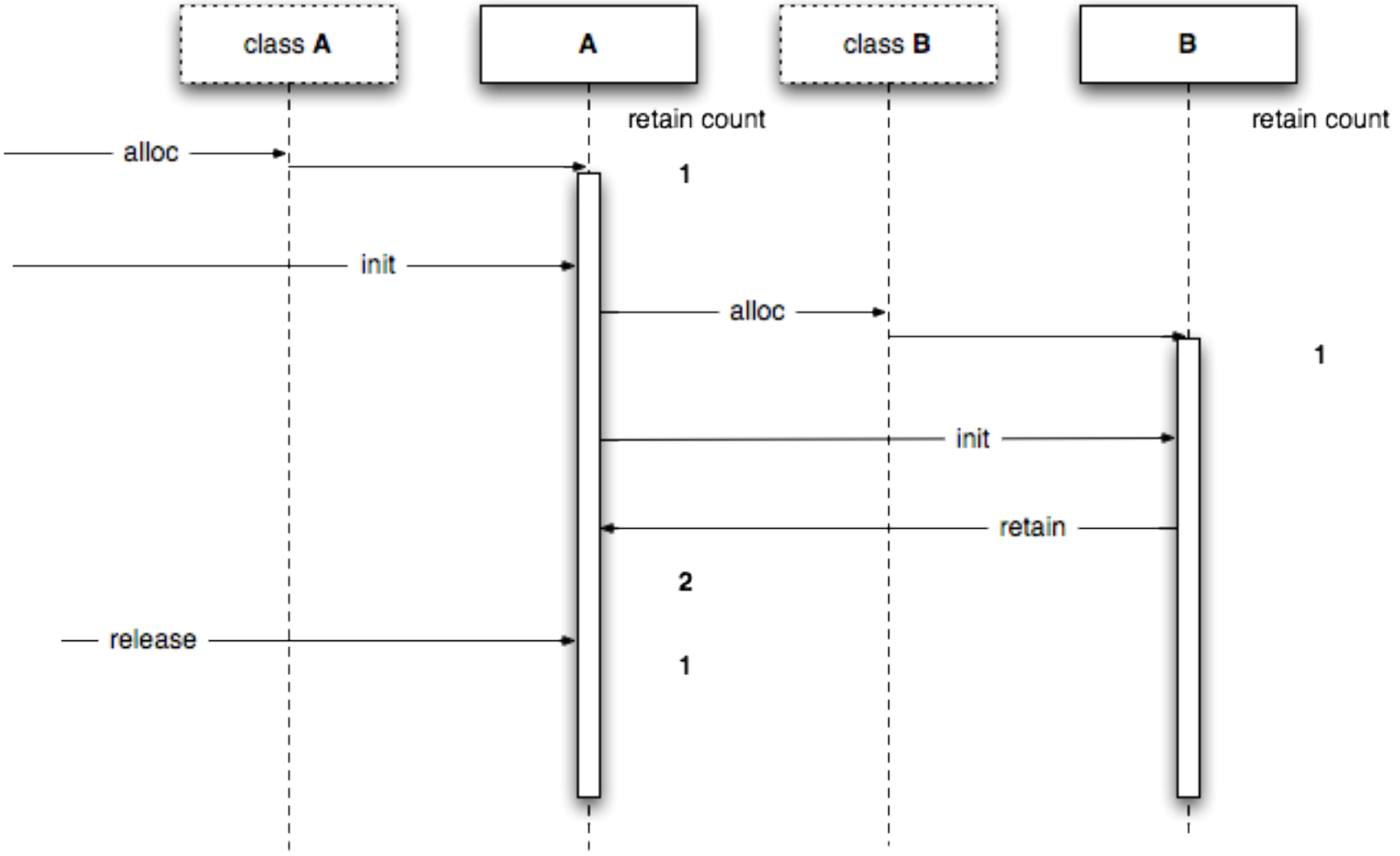
Memóriakezelés az iPhone-on

- Objective-C
  - alloc
    - A memória ilyenkor már le lesz foglalva
    - Minden tagváltozó memóriaterülete ki van nullázva
  - dealloc (kb.: destruktor a C++-ban)
    - Saját kezűleg csak dealloc metódusokon belülről hívjuk ([super dealloc];)
    - A program kilépésekor nem feltételenül kerül meghívásra
      - Az OS hatékonyabban “takarít”
      - Nem érdemes logikát tenni bele, csak felszabadítani a saját felelősségünkbe tartozó memóriákat

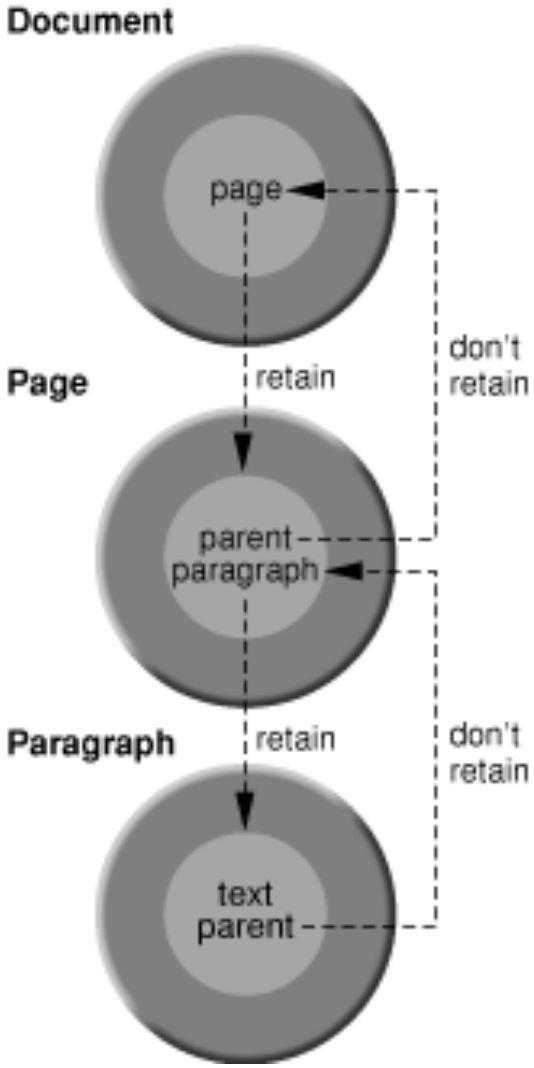
Memóriakezelés az iPhone-on

- Erős konvenciók
  - Minden a programozó által birtokolt objektum felszabadítása (release) a programozó felelőssége
  - A programozó birtokol egy objektumot, ha ...
    - ... ő hozza létre
      - létrehozásnak számít minden “alloc”, “new” illetve “copy” tartalmú üzenettel történő előállítás
    - ... egy vagy több `retain` üzenetet küld neki
      - ugyanennyni release-t kell rajta meghívnia
  - A nem birtokolt objketumokat tilos felszabadítani
    - túl sok felszabadítás -> később valahol a framework is meghív egy utolsó release-t, de ekkor már a memória mást tartalmaz

Memóriakezelés az iPhone-on

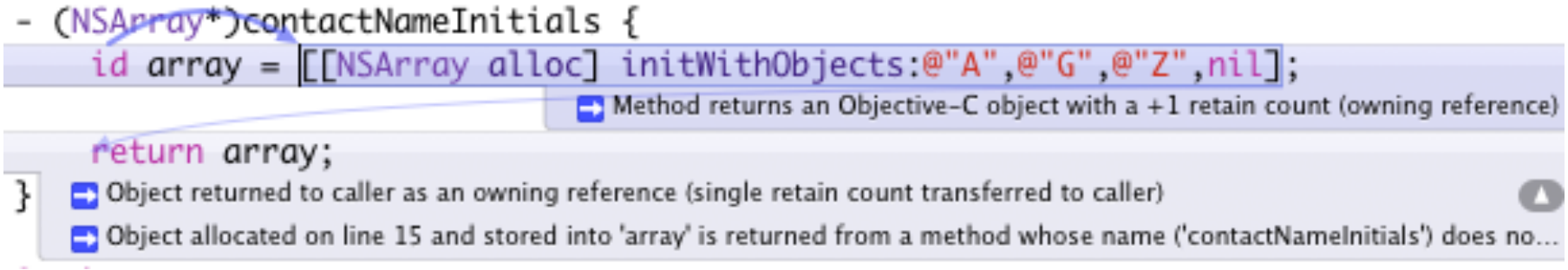


Memóriakezelés az iPhone-on



Memóriakezelés az iPhone-on

```
- (NSArray*)contactNameInitials {
    id array = [[NSArray alloc] initWithObjects:@"A",@"G",@"Z",nil];
    return array;
}
```



Memóriakezelés az iPhone-on

```
- (NSArray*)contactNameInitials {
    id array = [[NSArray alloc] initWithObjects:@"A",@"G",@"Z",nil];
    return array;
}
- (NSArray*)contactNameInitials {
    id array = [[NSArray alloc] initWithObjects:@"A",@"G",@"Z",nil];

    [array release]; //retain count = 0

    return array;
}
```

- A visszaadott objektumnak küldött üzenetek hibát okoznak

Memóriakezelés az iPhone-on

```
- (NSArray*)contactNameInitials {
    id array = [[NSArray alloc] initWithObjects:@"A",@"G",@"Z",nil];
    return array;
}
- (NSArray*)contactNameInitials {
    id array = [[NSArray alloc] initWithObjects:@"A",@"G",@"Z",nil];

    [array autorelease];

    return array;
}
```

- A visszaadott objektum referenciaszáma csak a legbelső autorelease pool felszabadításakor lesz csökkentve.
- De mi is az az AutoReleasePool?

NSAutoreleasePool

- Regisztrálja az autorelease üzenetet kapott objektumokat
- Ha felszabadul, minden regisztrált objektumnak release üzenetet küld
- A main szálban autómatikusan létrejön
- Új szálakban a programozónak kell létrehozni és felszabadítani

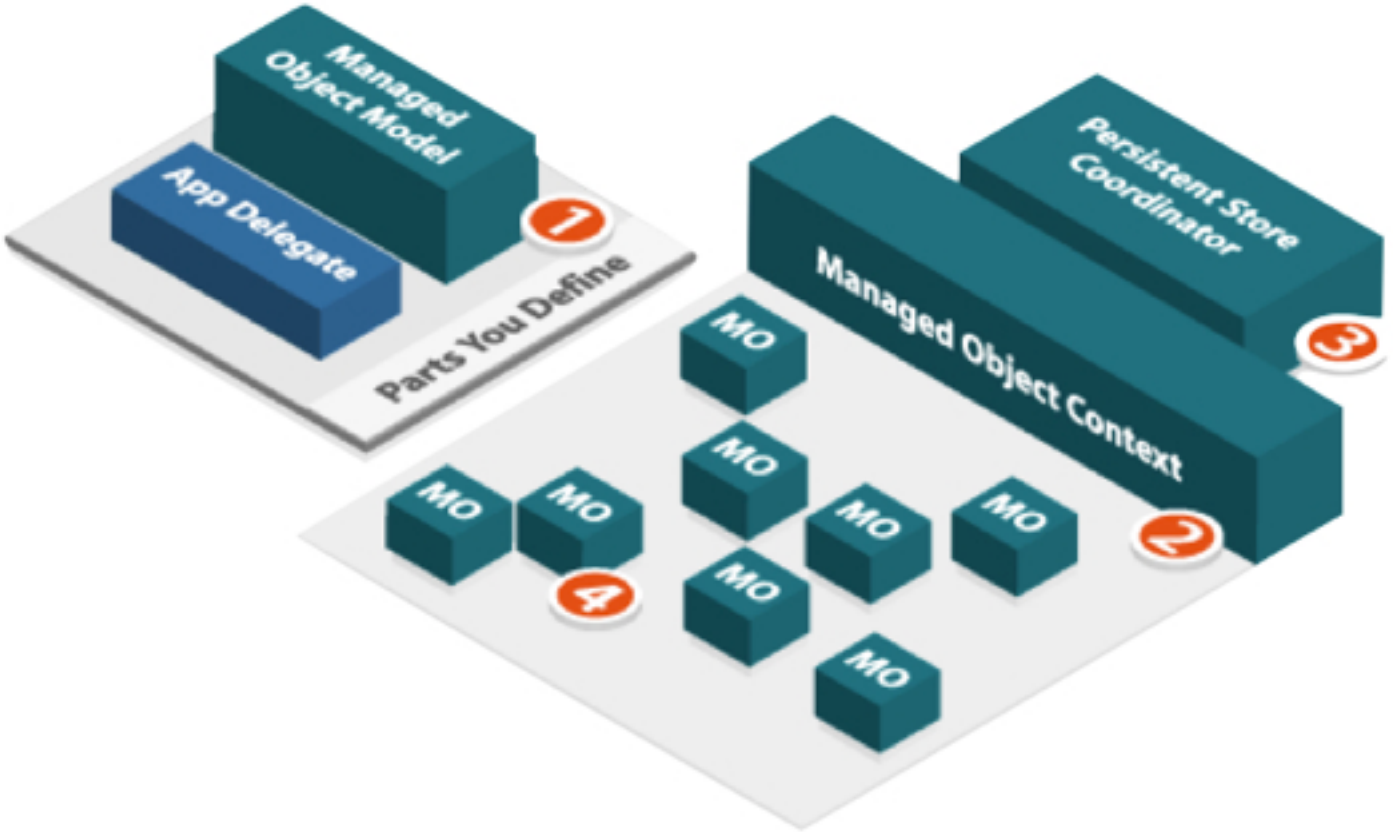
Adattárolás

Adattárolás - Core Data

- “Objektum gráfok és objektumok életciklus kezelésére, beleértve a perzisztenciát is”
- kb.objektum alapú “adatbázis”:
  - KVC, KVO-nak megfelelő objektumok automatikusan a modellből*
  - Property-k értékének automatikus validálása
  - Az objektumok referenciáinak (relációinak) karbantartása (change propagation)
  - Memórián belüli keresés és rendezés műveletek az objektumokkal
  - Az objektumok csak akkor töltődnek be ténylegesen, ha szükség van rájuk

Adattárolás – Core Data

- Core Data

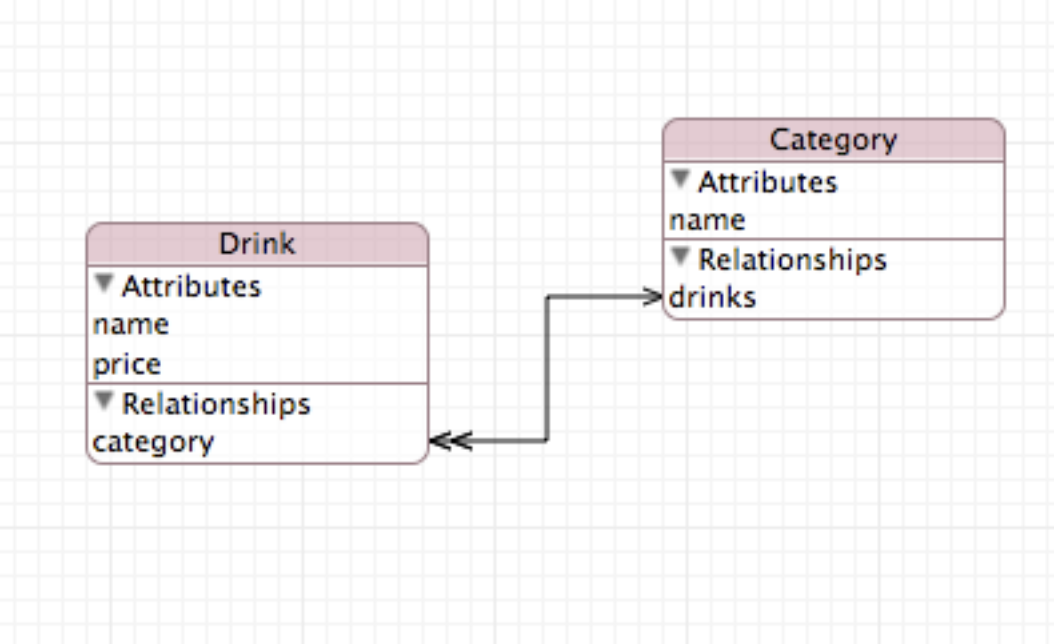


Adattárolás – Core Data

- Csak saját adatforrás, valós RDBMS-hez nem tud csatlakozni
- Nem ORM, csak nagyon hasonló
- SQLite

Adattárolás – Core Data

- Grafikus felület a relációs entitások tervezésére (kb ER diagram)
  - persze a modell létrehozható kódból is ...



Adattárolás – Core Data

- NSManagedObjectModel
  - Az entitások modellje
  - Változtatásakor a korábbi store-ok érvénytelenné válnak
- NSPersistentStoreCoordinator
  - Az adattárolás vezérlője
  - Itt adhatjuk meg a tárolás típusát
    - Bináris, SQLite, XML (iPhone-on nincs)

Adattárolás – Core Data

- NSManagedObjectContext
  - Az objektumok memórián belüli kezelésért felel
  - Több is tartozhat egy NSPersistentStoreCoordinator-hoz
  - Lekérdezések kezelése
    - NSFetchRequest
  - Objektumok frissítése a store-ból
    - Szálak között szinkronizálni kell!
      - Más is dolgozhat ugyanazokkal az adatokkal
  - Változások mentése a store-on keresztül

Adattárolás – Core Data

- Adatok “egyszerű” kinyerése

```
NSManagedObjectContext* moc = [[DataManager defaultManager] managedObjectContext];

NSEntityDescription* categoryEntity = [NSEntityDescription entityForName:@"Category"
                                                   inManagedObjectContext:moc];

NSFetchRequest* request = [NSFetchRequest new];
[request setEntity:categoryEntity];

NSPredicate* notEmptyPredicate = [NSPredicate predicateWithFormat:@"drinks.@count > 0"];
[request setPredicate:notEmptyPredicate];

NSSortDescriptor* sortByName = [[NSSortDescriptor alloc] initWithKey:@"name" ascending:YES];
[request setSortDescriptors:[NSArray arrayWithObject:sortByName]];
[sortByName release];

NSError* error = nil;

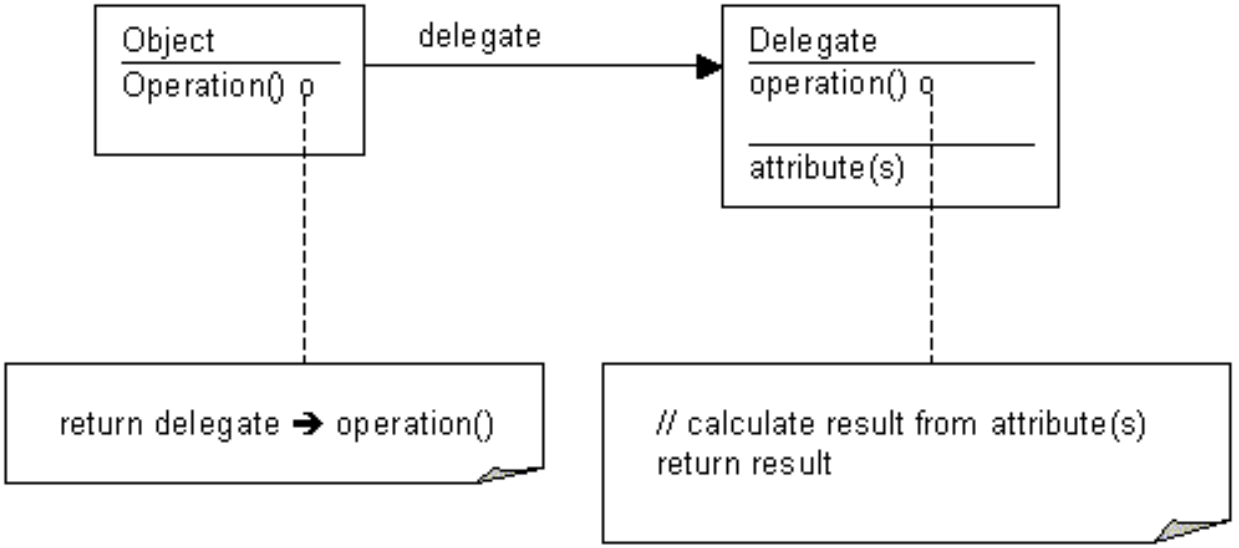
NSArray* result = [moc executeFetchRequest:request error:&error];

if (error) {
    NSLog(@"Error while fetching: %@",error);
}

...
```

Delegate

- Design Pattern
  - Egy objektum egy másik objektummal végezteti a saját feladatát
  - Azt az objetumot hívják delegate-nek, aki a műveletet végezi



Delegate

- Gyakorlatilag minden, aszinkronitást tartalmazó osztály ezt a tervezési mintát használja.
  - View-k és ViewController-ek
    - UITableView
    - UITextField
    - UINavigationController
  - Hálózatkezelő objektumok
    - NSURLConnection
  - Pozíciómeghatározás (Core Location)
    - CLLocationManager
  - …

Köszönöm a figyelmet!

AMORG APPLIED MOBILE RESEARCH GROUP

Mobilszoftverek

iPhone

Blázovics László
Blazovics.laszlo@aut.bme.hu

Nokia Fejlesztői Klub

- Időpont: 2011. április 27.
- Hely: BME Q BF15
  - Újdonsagok a Nokiától - 2011 április
  - Az S40 platform újdonságai. Touch & Type a gyakorlatban
  - Flash Lite alkalmazások: látvány vs. Teljesítmény
  - Alkalmazások UI megvalósítása gyakorlati szempontból
  - Meglepetés 2.

Ha részt vettél a márciusi fejlesztői Klub rendezvényen, akkor kérjük, hozd magaddal az akkor kapott pólódat!

Regisztráció: www.nokia.hu/developer

Tartalom

- Hálózat és szálkezelés
- UITableViewDemo

Hálózatkezelés

- Lehetőségek
  - BSD socket-ek (C)
  - CFNetwork (C) API
  - NSURLConncetion (Objective-C)
  - third party: pl.: ASIHTTPRequest (Objective-C)

Hálózatkezelés - NSURLConnection

- Egy HTTP lekérdezés végrehajtására és feldolgozására szolgál
  - NSURLRequest
    - Majdnem teljesen fix paraméterek
    - Az URL-t lehet csak megadni
  - NSMutableURLRequest
    - beállíthatjuk a HTTP kérés típusát (POST, GET)
    - a HTTP header paramétereket
    - a kérés testét (HTTP body)

Hálózatkezelés - NSURLConnection

- Küldhetünk szinkron kérést ...
  - megvárja, amíg megérkezik az adat és azzal tér vissza
- ... vagy aszinkront
  - ami a delegate-en keresztül értesít az egyes “eseményekről”
    - új adat jött
    - hiba történt
    - kapcsolat vége ...
- Az aszinkron végrehajtás runloop-on történik, nem feltétlenül kell hozzá önálló szál.
- De mi az a RunLoop?

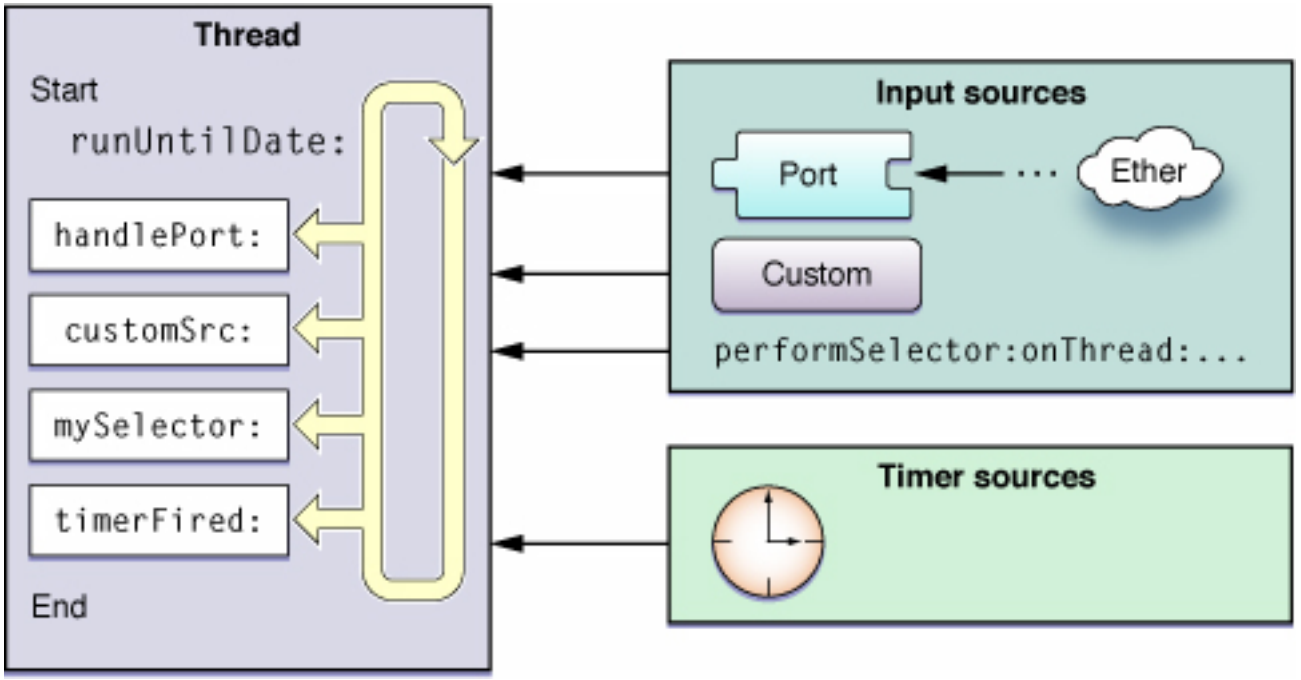
Szálkezelés

- Lehetőségek
  - POSIX Thread (C)
  - NSThread (Objective-C)

Szálkezelés - NSThread

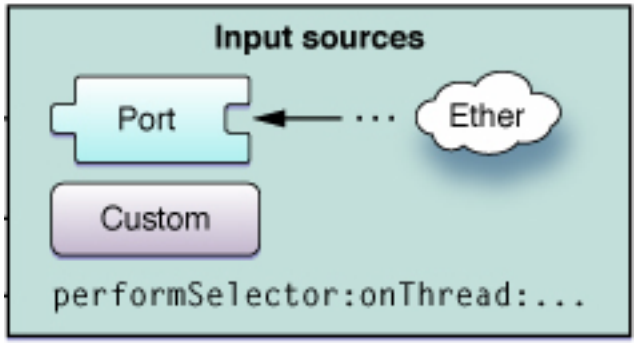
- Létrehozhatjuk
  - Hagyományosan
    - init majd
    - start
    - stop
  - “Leválasztással” Detach
    - detachNewThreadSelector:
    - El se kell indítani
    - Amint visszatér a függvény, a szál megszűnik
- Az AutoreleasePool-t mindkét esetben létre kell hozni!

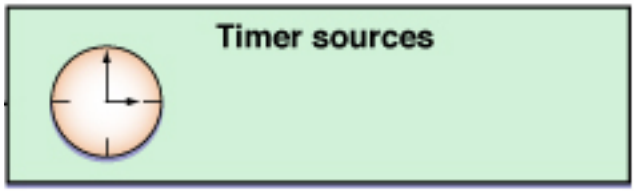
Szálkezelés - RunLoop



Szálkezelés - RunLoop

- RunLoop felépítése
  - Források
    - Portok
    - performSelector:onThread:... hívások
    - Időzítők
    - Egyesdi források
  - Futási módok
    - egy-egy itárció között is változtatható
    - egyfajta szűrő a forrásokra
    - pl.: `NSModalPanelRunLoopMode`





UITableView Demo

- NSURLRequest-el lekérünk egy XML-t
- Az XML-t parseoljuk
- Az eredményt megjelenítjük egy TableView-ban

Köszönöm a figyelmet!

AMORG APPLIED MOBILE RESEARCH GROUP

Mobilszoftverek

iPhone

Blázovics László
Blazovics.laszlo@aut.bme.hu

Tartalom

- Eseménykezelés
- Érintések eseménykezelése
- Multimédia
- Egyéb Frameworkök

Eseménykezelés

NSNotification

- Értesítések eseményekről
- Az érdeklődőnek nem kell ismernie a feladót
- Notification vs delegate
  - Egyszerre akárhány objektum megkaphatja az értesítést
    - cserébe nem lehet értéket visszaadni
  - A notification küldőjének nem is kell tudni a megfigyelők létezéséről
- Mindig azon a szálon kerül meghívásra a megfigyelő megfelelő üzenete, ahonnan a Notification-t küldték
  - csak többlet munkával alkalmas szálak közötti kommunikációra

NSNotification

- Létrehozása
  - `+ notificationWithName:object:`
  - `+ notificationWithName:object:userInfo:`

NSNotificationCenter

- Singleton
  - `+ defaultCenter`
- Szinkron küldési mechanizmus
  - a küldő üznet addig nem tér vissza, amíg az összes megfigyelő meg nem kapta
- aszinkron működéshez NSNotificatoinQueue-k

NSNotificationCenter

- Megfigyelők kezelése
  - `- addObserver:selector:name:object:`
  - `- removeObserver:`
  - `- removeObserver:name:object:`
- Notification küldése
  - `- postNotification:`
  - `- postNotificationName:object:`
  - `- postNotificationName:object:userInfo:`

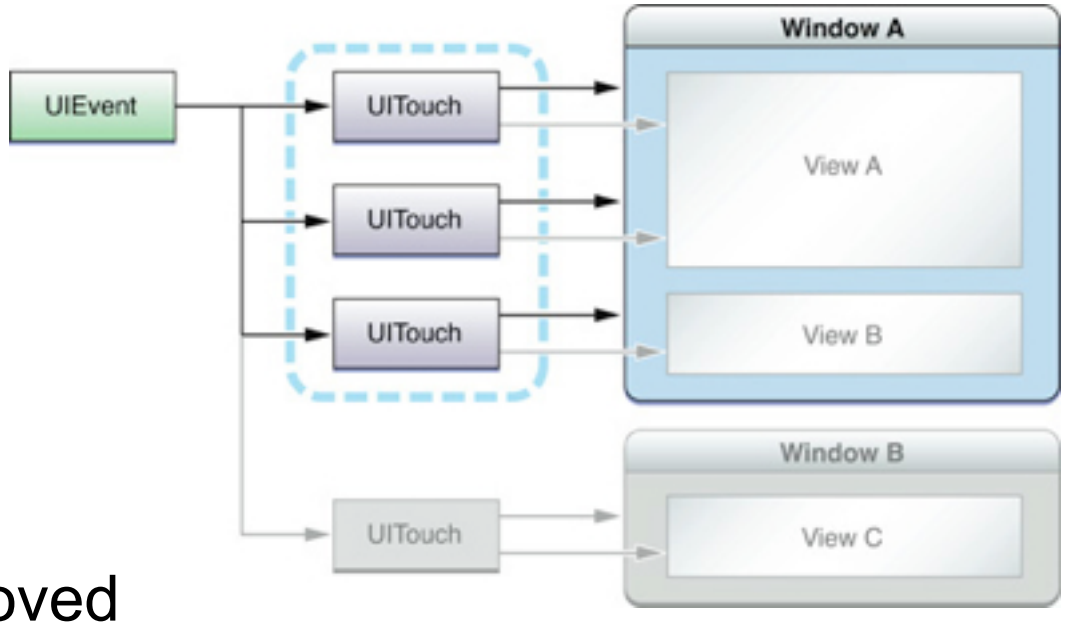
Érintések eseménykezelése

Eseménykezelő üzenetek

- Az egyes fázisok üzenetei
  - touchesBegan:withEvent:
  - touchesMoved:withEvent:
  - touchesEnded:withEvent:
  - touchesCancelled:withEvent:
- Első paraméter egy NSSet, ami az összes aktív UITouch objektumot tartalmazza
  - Adott ujj érintésétől a felemeléséig ugyanaz a UITouch objektum reprezentálja az érintést
- Második paraméter az aktuális UIEvent

UITouch

- locationInView:
- previousLocationInView:
- view
- window
- tapCount
- timeStamp
- Phase
  - UITouchPhaseMoved
  - …



UIGestureRecognizer

- “Komplexebb” gesture-ok felismerése
  - UITapGestureRecognizer
  - UIPinchGestureRecognizer
  - UIRotationGestureRecognizer
  - UISwipeGestureRecognizer
- Általában View-hoz lehet rendelni

UIGestureRecognizer

- Létrehozás
  - - (id)initWithTarget:(id)*target* action:(SEL)*action*
- Kezelés
  - - (void)handleGesture;
  - - (void)handleGesture:(UIGestureRecognizer *) gestureRecognizer;
- Példa:
  - UIRotationGestureRecognizer
    - UIGestureRecognizer leszármazott
    - rotation
    - velocity

Multimédia

AVAudioPlayer

- Objective-C osztály
- Mikor érdemes használni?
  - mindig, ha ...
    - ... nincs szükség stereo pozícionálásra
    - ... precíz szinkronizációra
    - ... és nem stream-et játszunk le

AVAudioPlayer

- Lehetőségek
  - Bármilyen hosszú hang
  - Lejátszás file-ból vagy memória buffer-ből
  - Ismétlés
  - Párhuzamos lejátszás
  - Relatív hangerőszabályozás
  - Tekerés előre/hátra
  - lekérdezhető az egyes csatornák szintje

AVAudioPlayer

- AVAudioPlayerDelegate protokol
  - lejátszás befejezése
  - dekódolási hiba jelzése
  - megszakítás kezelés

MPMoviePlayer

- MPMoviePlayerController osztály

MPMoviePlayer

- MPMoviePlayerController
- Már ViewController (iOS 4.0 óta)
- Egyszerre egy videó
- iPhone-on teljes képernyős
- iPad-en lehet beágyazott
- HTTP Live Streaming támogatás

Egyéb Frameworkök

Frameworkök

- Address Book UI
  - Globális conatct list az alkalmazásunkban
  - UI felületet is biztosít
- Game Kit
  - Peer To Peer kommunikáció (csak iOS)
  - UI támogatás
  - Nem kell hálózatkezelő kódot írni

Frameworkök

- iAd
  - Bannereket lehet vele elhelyezni a UI-on.
  - $$$
- Store Kit
  - In App Purchase
  - Szerver oldali támogatás
  - Csak virtuális terméket lehet vele forgalmazni

Frameworkök

- Map Kit
  - Térkép megjelenítése
  - Online
  - Google Maps
- Core Location
  - GPS alapú helymeghatározás
  - Delegate-en keresztül frissít

Köszönöm a figyelmet!

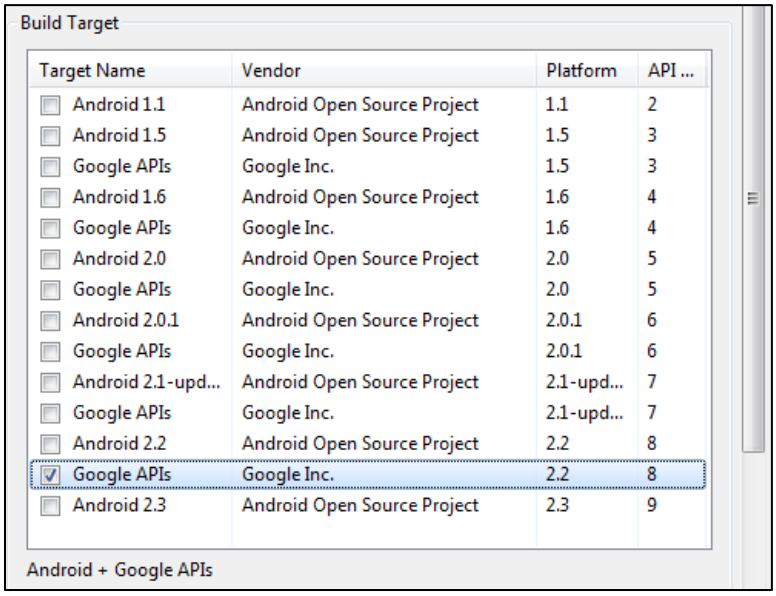
Android tippek & trükkök

MD5 fingerprint generálás MapView használatához:

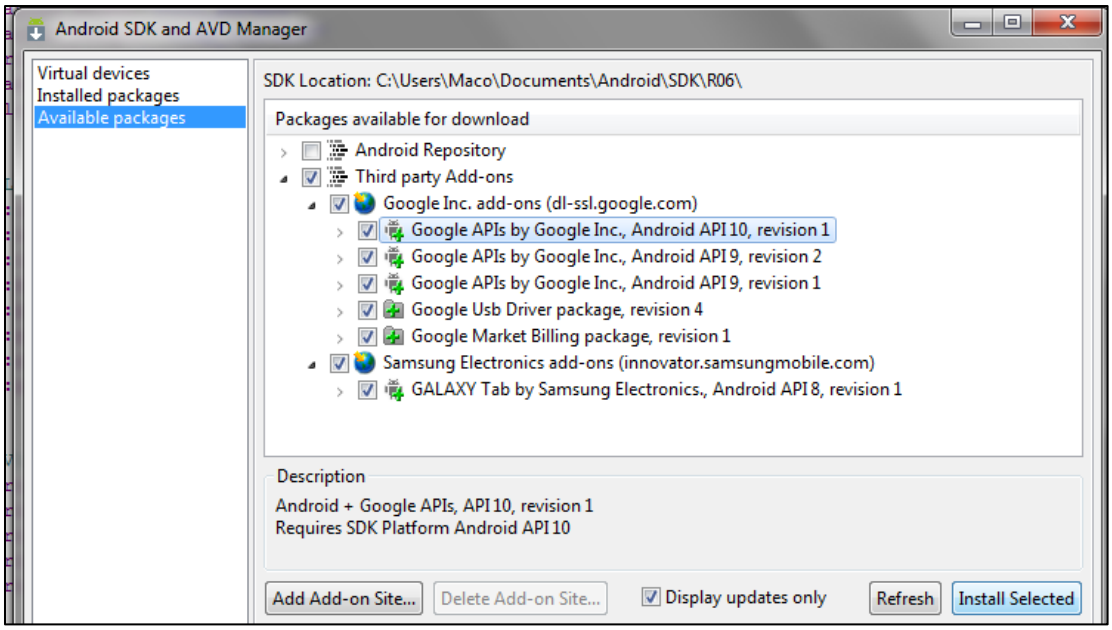
1. Kell lennie egy debug.keystore nevű fájlnak a C:\Users\<felhasználóneved>\.android könyvtárban. Ezt másold át a C:\Java\jdk<verziószám>\bin könyvtárba
2. commans prompt-ban navigálj a Java\jdk<verziószám>\bin könytvárba, és írd be a következő parancsot:
3. keytool.exe –list –alias androiddebugkey –keystore debug.keystore –storepass android –keypass android
4. ez elvileg kiadja az MD5 fingerprintet
5. http://code.google.com/intl/hu/android/add-ons/google-apis/mapkey.html oldalon lévő leírásban megtaláljátok a többit (Registering the Certificate Fingerprint with the Google Maps Service-től)

MapView használatának előfeltétele:

Az Android alkalmazásokban csak akkor működik a MapView (és a többi Google-szolgáltatást használó komponens), ha az alkalmazás Target Platform-nak valamelyik Google APIs nevű van beállítva.

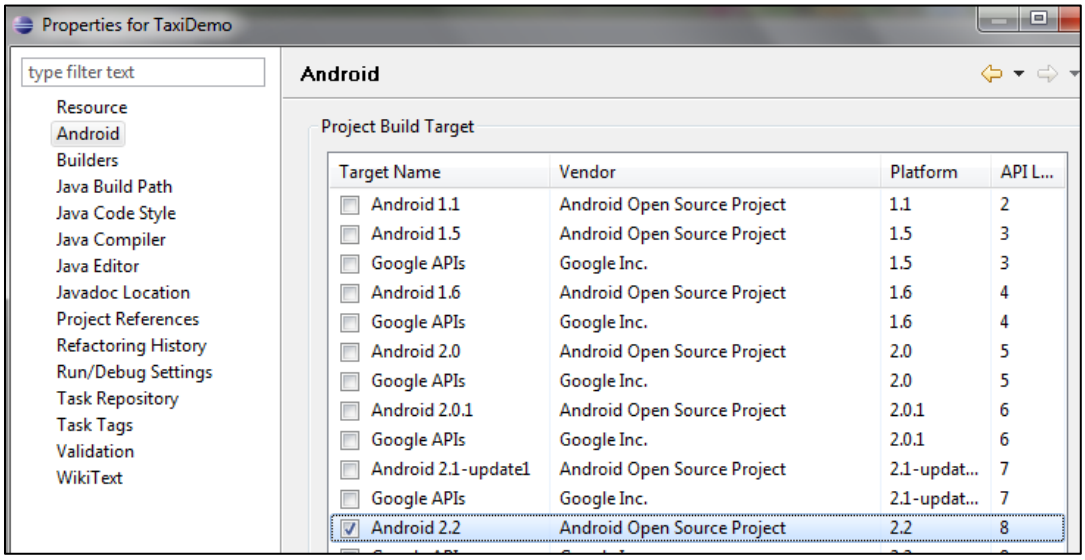


Ha nincs ilyen, akkor le kell tölteni innen:



Már létrehozott projekt Target platform átállítása:

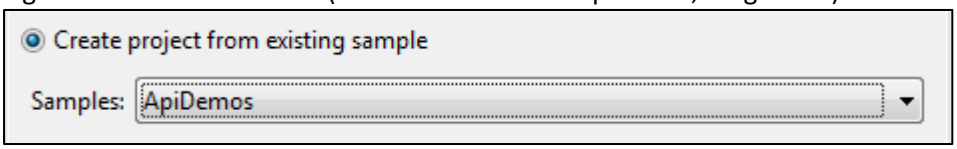
1. Jobb gomb projekt nevén -> Properties
2. Android oldal
3. Project Build Target



API Demos használata

Az Android SDK-val együtt települ egy csomó Google által kiadott demó kód, amivel az oprendszer képességeit szemléltetik. Ezek nagyon hasznosak tudnak lenni, mert egy az egyben másolhatók a kódrészletek. Telepítése:

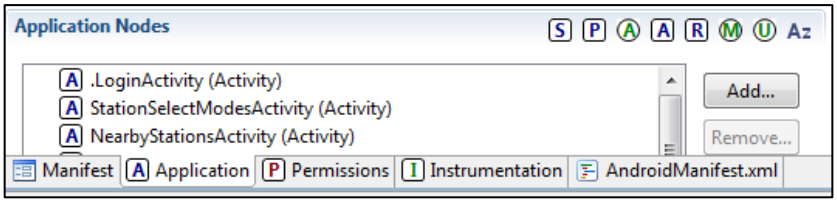
1. Új Android projekt, név megadása
2. Target kiválasztása (Pl. Android 2.2)
3. Contents résznél (felül) Create project from existing samples kiválasztása, majd a Sample legördülőből az API Demos (érdemes a többit is kipróbálni, megnézni!)

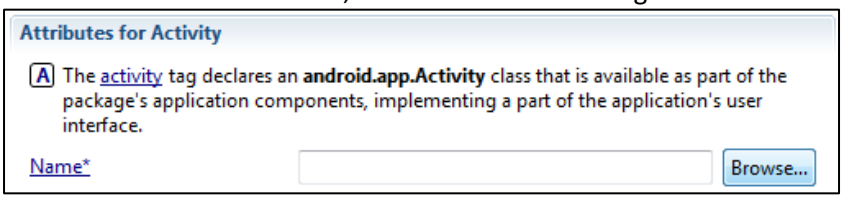


4.
5. Finish

Saját Activity életre bírása

Ha létrehoztok egy saját activity-t, akkor alapból nem lesz futásra képes. Ahhoz hogy az Android engedélyezze a futtatását, be kell rakni az AndroidManifest fájlba. Ennek bevált gyakorlata:

1. AndroidManifest megnyitása
2. Application tab (alul)
3. 

4. Add -> Activity
5. Ekkor jobb oldalt megjelenik egy Attributes for Activity nevű szekció
6. itt a Name kitöltése kötelező, katt mellette a Browse gombra
7. 

8. A felugró ablakban kezdd el beírni az új activity nevét, és megjelenik a listában. Válaszd ki, majd OK
9. mentés

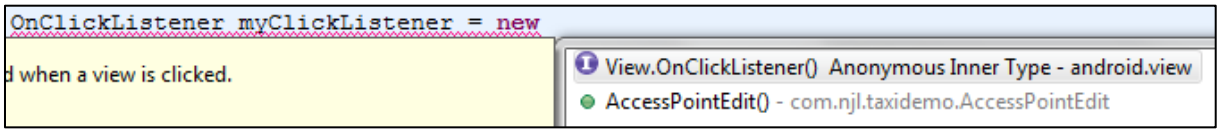
Importok rendberakása

Az Activity-k írása során egy csomó komponenst használunk, amiket be kell importálni. Tipikusan ilyenek a felhasználói felület elemek, pl TextView, EditText, stb. Ezek importálását szerencsére nem kell a fejlesztőnek kezelnie, a Ctrl+Shift+O shortcut rendbe rak mindent. Ami szükséges azt behúzza, ami felesleges azt törli. Szokjatok rá a használatára!

Eseménykezelők keretének gyors behúzása

A felüldefiniálható függvények(tipikusan eseménykezelő, pl OnClickListener) írásához szerencsére nem kell fejből tudni a pontos szignatúrát, ehhez is van támogatás az Eclipse-ben. Ha elkezditek írni a metódus nevét, majd nyomtok egy Ctrl+Entert, akkor felajánlja az így kezdődő felüldefiniálható függvények nevét. Az eseménykezelők esetén ilyenkor az egész törzset odamásolja, ami nagyon hasznos.

new után Ctrl+Enter:



A View.OnClickListener() kiválasztásakor generálódik:

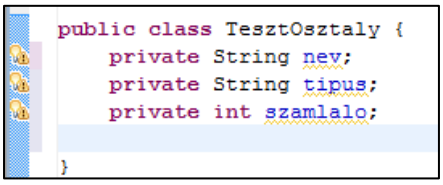
```
OnClickListener myClickListener = new OnClickListener() {

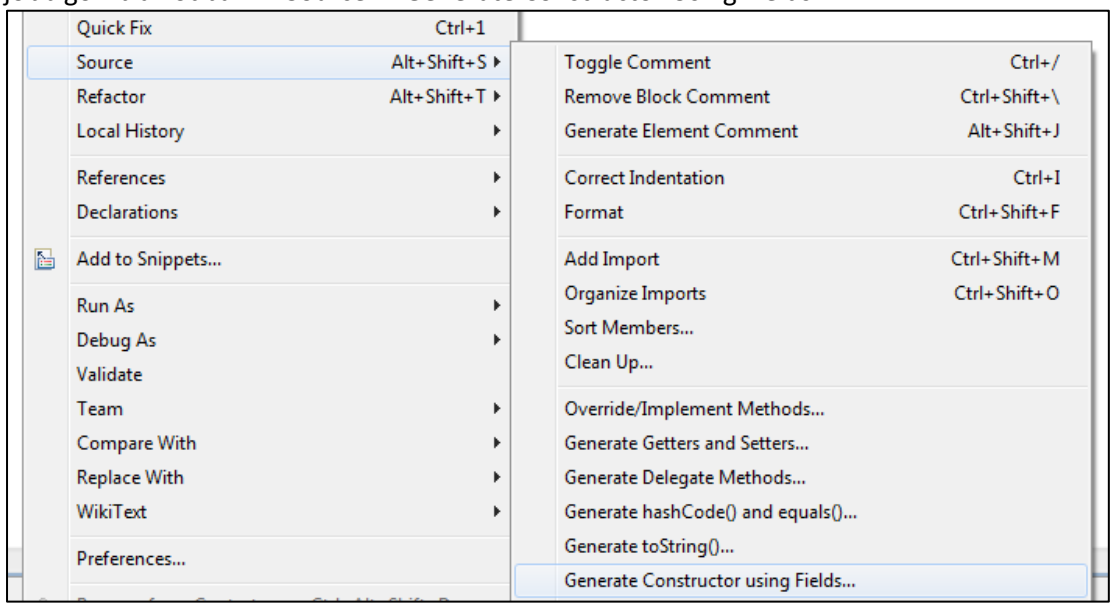
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub

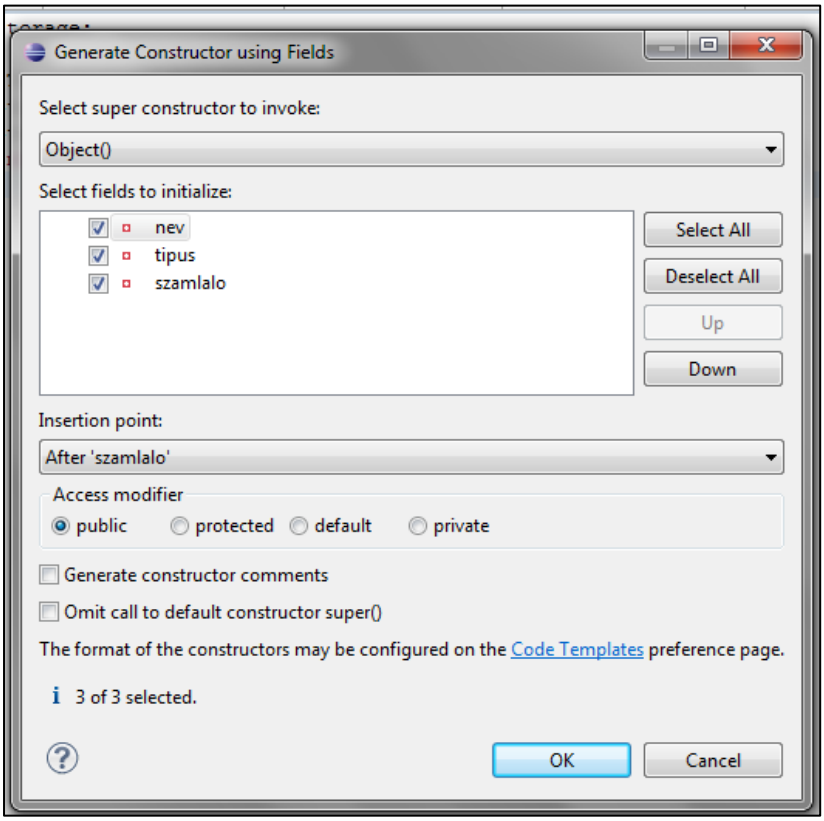
    }
};
```

Konstruktor, getter-setter generálás

Saját osztályok (nem csak Activity) írásakor a mechanikus dolgok, mint konstruktor és getter-setter generálás feladatát át tudja venni az Eclipse. Működése:

1. Megírjuk az osztály fejlécét és a tagváltozóit:
2. 
```
public class TesztOsztaly {
    private String nev;
    private String tipus;
    private int szamlalo;

}
```
3. jobb gomb a kódban -> Source -> Generate Constructor Using Fields
4. 

5. A felugró ablakban még lehet beállítgatni hogy melyik mezőket akarjuk a konstruktorba, majd OK
6. 

7. kész

```java
public class TesztOsztaly {
    private String nev;
    private String tipus;
    private int szamlalo;

    public TesztOsztaly(String nev, String tipus, int szamlalo) {
        super();
        this.nev = nev;
        this.tipus = tipus;
        this.szamlalo = szamlalo;
    }

}
```

8. (érdemes egy üres konstruktort is csinálni, ami paraméterek nélkül működik)

Getter-setter generálás:

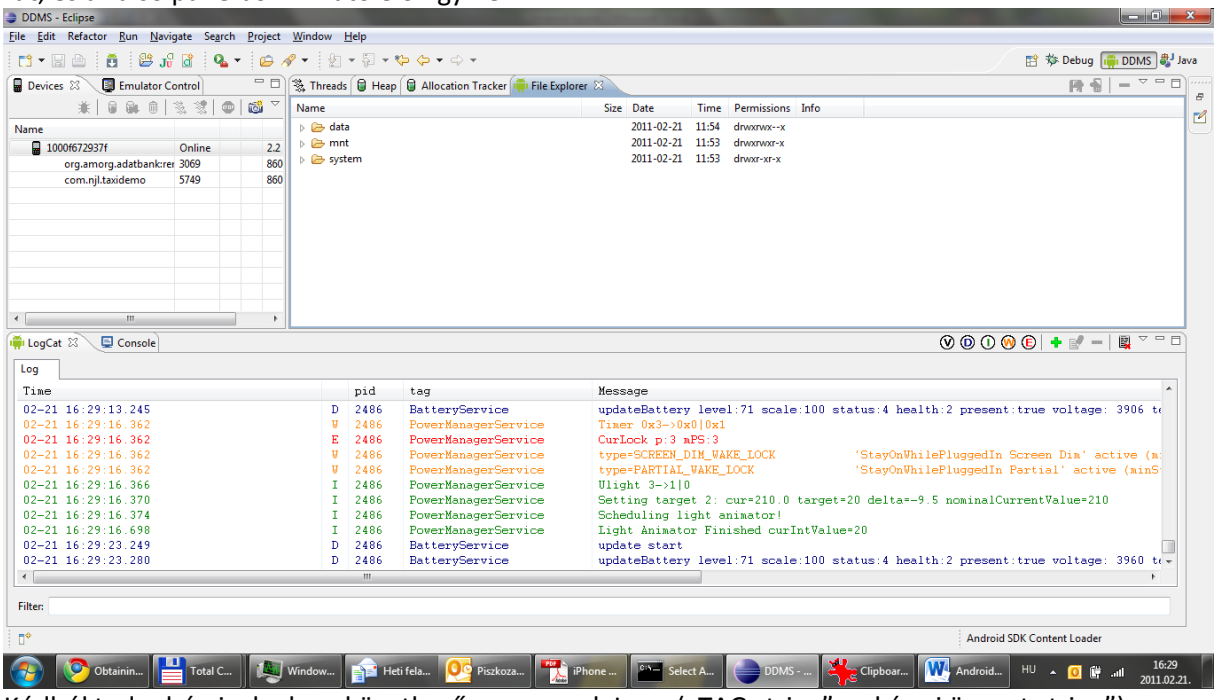
1. jobb gomb a kódban -> Source -> Generate Getters and Setters
2. Select All
3. OK

DDMS felület előhívása

Szintén az SDK-val együtt kapunk az Eclipse-hez egy úgynevezett DDMS felületet, amivel a futó emlulátor vagy csatlakoztatott telefon irányítható, lekérdezhető, log vizsgálható, memória figyelhető, stb. Előhívása:

1. Eclipse Window menü -> Open Perspective ->Other ->DDMS
2. Ha fut emulator vagy telefon, akkor bal fent a Devices fülön látszik a neve. Erre kattintva betölti az adatokat

Itt nézzetek szét, sokmindent tud. Leggyakrabban a napló vizsgálatára használjuk ami LogCat néven fut, és az alsó panelből hívható elő. Így néz ki:



Kódból tudunk írni a logba a következő paranccsal: Log.v(„TAG string”, „akármi üzenet string”);

Android

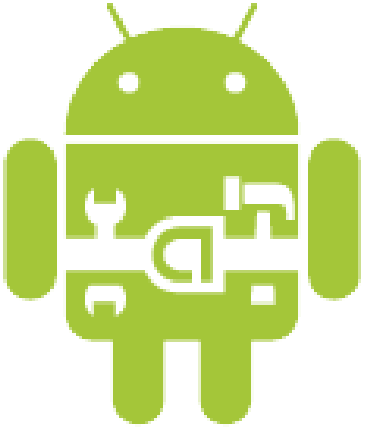
Mobilszoftverek

A1

Fehér Marcell

Feher.Marcell@aut.bme.hu

Tartalom képekben



Tartalom - Android

- 3 előadás + 2 gyakorlat
- Témák
  - Ma
    - Platform bemutatás
    - Felhasználói felület készítés
    - Alkalmazás komponensek kommunikációja
  - Jövő héten
    - Adatmegosztás, adatkötés, globális események kezelése
    - Animáció
    - Felhasználó értesítésének módjai
  - Két hét múlva
    - Térkép és pozíció alapú szolgáltatások

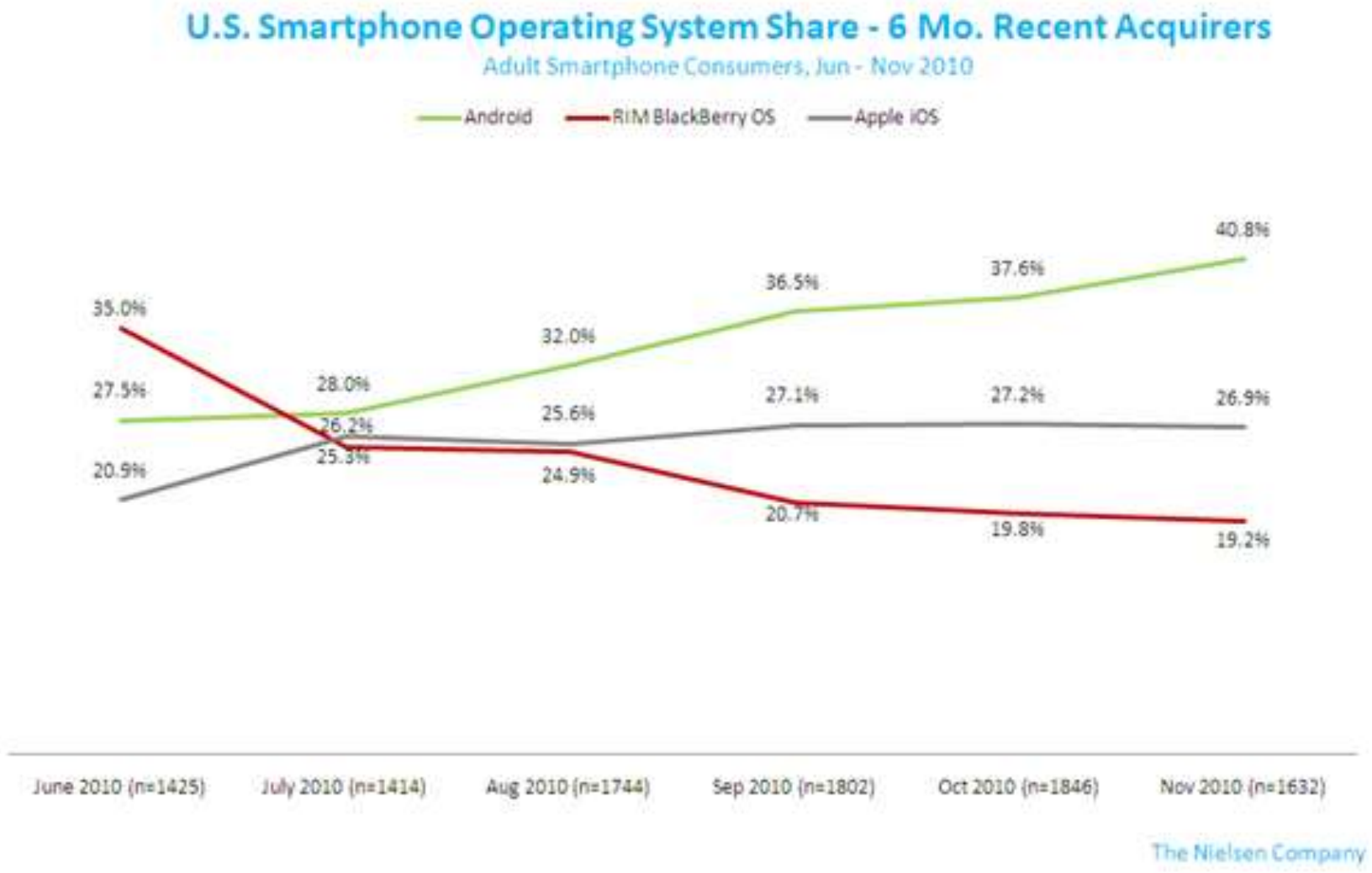
A1 - Tartalom

- Platform bevezetés
- Activity-k és a felhasználói felület
  - Erőforrások
  - Nézetek, UI építőkockák
  - Hardver függetlenség, lokalizáció
- Alkalmazás-komponensek kommunikációja
  - Intentek
  - Intent filterek

Android

Platform bemutatása

Miért beszélünk az Androidról?



Miért beszélünk az Androidról?

- Naponta több, mint 300.000 új Androidos készüléket aktiválnak
- Android Marketben (alkalmazásbolt) összesen ~175.000 szoftver elérhető
- Ingyenes fejlesztői eszközök, alkalmazás feltöltés
- Minden jel szerint a növekedés tovább folytatódik
  - Mind telefonon, mind tableten

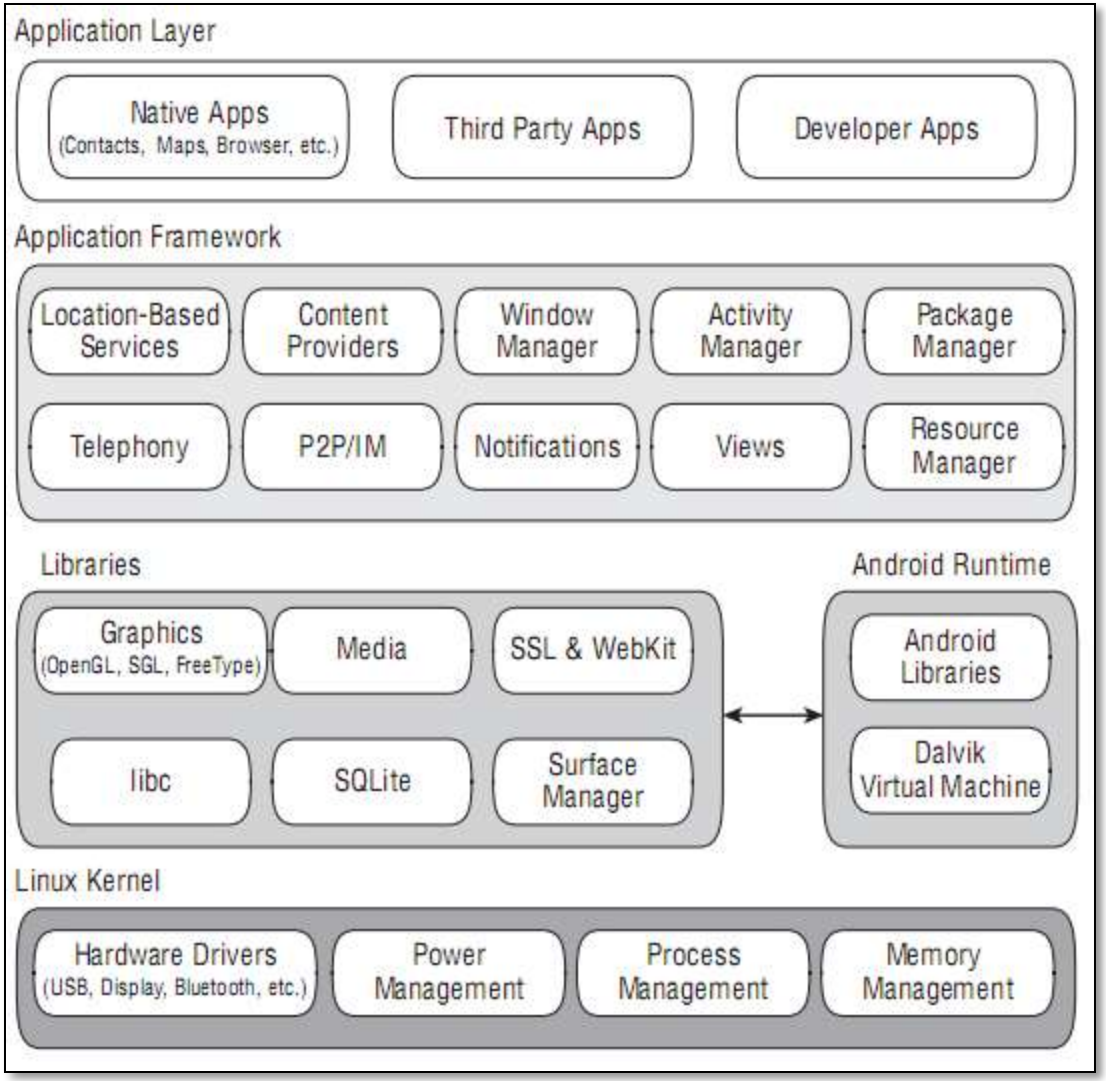
Platform bemutatás

- Az Open Handset Alliance fejleszti
  - Meghatározó a Google, sok olyan IT gyártó a tagja, akik ekkor léptek be a mobil piacra
- Nyílt Linux kernelre épül
- Mobil készülékekre optimalizált virtuális gépet használ
  - „Dalvik Virtual Machine”
  - Arra tervezték, hogy hatékonyan futtasson párhuzamosan több JavaVM-t

Nyílt platform

- A platform teljesen nyílt
  - például egy alkalmazás képes a készülék alap programjaival kommunikálni, el tudja érni a telefonáláshoz szükséges funkciókat és használhatja a készülékbe épített hardware eszközöket (kamera, BlueTooth, stb)
- (…és, mivel az alkalmazások is egyenrangúak …)
- Akár a „beépített” szoftvereket is le lehet cserélni

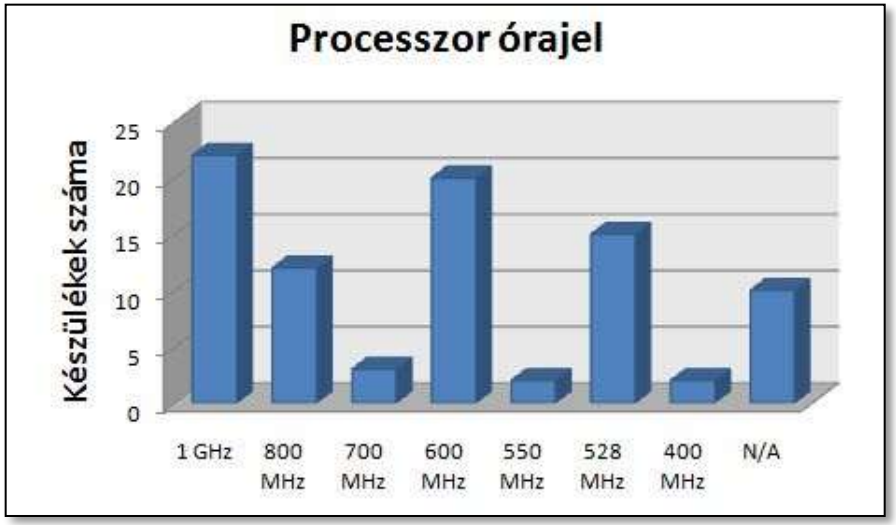
Felépítés

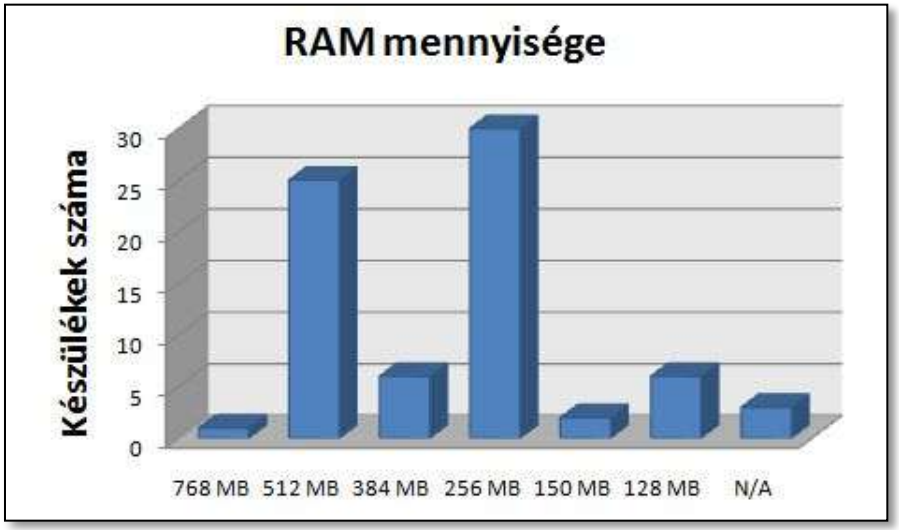

Application Layer
Native Apps
(Contacts, Maps, Browser, etc.)
Third Party Apps
Developer Apps
Application Framework
Location-Based Services
Content Providers
Window Manager
Activity Manager
Package Manager
Telephony
P2P/IM
Notifications
Views
Resource Manager
Libraries
Graphics
(OpenGL, SGL, FreeType)
Media
SSL & WebKit
libc
SQLite
Surface Manager
Android Runtime
Android Libraries
Dalvik Virtual Machine
Linux Kernel
Hardware Drivers
(USB, Display, Bluetooth, etc.)
Power Management
Process Management
Memory Management

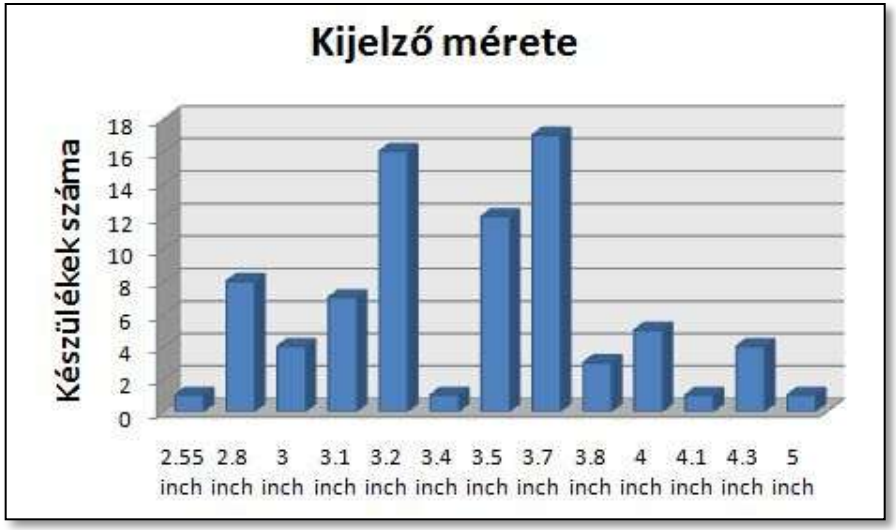
Hardware

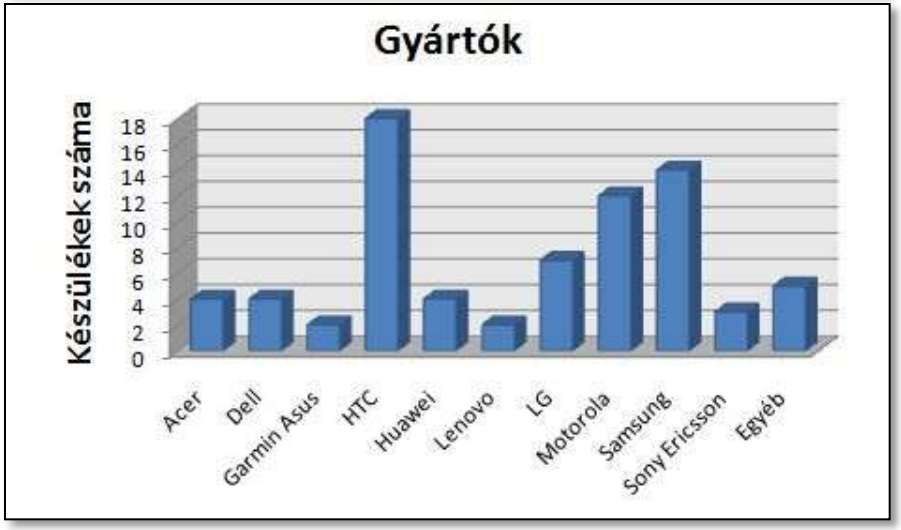
- Jelenleg ~100 telefon és tablet futtat Andriodot
  - Valamint további 20-30 eszköz, aminek semmi köze nincs a telefóniához (pl kávéfőző)
- Nincs HW követelmény, megbirkózik nagyjából bármilyen konfigurációval
- Tabletre a 3.x verziók ajánlottak, telefon méretű képernyőre 2.x

Hardware – 2010 október

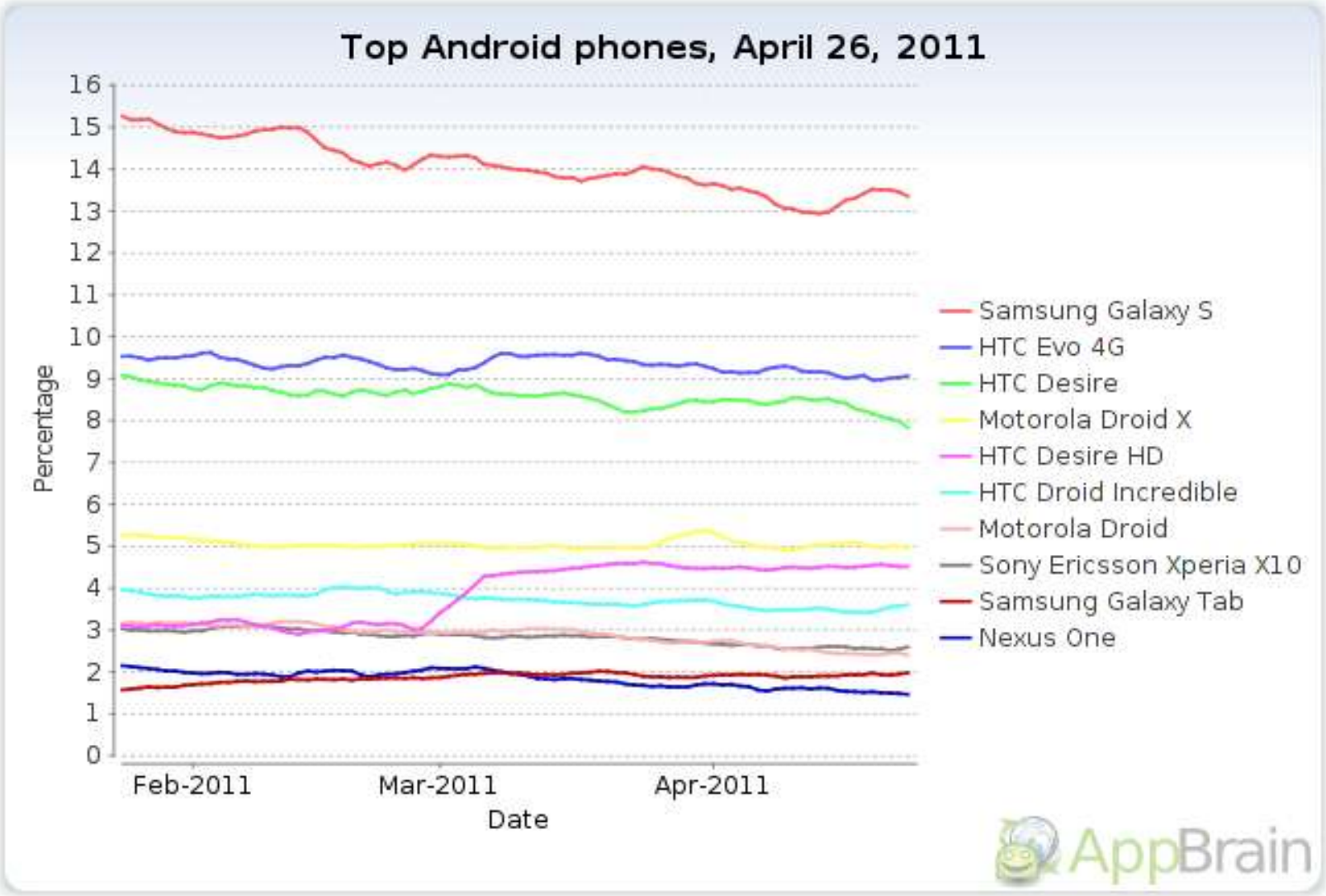






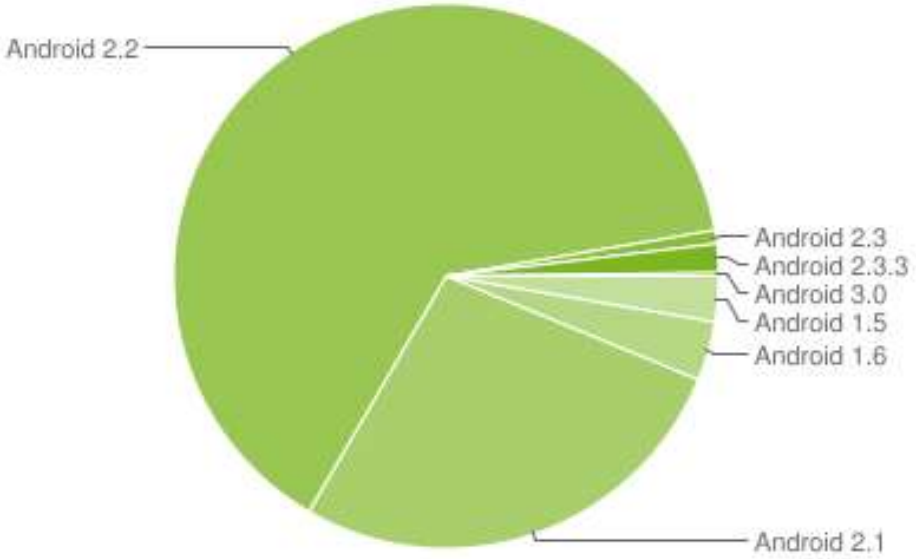


Hardware – 2011 április



Software– 2011 április

- Új verzió kb félévente
- A felhasználók hajlandók frissíteni
  - Kényelmes update, megmarad minden alkalmazás, beállítás, stb…



Software

- Sok gyártó – sokféle kinézet
- Ugyanaz a core, csak a driverek és a UI saját

HTC Sense

MotoBLUR

Default Android

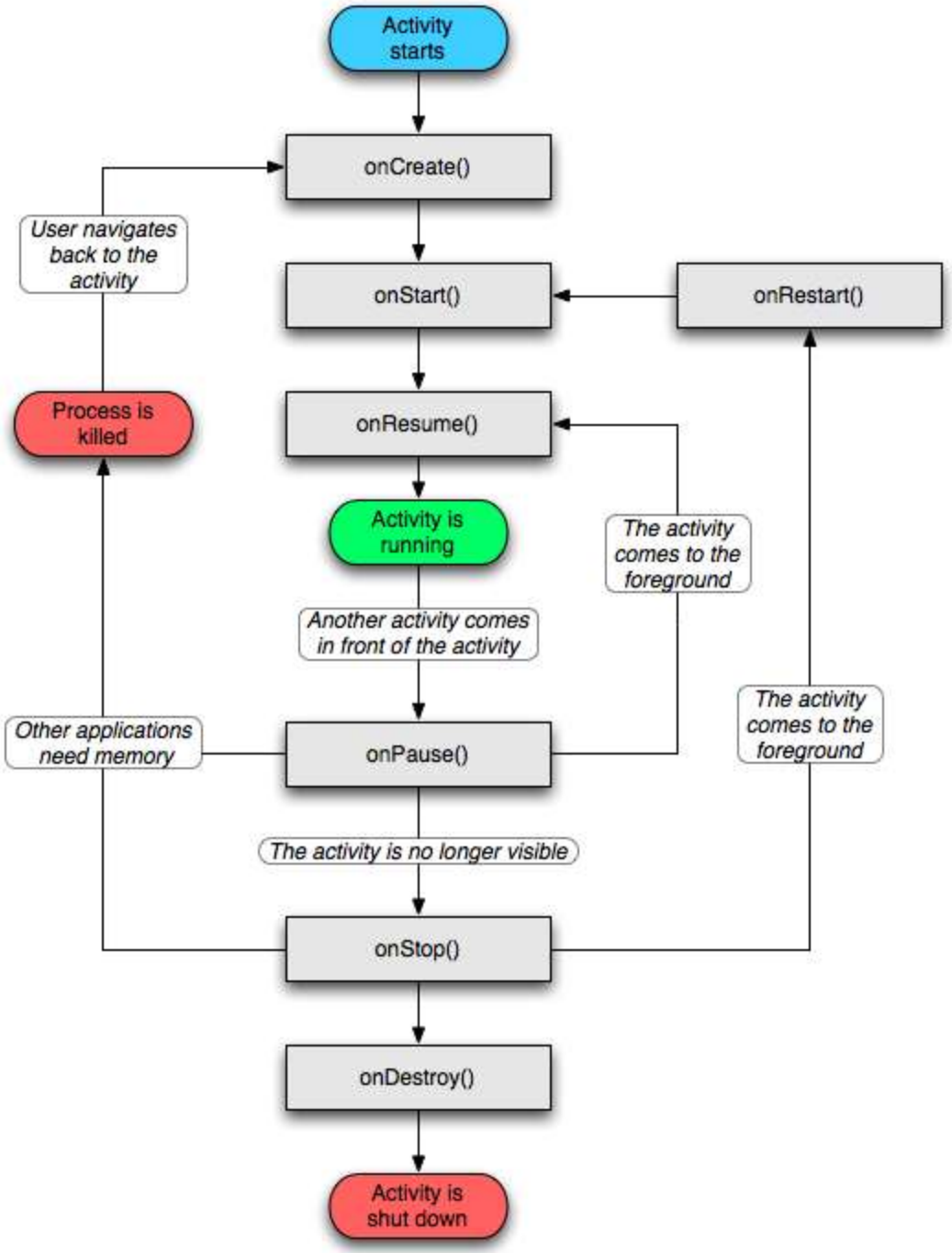
Mit telepítsünk?

- Android SDK
  - Jelenleg R06 (3.0 platform)
- Eclipse
  - **3.5 Galileo** (3.6 Helios nem ajánlott!)
- Android Development Tools plugin (ADT plugin)
- Készítsünk Android Virtual Device (AVD)-t

Android

Felhasználói felület, erőforrások kezelése

Activ


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

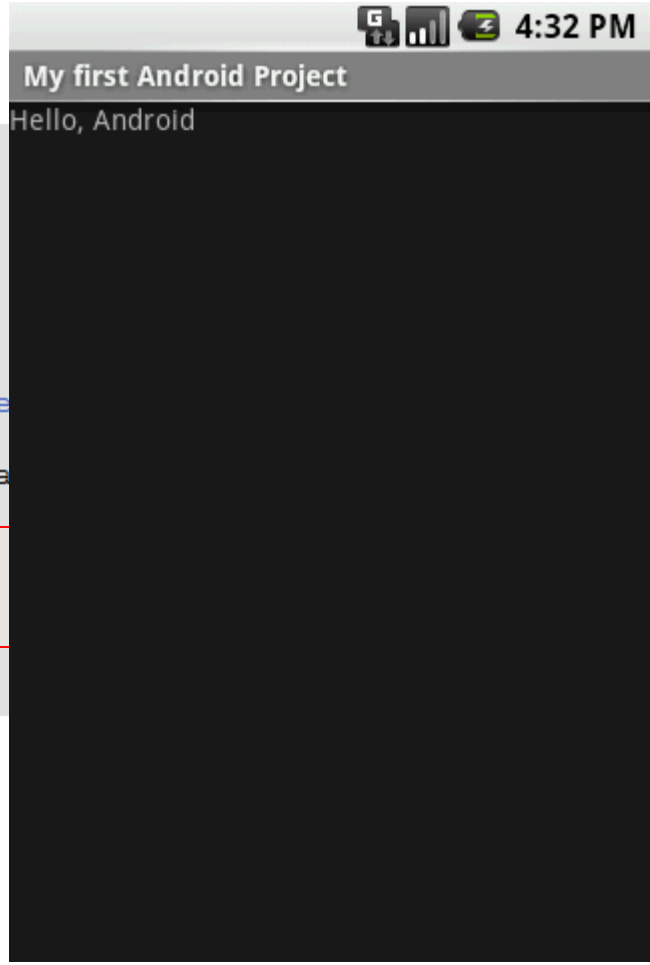
DEMO

- Készítsünk HelloWorld alkalmazást 1 perc alatt

```
package bute.examples.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

```
package bute.examples.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

```
package bute.examples.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first create
    @Override
    public void onCreate(Bundle savedInstanceSta
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```



Vissza az erőforrás fájlhoz

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
```

/res/values/strings.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloAndroid</string>
    <string name="app_name">HelloAndroid</string>
</resources>
```

Erőforrás hivatkozása

```
package hu.bute.helloandroid;

import android.app.Activity;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView img = (ImageView) findViewById(R.drawable.mypicture);
    }
}
```

Erőforrás hivatkozása

```
package hu.bute.helloandroid;

import android.app.Activity;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView img = (ImageView) findViewById(R.drawable.mypicture);
    }
}
```

Erőforrás hivatkozása

```
package hu.bute.helloandroid;

import android.app.Activity;

public class Hell
    /** Called wh                                    ed. */
    @Override
    public void o                                    ate) {
        super.onC
        setConten
        ImageView                                    d(R.drawable.mypicture);
    }
}
```

[..]
[drawable]
[drawable-land-480x320]
[drawable-port-480x320]
[layout]
[values]

Hardverfüggetlenség és i18n

MCC és MNC	**Mobile Country Code, Mobile Network Code**
Language and region	**The two letter ISO 639-1 language code optionally followed by a two letter ISO 3166-1-alpha-2 region code (preceded by lowercase "r"). For example `fr`, `en-rUS`, `fr-rFR`, `es-rES`.**
Screen dimensions	`small (QVGA)`, `normal (HVGA)`, `large (VGA+)`
Wider/taller screens	`long`, `notlong`
Screen orientation	`port`, `land`, `square`
Screen pixel density	`ldpi (120)`, `mdpi (160)` , `hdpi (240)`, **vagy explicit (pl. 96dpi)**
Touchscreen type	`notouch`, `stylus`, `finger`
Whether the keyboard is available to the user	`keysexposed`, `keyshidden`, `keyssoft`
Primary text input method	`nokeys`, `qwerty`, `12key`
Whether the navigation keys are available to the user	`navexposed`, `navhidden`
Primary non-touchscreen navigation method	`nonav`, `dpad`, `trackball`, `wheel`
Screen dimensions	`320x240`, `640x480`, etc.
~~**SDK version**~~	~~**The Android 1.0 SDK is `v1`, the 1.1 SDK is `v2`, and the 1.5 SDK is `v3`.**~~

Hardverfüggetlenség és i18n

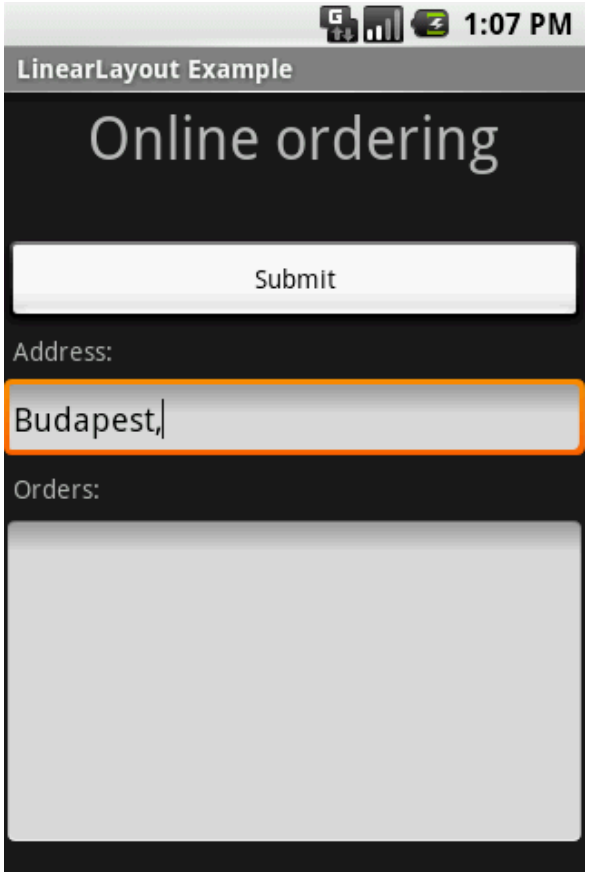
MCC és MNC	**Mobile Country Code, Mobile Network Code**
Language and region	**The two letter ISO 639-1 language code optionally followed by a two letter ISO 3166-1-alpha-2 region code (preceded by lowercase "r"). For example `fr`, `en-rUS`, `fr-rFR`, `es-rES`.**

```
MyApp/
    res/
        drawable-en-rUS-finger/
        drawable-port/
        drawable-port-160dpi/
        drawable-qwerty/
```

the user	
Primary text input method	`nokeys`, `qwerty`, `12key`
Whether the navigation keys are available to the user	`navexposed`, `navhidden`
Primary non-touchscreen navigation method	`nonav`, `dpad`, `trackball`, `wheel`
Screen dimensions	`320x240`, `640x480`, etc.
SDK version	**The Android 1.0 SDK is `v1`, the 1.1 SDK is `v2`, and the 1.5 SDK is `v3`.**

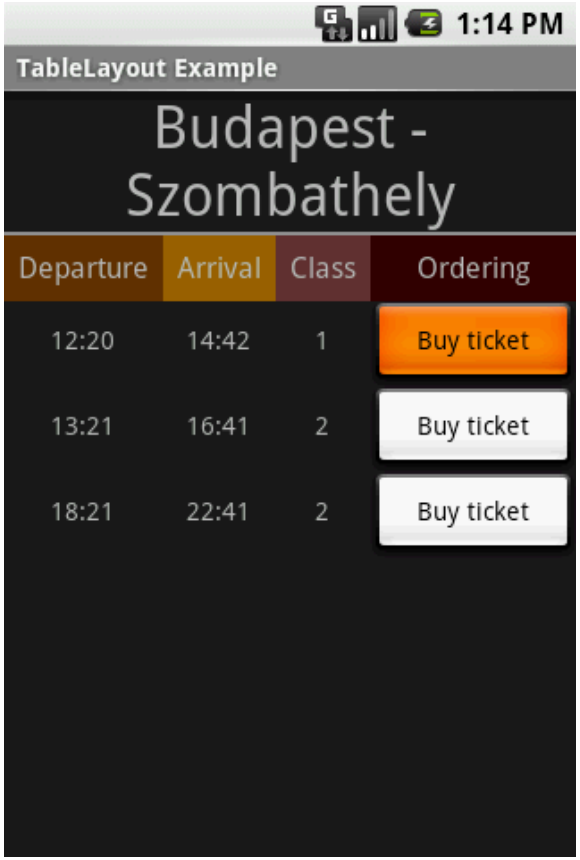
Néhány fontosabb elrendezés

- LinearLayout



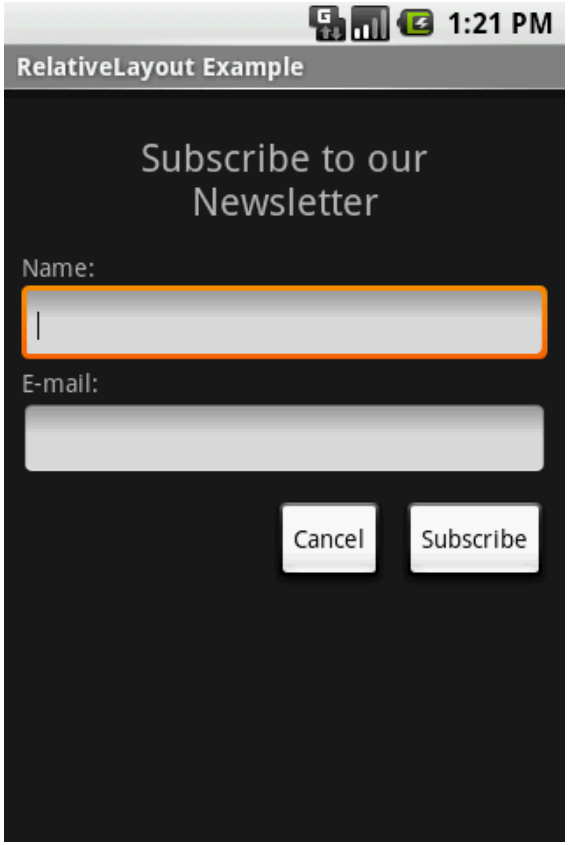
Elrendezések 2

- TableLayout



Elrendezések 3

- RelativeLayout



Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton

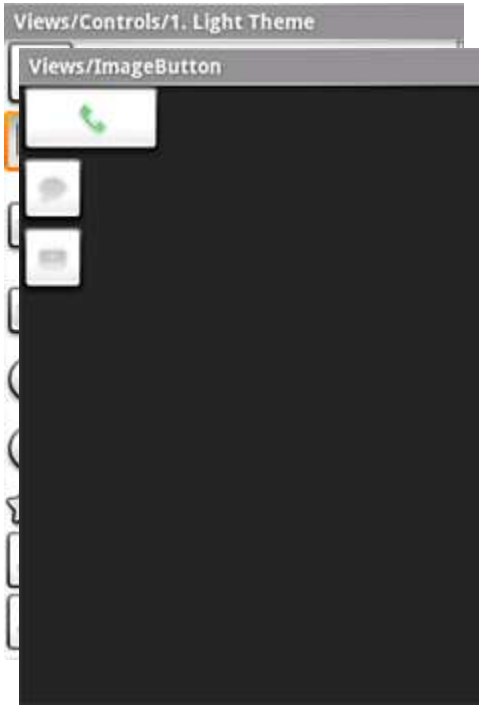
Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton

Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton

Widgetek



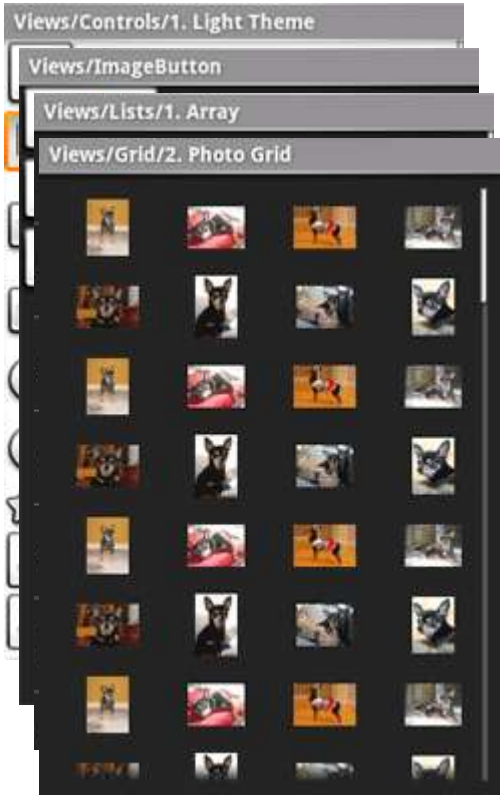
- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton

Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton

ListView

Widgetek

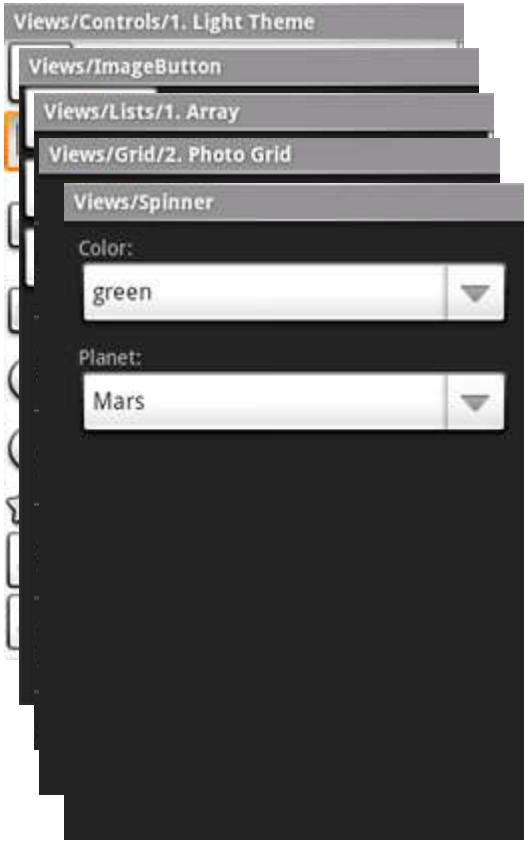


- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton

ListView

GridView

Widgetek



- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton

ListView

GridView

Spinner

Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton
- ListView
- GridView
- Spinner
- AutoCompleteTextView

Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton
- ListView
- GridView
- Spinner
- AutoCompleteTextView
- Gallery

Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton
- ListView
- GridView
- Spinner
- AutoCompleteTextView
- Gallery
- ImageSwitcher

Widgetek

- Button, EditText, CheckBox, RadioButton, ToggleButton
- ImageButton
- ListView
- GridView
- Spinner
- AutoCompleteTextView
- Gallery
- ImageSwitcher
- DatePicker, TimePicker

Perzisztencia

- Shared Preferences
- File-ok
- SQLite adatbázis
- Content Provider

Shared Preferences

- Egyszerű mód kulcs/érték párok tárolására
- Tipikusan: alkalmazás beállítások, vagy komponensek közötti adatmegosztás
  - boolean, string, float, long, integer

Példa

```
SharedPreferences mySharedPreferences = getSharedPreferences(
    “MySavedPrefKey”, Activity.MODE_PRIVATE);
SharedPreferences.Editor editor = mySharedPreferences.edit();
editor.putBoolean(“isTrue”, true);
editor.putFloat(“yourAvg”, 1f);
editor.commit();
//----------------------------------------------------
// Adatok visszatöltése
boolean isTrue = mySharedPreferences.getBoolean(“isTrue”, false);
float avg = mySharedPreferences.getFloat(“lastFloat”, 0f);
```

Alternatíva: onSaveInstanceState

- Az Activity ennek segítségével Bundle-ba menthet
  - Bundle: String – Parceable párok
- onCreate és onRestoreInstanceState megkapja
- Tipikusan UI-állapot elmentésére
  - …hisz az alkalmazásunk ki lehet söpörve, amikor memória-éhség van
  - DE: adatperzisztálásra ne ezt használjuk, hanem pl. onPause-t
    - onSaveInstanceState nem része az életciklusnak, nem lesz meghívva, ha nincs is esély az alkalmazás visszaállítására

Példa (mentés)

```
@Override
public void onSaveInstanceState(Bundle outState) {
    TextView myTextView = (TextView)findViewById(R.id.myTextView);
    // Save its state
    outState.putString(”TEXTVIEW_STATE_KEY”,
            myTextView.getText().toString());
    super.onSaveInstanceState(outState);
}
```

Példa (töltés)

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    TextView myTextView = (TextView)findViewById(R.id.myTextView);
    String text = “”;
    if (savedInstanceState != null &&
            savedInstanceState.containsKey(”TEXTVIEW_STATE_KEY”))
            text = savedInstanceState.getString(”TEXTVIEW_STATE_KEY”);
    myTextView.setText(text);
}
```

AndroidManifest.xml

- Eddig az Androidos szemléletben tehát
- …vannak Activity-jeink saját életciklussal
- Az alkalmazást leíró fájlban (AndroidManifest.xml) fel kell sorolni az alkalmazásunk Activity-jeit
  - Meg még sokminden mást…
  - Az Android alkalmazás építőkövei:
    - Activity
    - Broadcast Intent Receiver
    - Service
    - Content Provider

Példa

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="hu.bute.helloandroid"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- <uses-permission xmlns:android="http://schemas.android.com/apk/res/android" android:name="android.permission.INTERNET"></uses-permission>
- android:debuggable="true"

Android

Alkalmazás komponensek kommunikációja

Hogyan váltunk az Activity-k között?

- Úgynevezett Intent-ekkel
  - passzív adatstruktúra, az elvárt művelet absztrakt leírásával
- Ez a komponensek közötti késői, futás-idejű kötést valósít meg
- Egy elvárt művelet absztrakt leírásával (vagy egy megtörtént esemény paramétereivel)

Intentek 3 használata:

- Context.startActivity() vagy Activity.startActivityForResult()
- Context.startService()
- Broadcast metódusoknak (ld. később):

  pl. Context.sendBroadcast() (ezt kapják meg a broadcast receiver-ek)

Példa

```
Intent myIntent = new Intent(getApplicationContext(),
    SecondActivity.class);
myIntent.putExtra("MyValue", "Hi there!");
startActivity(myIntent); //már a stacken van?
```

- Ha nincs megnevezve explicit módon: a lehetséges cél Activity-k *intent filter*-eit vizsgálja meg az Intent feloldás folyamata során.

„Task” fogalma az Androidban

- Amit a felhasználó egy Task-ként, futó alkalmazásként él meg
- A root activityvel indul
- További, hívott activityk bárhonnan származhatnak (pl. térkép activity)
- A hívott activity-k egy stack-en
  - back = pop
- Task = activity stack

Intent Action

- Action: egy string, ami leírja az elvárt (vagy broadcastereknél a végbement) akciót
- Néhány „beépített” példa:
- Kb. mint egy függvénynév
  - Határozza meg a paramé-reket
- Saját:

Constant	Target component
ACTION_CALL	activity
ACTION_EDIT	activity
ACTION_MAIN	activity
ACTION_SYNC	activity
ACTION_BATTERY_LOW	broadcast receiver
ACTION_HEADSET_PLUG	broadcast receiver
ACTION_SCREEN_ON	broadcast receiver
ACTION_TIMEZONE_CHANGED	broadcast receiver

"com.example.project.SHOW_COLOR"

Intent Data

- Az adat URI-ja és MIME típusa, pl.
  - ACTION_EDIT esetén egy szerkesztendő dokumentum URI-ja
  - ACTION_CALL esetén „tel:1234”
  - ACTION_VIEW és egy „http://….” ha böngészőt szeretnénk nyitni
- SetType vagy SetDataAndType: a MIME típus explicit megadása (hogy pl. egy file URI-ra képnézőt vagy audioplayert várunk?)
- Content Provider által felügyelt tartalmak (ami a készüléken van): „content://” …: meghatározza a típust.

Intent Category

- Egy sztring további elvárásokkal a kezelő komponenssel szemben, pl.
  - CATEGORY_GADGET: egy Gadget-eket hostoló Activity-be ágyazható
  - CATEGORY_PREFERENCE: beállítás panelt tartalmazó Activity
  - CATEGORY_LAUNCHER: az alkalmazás indító futtathatja

Intent feloldás

- Tipikusan alkalmazások közötti Intent hívásra
- Intent filter: alkalmazás komponensekhez rendelt struktúrák
- Meghatározzák, mely típusú Intenteket tud kezelni
- Filter nélküli komponens csak explicit Intentet tud kezelni
- Az Intent három mezője játszik a feloldásban:
  - Action
  - Data (az URI és a MIME típus is)
  - Category

Intent filter

- Az alkalmazás futtatása nélkül is tudni kell, mit kezel egy adott komponens
  - →AndroidManifest.xml
- Kódból általában csak broadcast receivereket szokás regisztrálni
  - Context.registerReceiver();

Intent filter: Action

- Legalább egy Action kell a filterbe
- Ha az Intent nem definiál Actiont, a teszt sikerül.
  - Ha viszont az Intent filter nem definiál Action-t, semmilyen match nem sikerülhet.

```
<intent-filter . . . >
    <action android:name="com.example.project.SHOW_CURRENT" />
    <action android:name="com.example.project.SHOW_RECENT" />
    <action android:name="com.example.project.SHOW_PENDING" />
    . . .
</intent-filter>
```

Intent filter: Category

- Az Intentben definiált összes Category-nak meg kell felelnie egy elemnek az Intent filterben
- Az implicit hívandó Activity-kben kell lenni egy „android.intent.category.DEFAULT” kategóriának (kivéve MAIN action vagy LAUNCHER category)

```
<intent-filter . . . >
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    . . .
</intent-filter>
```

Intent filter: Data

- Data type (MIME) (pl. „text/*”)
- URI (scheme://host:port/path)

  - Intent URI és type nélkül csak ugyanilyen filterre illeszthető
  - Intent csak URI-val: csak type nélküli, min. egy illeszthető URI-val rendelkező filterre (pl. *mailto:* vagy *tel:*)
  - Intent csak type-al: A filter is csak egy megfelelő type-ot ír elő
  - Intent URI és (esetleg URI-ból implicit) Type adatokkal:
    - Ha a filterben is van megfelelő type és URI is
    - Vagy a filterben nincs URI, de az Intent URI-ja *content*: vagy *file*: .

Nézzünk példát!

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(Menu.NONE, MENUITEM_CONTACTPICKER, Menu.NONE, "Select Contact");
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    if(item.getItemId() == MENUITEM_CONTACTPICKER)
    {
        Uri uri = Uri.parse("content://contacts/people");
        Intent intent = new Intent(Intent.ACTION_PICK, uri);
        startActivityForResult(intent, REQUEST_PICK_CONTACT );
    }
    return super.onOptionsItemSelected(item);
}
```

Példa folyt.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch(requestCode) {
    case  REQUEST_PICK_CONTACT:
        if(resultCode == RESULT_OK)
        {
            mUri   = data.getData();
            //most a kivalasztott contact részletes adatai kellenének
            //ezt meg megirjuk egyszer...
        }
        break;
    }
super.onActivityResult(requestCode, res
}
```

Készüléken az egyes alkalmazások adatbázisait kell elérnünk valahogy.

Ehhez: ContentProvider-ek

Példa folyt.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch(requestCode) {
    case REQUEST_PICK_CONTACT:
        if(resultCode == RESULT_OK)
        {
            mUri  = data.getData();
            //most a kivalasztott contact részletes adatai kellenének
            //ezt meg megirjuk egyszer...
        }
        break;
    }
super.onActivityResult(requestCode, res
}
```

Készüléken az egyes alkalmazások adatbázisait kell elérnünk valahogy.

Ehhez: ContentProvider-ek

Emlékeztek-e még?

- Hogyan történik Android platformon a hardver- vagy nyelvfüggő erőforrások megkülönböztetése, kezelése?

Android

Mobilszoftverek

A3

Fehér Marcell

Feher.Marcell@aut.bme.hu

Az előző rész tartalmából

- Platform bevezetés
- Activity-k és a felhasználói felület
  - Erőforrások
  - Nézetek, UI építőkockák
  - Hardver függetlenség, lokalizáció
- Alkalmazás-komponensek kommunikációja
  - Intentek
  - Intent filterek

Emlékeztek-e még?

- Milyen „leírói” vannak egy Intentnek?

Emlékeztek-e még?

- Milyen „leírói” vannak egy Intentnek?
  - Action
  - Category
  - Data

Tartalom

- Adatok megosztása alkalmazások között
  - ContentProvider mechanizmus
- Adatkötés megvalósítása
  - UI elemek és adat összekötése
- Animációk
- Broadcast Receiver-ek
  - Rendszerszintű események kezelése
- Notification

Breaking news

- Google IO jövő héten
  - Éves Google konferencia
  - Május 10-11.
  - http://www.google.com/events/io/2011/
  - Youtube élő közvetítés
- Az Android átvette a vezetést a Marketben lévő ingyenes alkalmazások számának tekintetében!

Android

Content Provider

Content Provider

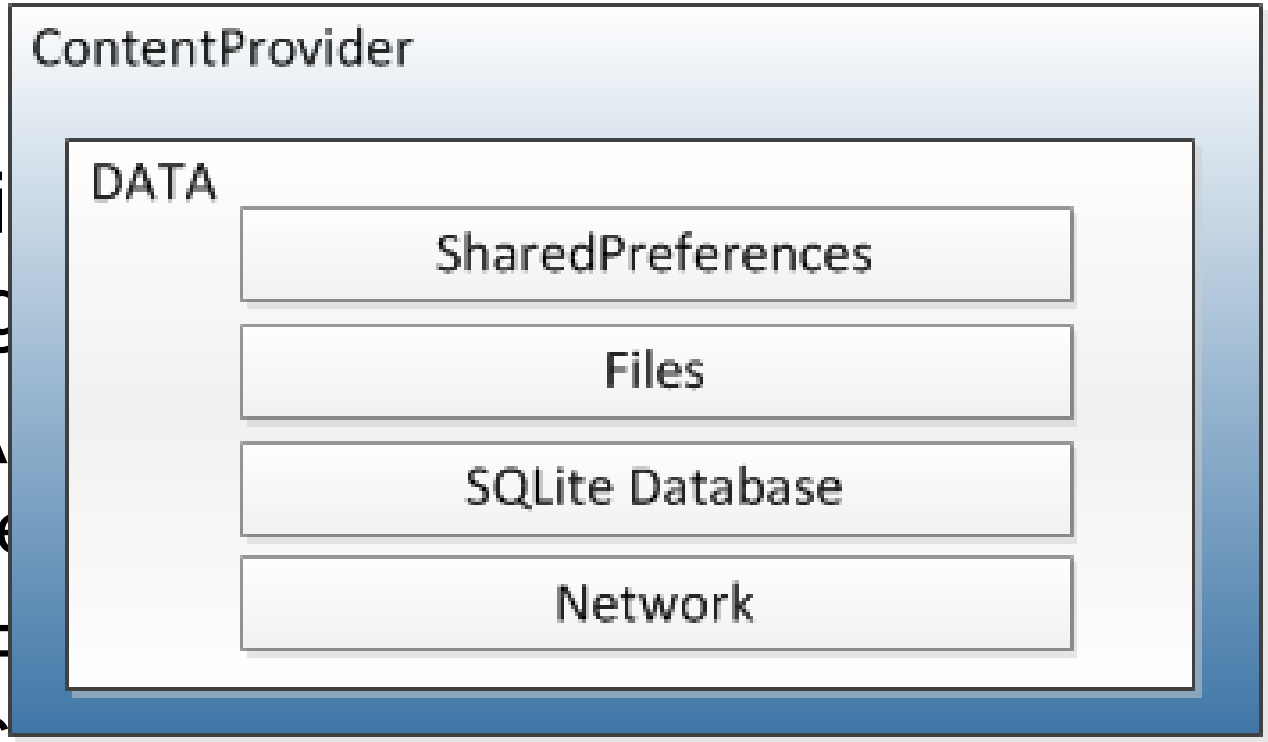
- Csupán csomagoló objektum az adat köré
- Akkor szükséges, ha az alkalmazás meg szeretné osztani az adatokat
- Androidon nincs olyan tároló terület, amelyet minden alkalmazás közösen elér
  - Content Provider mechanizmus a megoldás
- Rengeteg beépített az Androidban
  - személyes adatok, névjegyzék, zenék, videók, képek, hívásnapló, stb…
  - Az olvasásukhoz néha Permission szükséges

Adattárolás

- Konkrét adattárolási módszer: az alkalmazásra bízva
- Ami szükséges: a megfelelő interface megvalósítása
  - Amit aztán ContentResolver objektumon keresztül érnek el az alkalmazások
  - (Providerenként 1 példányt ér el az összes ContentResolver)

Adattárolás

- Konkrét adattárolási módszer: az alkalmazásra bízva
- Ami
  - A
  - (F



Hasonló koncepciók

- Weboldal
  - Minden Content Providert regisztrálni kell, mint domain név (neve: authority)
  - Ezen a címen lesznek elérhetők az adatok amiket szolgáltat

    <provider

    android:name="NotePadProvider„
    android:authorities=„amorg.demo.NotePad"

    />

    Adatai a következő URI-val elérhetők:
    content://amorg.demo.NotePad/

Hasonló koncepciók

- REST
  - REpresentational State Transfer
  - Az erőforrásokhoz egyedi URI tartozik, aminek struktúrája:
    - http://example.com/resources/ef7d-xj36p
  - Content Provideren keresztül elérhető jegyzet:
    - content://amorg.demo.NotePad/Notes/13
    - (content://contacts/people/23)
    - (content://media/external/images)

Hasonló koncepciók

- WebService
  - A Content Provider szolgáltatásként kiajánlja az adatokat
  - Melyek egy URI-n keresztül érhetők el
  - Nem teljesen azonos
    - Nem típusos, mint például a SOAP
    - A hívónak tisztában kell lennie a kapott adat típusával
      - Azért nem olyan rossz a helyzet, az Android biztosít lehetőséget a MIME type kikövetkeztetésére

Hasonló koncepciók

- Tárolt eljárások
  - Szolgáltatás-alapú hozzáférés a relációs adatbázishoz
  - A query rejtve benne van az URI-ban
    - Honnan, mit szeretnénk megkapni
  - Cursor objektummal tér vissza, ami az eredmény(halmaz)ra mutat

Nézzünk példát!

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(Menu.NONE, MENUITEM_CONTACTPICKER , Menu.NONE, "Select Contact");
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    if(item.getItemId() == MENUITEM_CONTACTPICKER)
    {
        Uri uri = Uri.parse("content://contacts/people");
        Intent intent = new Intent(Intent.ACTION_PICK, uri);
        startActivityForResult(intent, REQUEST_PICK_CONTACT );
    }
    return super.onOptionsItemSelected(item);
}
```

Nézzünk példát!

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(Menu.NONE, MENUITEM_CONTACTPICKER, Menu.NONE, "Select Contact");
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    if(item.getItemId() == MENUITEM_CONTACTPICKER)
    {
        Uri uri = Uri.parse("content://contacts/people");
        Intent intent = new Intent(Intent.ACTION_PICK, uri);
        startActivityForResult(intent, REQUEST_PICK_CONTACT );
    }
    return super.onOptionsItemSelected(item);
}
```

Nézzünk példát!

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(Menu.NONE, MENUITEM_CONTACTPICKER, Menu.NONE, "Select Contact");
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    if(item.getItemId() == MENUITEM_CONTACTPICKER)
    {
        Uri uri = Uri.parse("content://contacts/people");
        Intent intent = new Intent(Intent.ACTION_PICK, uri);
        startActivityForResult(intent, REQUEST_PICK_CONTACT );
    }
    return super.onOptionsItemSelected(item);
}
```

Nézzünk példát!

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(Menu.NONE, MENUITEM_CONTACTPICKER, Menu
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    if(item.getItemId() == MENUITEM_CONTACTPICKER)
    {
        Uri uri = Uri.parse("content://contacts/peopl
        Intent intent = new Intent(Intent.ACTION_PICK
        startActivityForResult(intent, REQUEST_PICK_C
    }
    return super.onOptionsItemSelected(item);
}
```

Nézzünk példát!

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(Menu.NONE, MENUITEM_CONTACTPICKER, Menu
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    if(item.getItemId() == MENUITEM_CONTACTPICKER)
    {
        Uri uri = Uri.parse("content://contacts/peopl
        Intent intent = new Intent(Intent.ACTION_PICK
        startActivityForResult(intent, REQUEST_PICK_C
    }
    return super.onOptionsItemSelected(item);
}
```

Példa folyt.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch(requestCode) {
    case  REQUEST_PICK_CONTACT:
        if(resultCode == RESULT_OK)
        {
            mUri   = data.getData();
            //most a kivalasztott contact részletes adatai kellenének
            //ezt meg megirjuk egyszer...
        }
        break;
    }
super.onActivityResult(requestCode, res
}
```

Készüléken az egyes alkalmazások adatbázisait kell elérnünk valahogy.

Ehhez: ContentProvider-ek

Példa folyt.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch(requestCode) {
    case REQUEST_PICK_CONTACT:
        if(resultCode == RESULT_OK)
        {
            mUri  = data.getData();
            //most a kivalasztott contact részletes adatai kellenének
            //ezt meg megirjuk egyszer...
        }
        break;
    }
super.onActivityResult(requestCode, res
}
```

Készüléken az egyes alkalmazások adatbázisait kell elérnünk valahogy.

Ehhez: ContentProvider-ek

Adatok lekérdezése

- A ContentResolver query függvényével
  - startManagingCursor(): Cursor életciklus automatikus kezelésére, pl. requery, ha visszatérünk az app-ba.
- Paraméterek:
  - Uri
  - Oszlopok nevei
  - WHERE klóz + paraméterei
  - Rendezési feltétel

Adatok lekérdezése

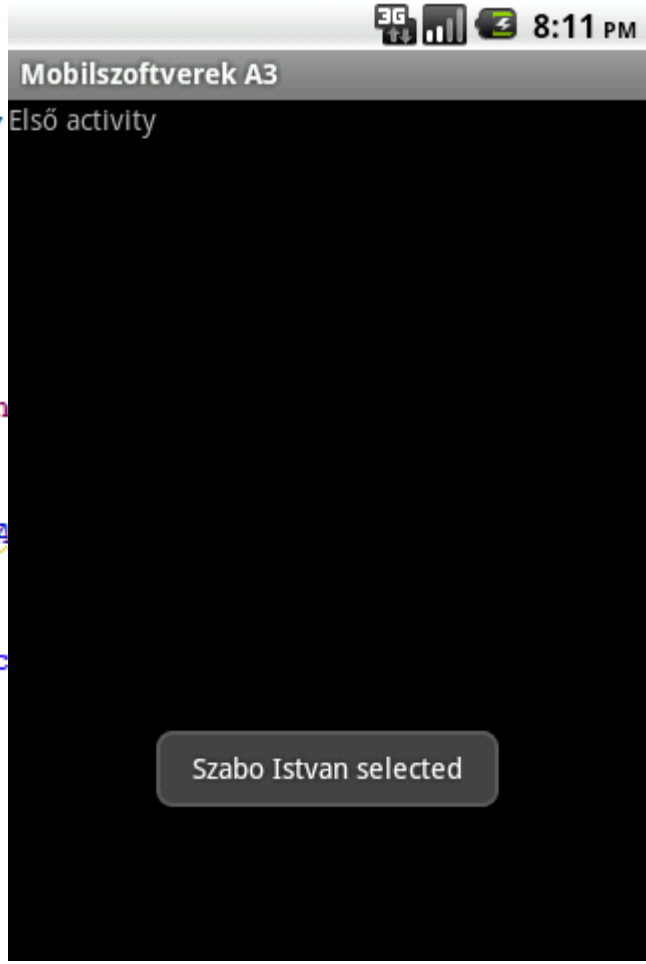
- Visszatérés: Cursor-ral, 0+ sor
- Az adatok csak olvashatóak ezzel
  - moveToFirst
  - moveToNext
  - moveToPrevious
  - getCount
  - getColumnIndexOrThrow
  - getColumnName
  - getColumnNames
  - moveToPosition
  - getPosition

A példa folytatása

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(resultCode == RESULT_CANCELED)
        return;

    switch (requestCode) {
    case REQUEST_CONTACT:
        Uri contactUri = data.getData();
        Cursor cursor = getContentResolver().query(contactUri, null, null, null, null);
        startManagingCursor(cursor);
        if(cursor.moveToFirst()){
            int nameIdx = cursor.getColumnIndexOrThrow(People.NAME);
            String name = cursor.getString(nameIdx);

            Toast.makeText(getApplicationContext(), name+" selected", Toast.LENGTH_LONG).show();
        }

        break;

    default:
        break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

A példa folytatása

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
    if(resultCode == RESULT_CANCELED)
        return;

    switch (requestCode) {
    case REQUEST_CONTACT:
        Uri contactUri = data.getData();
        Cursor cursor = getContentResolver().query(contactUri, n
        startManagingCursor(cursor);
        if(cursor.moveToFirst()){
            int nameIdx = cursor.getColumnIndexOrThrow(People.NA
            String name = cursor.getString(nameIdx);

            Toast.makeText(getApplicationContext(), name+" selec
        }

        break;

    default:
        break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```



Adatmódosítás/beszúrás

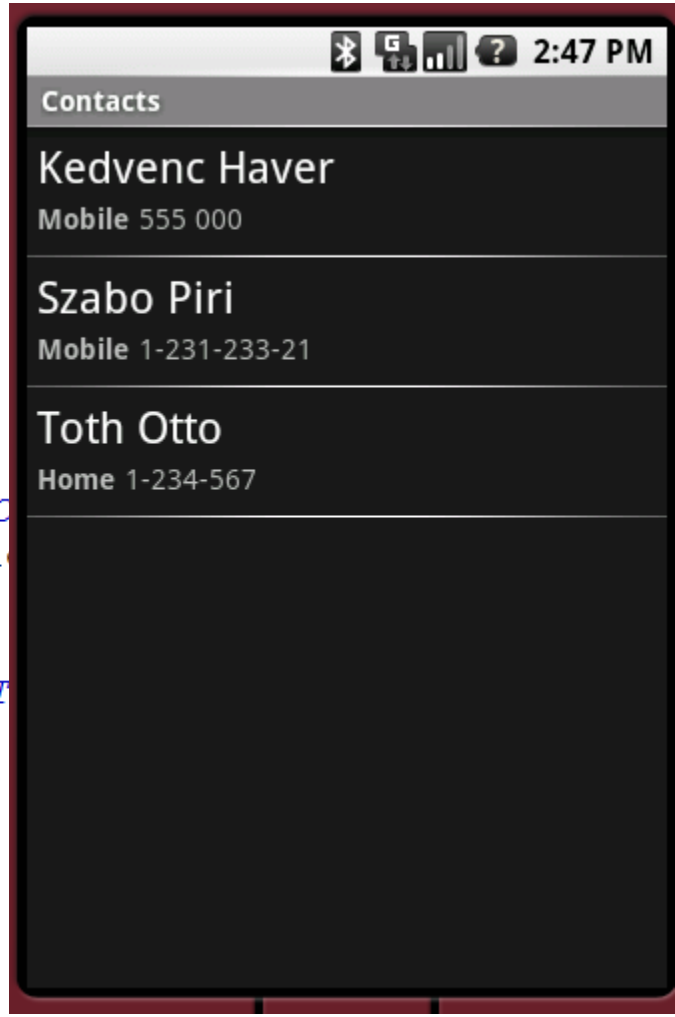
- ContentValues kulcs-érték párok segítségével definiáljuk az elemet
- ContentResolver.insert-tel beletesszük (visszatér az új elem uri-jával
- Pl. új contact, majd telefonszámmal bővítés

Példa

```
if(item.getItemId() == MENUITEM_CONTACTINSERT)
{
      ContentValues values = new ContentValues();
      values.put(People.NAME, "Kedvenc Haver");
      values.put(People.STARRED, 1);
      Uri uri = getContentResolver().insert(People.CONTENT_URI, values);
      Uri phoneUri = Uri.withAppendedPath(uri, People.Phones.CONTENT_DIRECTORY);

      values.clear();
      values.put(People.Phones.TYPE, People.Phones.TYPE_MOBILE);
      values.put(People.Phones.NUMBER, "555 000");
      getContentResolver().insert(phoneUri, values);
}
```

Példa

```
if(item.getItemId() == MENUITEM_CONTACTINSERT)
{
      ContentValues values = new ContentValues();
      values.put(People.NAME, "Kedvenc Haver");
      values.put(People.STARRED, 1);
      Uri uri = getContentResolver().insert(People.C
      Uri phoneUri = Uri.withAppendedPath(uri, Peopl

      values.clear();
      values.put(People.Phones.TYPE, People.Phones.T
      values.put(People.Phones.NUMBER, "555 000");
      getContentResolver().insert(phoneUri, values);
}
```



Saját provider írása

- ContentProvider osztályból származtatás
  - Az onCreate-ben megnyitni az adatokat
- A megvalósításnak thread-safe-nek kell lenni (hisz többen is használhatják)
- Nem kötelező, de szokásos definiálni:
  - public static final Uri CONTENT_URI
  - (a ContentProvider osztály qualified nevével)
  - „táblánként” eltérő path, ha nem csak 1 van

Saját provider írása

- ContentProvider osztályból származtatás
  - Az onCreate-ben megnyitni az adatokat
- A megvalósításnak thread-safe-nek kell lenni (hisz többen is használhatják)
- Nem kötelező, de szokásos definiálni:

```
public static final Uri CONTENT_URI =
                Uri.parse("content://com.example.codelab.transporationprovider");
```

  - (a ContentProvider osztály qualified nevével)
  - „táblánként” eltérő path, ha nem csak 1 van

Saját provider írása

- ContentProvider osztályból származtatás
  - Az onCreate-ben megnyitni az adatokat
- A megvalósításnak thread-safe-nek kell lenni (hisz többen is használhatják)
- Nem kötelező, de szokásos definiálni:

```
public static final Uri CONTENT_URI =
                Uri.parse("content://com.example.codelab.transporationprovider");
```

```
content://com.example.codelab.transporationprovider/train
content://com.example.codelab.transporationprovider/air/domestic
content://com.example.codelab.transporationprovider/air/international
```

Saját provider írása

- A provider által visszaadandó oszlopneveket szintén public static String-be
- Legyen _ID konstanssal egy _id oszlop, az adatsorok megkülönböztetésére
  - SQLite tábla esetén INTEGER PRIMARY KEY AUTOINCREMENT oszlop
- A GetType(Uri) felüldefiniálása

Saját provider írása

- A provider által visszaadandó oszlopneveket szintén public static String-be
- Legyen _ID konstanssal egy _id oszlop, az adatsorok megkülönböztetésére
  - SQLite-tábla esetén INTEGER PRIMARY KEY AUTOINCREMENT oszlop
- A getType(Uri) feliüldefiniálása

• Ha az Uri egy itemre vonatkozik:
content://com.example.transportationprovider/trains/122
• Akkor a MIME pl:
vnd.android.cursor.item/vnd.example.rail

• Ha az Uri több elemre vonatkozik:
content://com.example.transportationprovider/trains
• Akkor a MIME:
vnd.android.cursor.dir/vnd.example.rail

Saját provider írása

- Felüldefiniálandó függvények:

  query()
  insert()
  update()
  delete()
  getType()

- Manifest.xml-ben be kell jegyezni a providerünket
  - A name attribútum a provide osztály minősített neve
  - Az authorities attribútum a content:// uri része (path nélkül!)

Saját provider írása

- Felüldefiniálandó függvények:
  - query()
  - insert()
  - update()
  - delete()
  - getType()

```
<provider
    name="com.example.transportationprovider.TransportationProvider"
    authorities="com.example.transportationprovider” . . . />
</provider>
```

DE:
com.example.transportationprovider/trains/ nem kell!

- Manifest.xml-ben be kell jegyezni a providerünket
  - A name attribútum a provide osztály minősített neve
  - Az authorities attribútum a content:// uri része (path nélkül!)

PótZH infó

- A ZH pótlása 2011.05.18-án 14:00 órakor lesz, az IB026 teremben.
- A pótZH-t csak az írhatja meg, aki erre az alkalomra a Neptunon feljelentkezik!!! (A Neptunban az alkalom Konzultáció néven szerepel)

Android

Adatkötés

Vissza az User Interface-hez

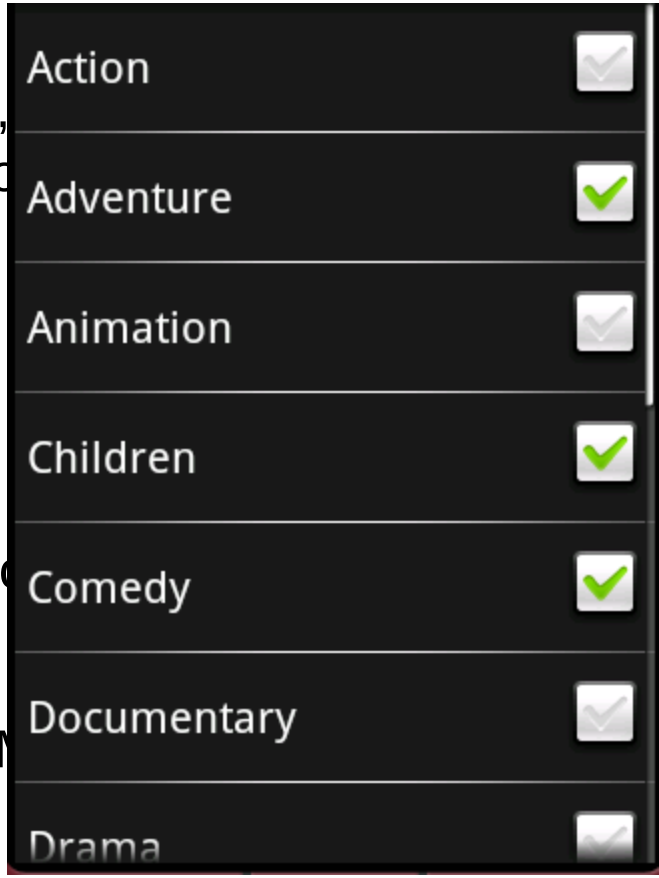
- Készítsünk adatkötött elemet: AdapterView leszármazottak
  - Pl. ListView, Gallery, Spinner, GridView, …
- Ez egy olyan ViewGroup, amely a gyermek elemeit egy adaptertől fogja kapni
  - Az Adapter adja az adatokat
  - És felelős az egyes adatok View-jának kirajzolásáért is!
- Például ArrayAdapter, CursorAdapter

Példa: ArrayAdapter

```
private static final String[] ITEMS= new String[] {
    "Action", "Adventure", "Animation", "Children", "Comedy",
    "Documentary", "Drama",       "Foreign", "History", "Independent",
    "Romance", "Sci-Fi", "Television", "Thriller"
};
final ArrayAdapter<String> aa;
aa = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_multiple_choice, ITEMS);
myListView.setAdapter(aa);
myListView. setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
```

Példa: ArrayAdapter

```
private static final String[] ITEMS= new String[] {
    "Action", "Adventure", "Animation", "Children",
   "Documentary", "Drama",       "Foreign", "Histo
   "Romance", "Sci-Fi", "Television", "Thriller"
};
final ArrayAdapter<String> aa;
aa = new ArrayAdapter<String>(this,
   android.R.layout.simple_list_item_multiple_ch
myListView.setAdapter(aa);
myListView. setChoiceMode(ListView.CHOICE_M
```

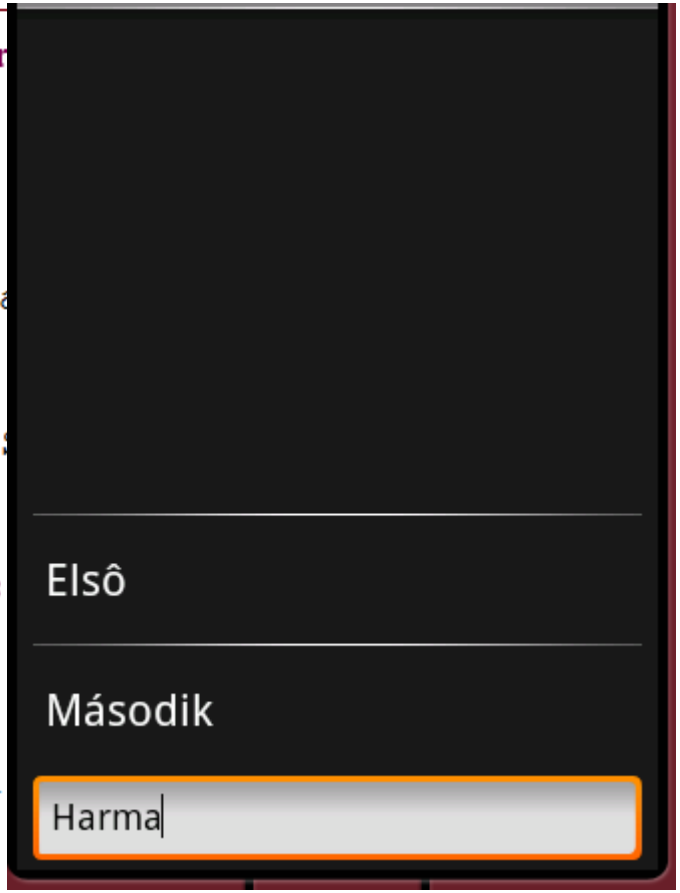


Példa 2.

```
public class Transcripter extends ListActivity implements OnClickListener,
OnKeyListener {

    private EditText mUserText;
    private ArrayAdapter<String> mAdapter;
    private ArrayList<String> mStrings = new ArrayList<String>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mylayout);
        mAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, mStrings);
        setListAdapter(mAdapter);

        mUserText = (EditText) findViewById(R.id.userText);
        mUserText.setOnClickListener(this);
        mUserText.setOnKeyListener(this);
    }
```

Példa 2.

```
public class Transcripter extends ListActivity im
OnKeyListener {

    private EditText mUserText;
    private ArrayAdapter<String> mAdapter;
    private ArrayList<String> mStrings = new Arra

    @Override
    protected void onCreate(Bundle savedInstance
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mylayout);
        mAdapter = new ArrayAdapter<String>(this
            android.R.layout.simple_list_item_1,
        setListAdapter(mAdapter);

        mUserText = (EditText) findViewById(R.id
        mUserText.setOnClickListener(this);
        mUserText.setOnKeyListener(this);
    }
```



Példa 2. folyt.

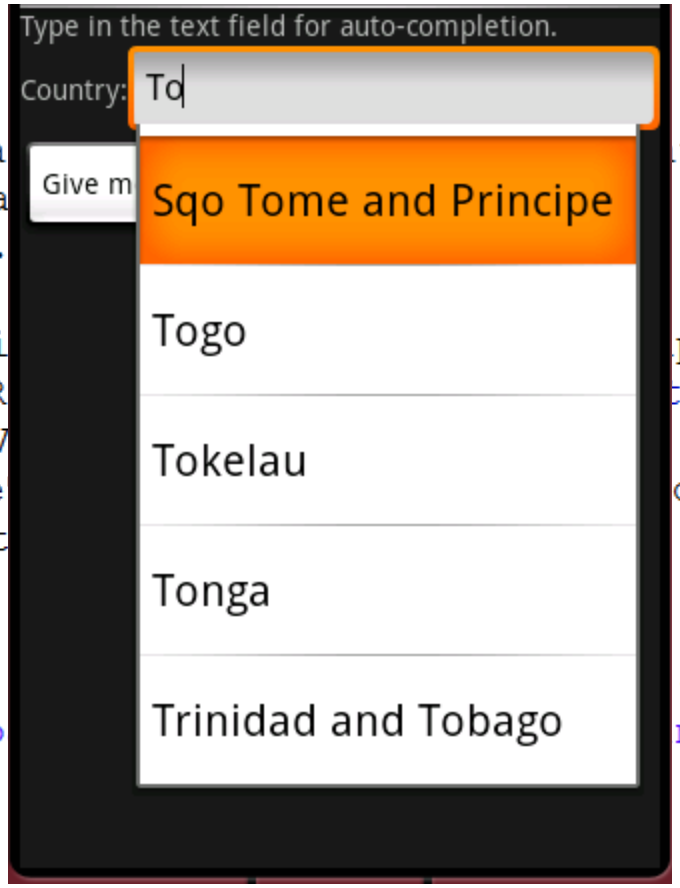
```
public void onClick(View v) {
    sendText();
}

private void sendText() {
    String text = mUserText.getText().toString();
    mAdapter.add(text);
    mUserText.setText(null);
}

public boolean onKey(View v, int keyCode, KeyEvent event) {
    if (event.getAction() == KeyEvent.ACTION_DOWN) {
        switch (keyCode) {
            case KeyEvent.KEYCODE_DPAD_CENTER:
            case KeyEvent.KEYCODE_ENTER:
                sendText();
                return true;
        }
    }
    return false;
}
```

Tipikusan Google-s:Autocomplete

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.mylayout);

    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, COUNTRIES);
    AutoCompleteTextView textView =
        (AutoCompleteTextView) findViewById(R.id.edit);
    textView.setAdapter(adapter);
}

static final String[] COUNTRIES = new String[] {
  "Afghanistan", "Albania", "Algeria", "American Samoa", "Andorra",
.....
```

Tipikusan Google-s:Autocomplete

```
@Override
protected void onCrea                                    te) {
    super.onCreate(sa
    setContentView(R.

    ArrayAdapter<Stri                                    pter<String>(this,
            android.R                                    tem_1line, COUNTRIES);
    AutoCompleteTextV
        (AutoComplete                                    d.edit);
    textView.setAdapt
}

static final String[]                                    {
  "Afghanistan", "Alb                                    n Samoa", "Andorra",
.....
```



Android

Animáció

Animáció

- Kétféle:
  - Tweened animation: View-k forgatása, mozgatása, nyújtása, halványítása
  - Frame-by-frame animáció (képsorozat)
- Animáció létrehozása kódból vagy XML fájllal (/res/anim)

Tweened animation

- Típusok
  - Alpha
  - Scale
  - Translate
  - Rotate
- Set-be szervezhetőek
- Paraméterek
  - Duration (ms)
  - startOffset (mikor kezdje, ms)
  - fillBefore
  - fillAfter
    - (Kezdés előtt, illetve után alkalmazza-e az animációt)
  - Interpolator
    - A sebesség változtatásához

Példa: splash screen

```
private void splash() {
        setContentView(R.layout.splash);
        ImageView splashview =
                (ImageView) findViewById(R.id.splash_image);
        splashview.setImageBitmap(BitmapFactory.decodeResource(
                getResources(), R.drawable.splash));
        AnimationSet fadeAnimation = new AnimationSet(true);
        AlphaAnimation alphaIn = new AlphaAnimation(0, 1);
        alphaIn.setFillAfter(true);
        alphaIn.setDuration(1000);
        alphaIn.setStartOffset(0);
        fadeAnimation.addAnimation(alphaIn);
        AlphaAnimation alphaOut = new AlphaAnimation(1, 0);
        alphaOut.setFillAfter(true);
        alphaOut.setDuration(1000);
        alphaOut.setStartOffset(4000);
        fadeAnimation.addAnimation(alphaOut);
        splashview.startAnimation(fadeAnimation);
        fadeAnimation.setAnimationListener(this);
}
```

Példa: splash screen

```
private
        s
        I

        s

        A
        A
        a
        a
        a
        f
        A
        a
        al
        alphaAnimation.setDuration(1000);
        fadeAnimation.addAnimation(alphaOut);
        splashview.startAnimation(fadeAnimation);
        fadeAnimation.setAnimationListener(this);
}
```

```
…implements AnimationListener…

@Override
public void onAnimationEnd(Animation animation) {
        //splash vége, folyt…
        continueApp();
}

@Override
public void onAnimationRepeat(Animation animation) {

}

@Override
public void onAnimationStart(Animation animation) {

}
```

Példa: frame animáció

```
<animation-list
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:oneshot="false">
  <item android:drawable="@drawable/rocket1" android:duration="500" />
  <item android:drawable="@drawable/rocket2" android:duration="500" />
  <item android:drawable="@drawable/rocket3" android:duration="500" />
</animation-list>
```

Android

Broadcast Receiver

Broadcast Receiver

- Reagálhatunk globális eseményekre
- A Broadcast Receiver-ek intent filterekben deklarálják, mit kívánnak kezelni
  - Normál broadcastokra (Context.sendBroadcast) az összes lefut, véletlen sorrendben
  - Sorrendezett broadcast (sendOrderedBroadcast): egyszerre csak egy fut, lehetnek prioritások, leállíthatja a további propagálást, illetve küldhet választ

BroadcastReceiver objektum

- onReceive(Context, Intent) felüldeklarálni
- Csak a függvény futása alatt él! (ne indítsunk szálakat stb.)
- Ne dobjunk fel dialógust!
  - Használjuk a NotificationManager lehetőségeit! (ld. később)
- 5 másodperc alatt végeznie kell
  - Különben „Application Nonresponding”

Regisztrálás

```
<receiver android:name=”.myBroadcastReceiver”>
    <intent-filter>
      <action
        android:name=”hu.bute.action.GO_ON_STRIKE”/>
    </intent-filter>
</receiver>
```

---------- kódból: ----------

```
IntentFilter filter = new IntentFilter(GO_ON_STRIKE);
myBroadcastReceiver r = new myBroadcastReceiver();
registerReceiver(r, filter);
//unregisterReceiver(r);
```

Néhány broadcast a rendszertől

- **ACTION_AIRPLANE_MODE_CHANGED**
- **ACTION_BATTERY_LOW**
- **ACTION_BOOT_COMPLETED**
- **ACTION_CAMERA_BUTTON**
- **ACTION_DATE_CHANGED**
- **ACTION_DEVICE_STORAGE_LOW**
- **ACTION_HEADSET_PLUG**
- **ACTION_NEW_OUTGOING_CALL**
- **ACTION_PACKAGE_ADDED**
- **ACTION_SCREEN_OFF**
- **ACTION_SCREEN_ON**
- **ACTION_TIMEZONE_CHANGED**
- **ACTION_TIME_CHANGED**
-

Android

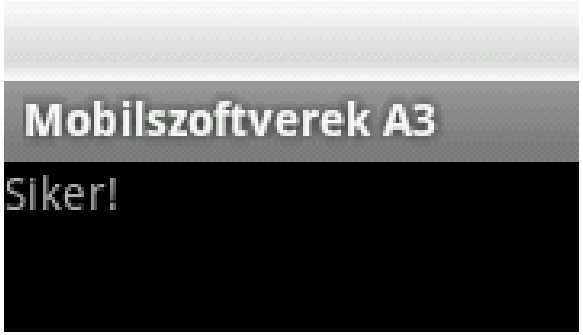
Felhasználó értesítése

Értesítési módok

- Activity
  - Nem túl elegáns
- Dialógusablak
- Toast
- **Notification**

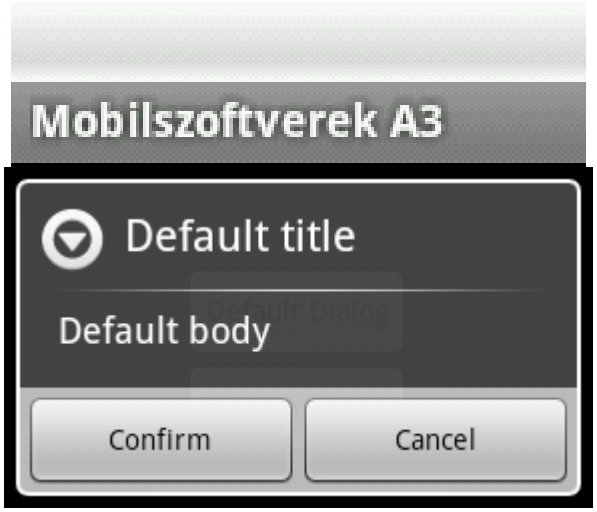
Értesítési módok

- Activity
  - Nem túl elegáns
- Dialógusablak
- Toast
- **Notification**



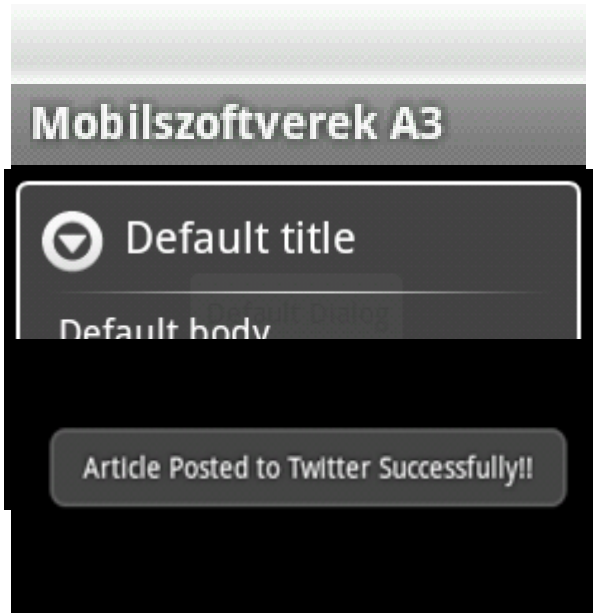
Értesítési módok

- Activity
  - Nem túl elegáns
- Dialógusablak
- Toast
- **Notification**



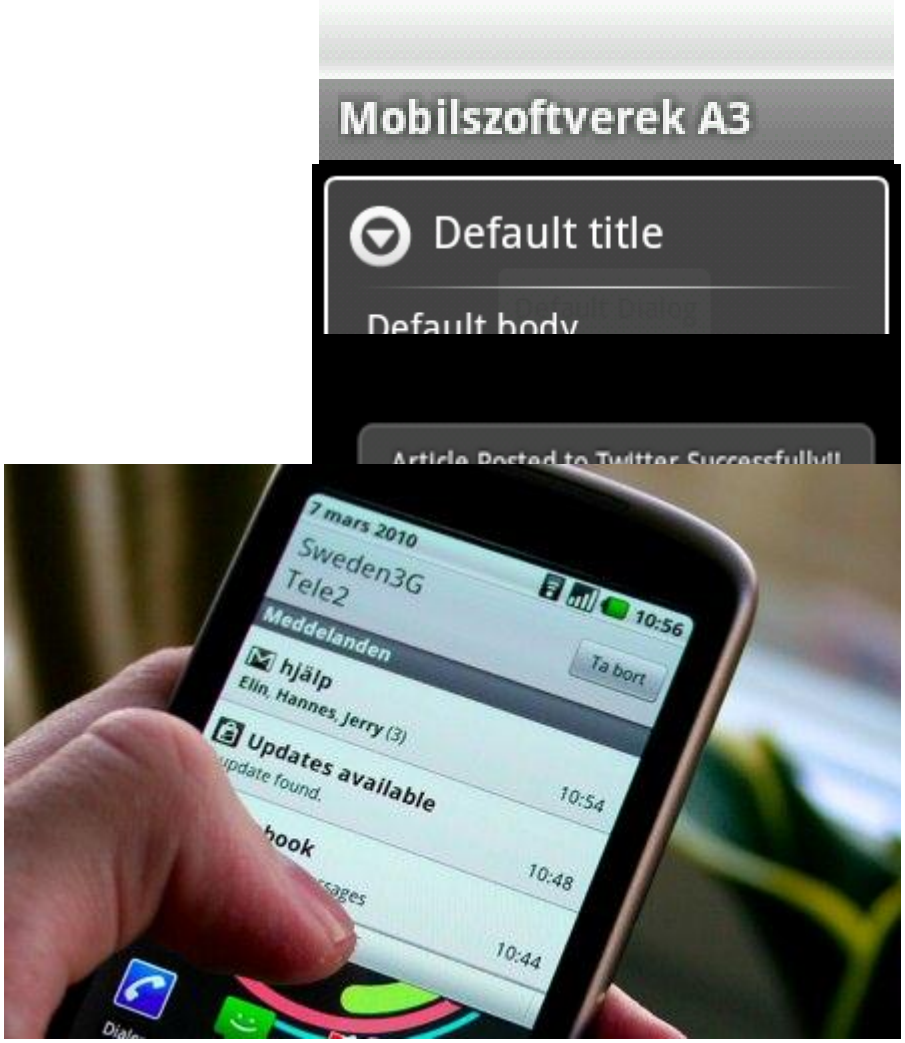
Értesítési módok

- Activity
  - Nem túl elegáns
- Dialógusablak
- Toast
- **Notification**



Értesítési módok

- Activity
  - Nem túl elegáns
- Dialógusablak
- Toast
- **Notification**



Notification-ök

- Activity indítása nélkül tájékoztatja a felhasználót
  - Service-k, broadcast receiverek, inaktív Activity-k, amelyek a felhasználó figyelmét igénylik
- Képességei:
  - Status bar ikon
  - Extended status bar info (és Intent indítás)
  - LEDek villogtatása
  - Vibrálás
  - Hanglejátszás

Notification-ök

- Activity indítása nélkül tájékoztatj
  - Service-k, broadcast receiverek, ina
    amelyek a felhasználó figyelmét igé
- Képességei:
  - Status bar ikon
  - Extended status bar info (és Intent i
  - LEDek villogtatása
  - Vibrálás
  - Hanglejátszás

Notification használat

- Mivel rendszer szolgáltatás, szerezzünk rá referenciát:

```
String svcName = Context.NOTIFICATION_SERVICE;
NotificationManager notificationManager;
notificationManager =
    (NotificationManager)getSystemService(svcName);
```

Notification készítése

- 1: Notification objektum, ikon, rövid scrollozandó üzenet, időzítés

```
int icon = R.drawable.icon;

String tickerText = “Showing text to the user”;

// Az extended status bar-on levő sorrend kialakításához

long when = System.currentTimeMillis();

Notification notification = new Notification(icon, tickerText,
    when);
```

Notification 2. lépés

- A felhasználó megnézheti, mi történt
  - extended status bar
- Itt a címen, dátumon és ikonon túl részletes magyarázat is megjeleníthető
- Beállíthatunk egy Intent-et, ami az eseményre klikkeléskor sül el (PendingIntent)

Notification 2. lépés

```
Context context = getApplicationContext();
String expandedText = “Here I explain why to bother the user”;
String expandedTitle = “The title for the expanded view”;
Intent intent = new Intent(this, MyActivity.class);
//statikus fv-nyel szerzünk PendingIntentet (context, req. code, intent, flags)
PendingIntent launchIntent = PendingIntent.getActivity(context, 0, intent, 0);
notification.setLatestEventInfo(context, expandedTitle, expandedText,
    launchIntent);
```

Notification trigger

- Triggerelés:

  notificationManager.notify(notificationRef, notification);

- notificationRef: a notification visszavonásához, update-léséhez stb.
  - notificationManager.cancel(notificationRef);
  - Update: notify új hívása azonos ref-fel
    - Az objektum lehet új

Notification update

- Ha újabb esemény jön: setLatestEventInfo új paraméterekkel
  - Pl. ikon változtatása, ha nemcsak 1 bejövő új üzenet van, hanem több is
- Az események számát a number tagváltozóban lehet jelezni
  - notification.number++;

További lehetőségek

- Vibrálás
  - <uses-permission android:name="android.permission.VIBRATE"/>
- Egy long tömbbel megadható a vibráció mintája

```
long[] vibrate = new long[] { 1000, 1000, 1000, 1000,
    1000 };

notification.vibrate = vibrate;
```

Notification flag-ek

- Folyamatban levő eseményre:
  - notification.flags = notification.flags | Notification.FLAG_ONGOING_EVENT;
  - Az extended view-ban szeparálva
  - Pl. telefonhívás
- „Kitartó” eseményre
  - notification.flags = notification.flags | Notification.FLAG_INSISTENT;
  - Cancel hívásig minden effekttel (vibrálás, villogás, ...)
  - Pl. ébresztő

Emlékeztek-e még?

- Milyen „beépített” broadcast intent-ek vannak?
- Mi az a kétféle animációtípus, amit az Android standard UI támogat?
- Milyen módokon tud a Notification jelezni a felhasználónak?

Android

Mobilszoftverek

A5

Fehér Marcell

Feher.Marcell@aut.bme.hu

Az előző rész tartalmából

- Content Provider-ek
  - Lekérdezés, készítés
- Adatkötés megvalósítása
- Animációk
- Broadcast Receiver-ek
- Notification-ök

Emlékeztek-e még?

- Milyen broadcast intenteket küld a rendszer?

Emlékeztek-e még?

- Milyen broadcast intenteket küld a rendszer?

Tartalom

- Location-based services
  - Provider-ek
  - Proximity alert
- Geocoding
- MapActivity, MapView
  - Overlay
  - ItemizedOverlay
  - View-k rögzítése

Location based services

- A telefon hordozható → mindig máshol van ☺
- A fizikai helytől függő szolgáltatásokat lehet nyújtani
  - Navigáció
  - Üzelti alkalamzás: pl. űrlapkitöltés, flottakövetés
  - Szabadidős alkalmazás: pl. cityguide, teleauto
  - Találkozószervezés rutinok alapján

A készülék pozíciója: komponensek

- Location Manager
  - Aktuális hely adatok lekérdezése
  - Mozgás adatok
  - „Közelségi” (Proximity) riasztások
- Location Provider
  - Durvább: cellatornyok alapján
  - Pontosabb: GPS
- Engedélyek
  - <uses-permission android:name=”android.permission.ACCESS_FINE_LOCATION”/>
  - <uses-permission android:name=”android.permission.ACCESS_COARSE_LOCATION”/>

Providerek

- LocationManager lm =
  (LocationManager) getSystemService(LOCATION_SERVICE);
- A providerek lekérdezése:

  List<String> providers = locationManager.getProviders(true);
- Kiválasztás pl:

  locationManager.getProvider(LocationManager.GPS_PROVIDER);
- Az egyes providerek eltérnek:
  - Pontosság
  - Visszaadott adatok (magasság? sebesség?)
  - Költség
  - Energiaigény

Provider kiválasztása

- Kritériumok alapján

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);
//csak az enabled providereket
String bestProvider = locationManager.getBestProvider(criteria, true);
```

- Kritériumok lazítása, ha nincs megfelelő (energia, pontosság, majd a többi.)
  - A költségek nem!

Helyszín megismerése

```
Location location = locationManager.getLastKnownLocation(provider);
    if (location != null) {
        double lat = location.getLatitude();
        double lng = location.getLongitude();
    }
```

- Ez nem kéri meg a providereket, hogy frissítsék az adataikat!
- Visszakapjuk a mérés dátumát.

Helyszín változás

```
locationManager.requestLocationUpdates(provider, time, distance,
    new LocationListener() {
        public void onLocationChanged(Location location) {}
        public void onProviderDisabled(String provider){}
        public void onProviderEnabled(String provider){}
        public void onStatusChanged(String provider, int status, Bundle
    extras){}
});
```

- Ha a beállított idő (timer) eltelik, vagy mozgás történik (distance, m)
- locationManager.removeUpdates(myLocationListener);

Proximity Alert-ek

- Gyakran szükség lehet rájuk
- Egy adott helyszín megközelítésekor, vagy elhagyásakor jeleznek PendingIntent-tel
- Intelligensen választja ki a providereket
- Ehhez kell: egy koordináta és egy sugár
- Lejárhat az alert (ha a timeout nem -1)

Proximity Alert

```
Intent intent = new Intent(ACTION_PROXIMITY_ALERT);

PendingIntent pintent = PendingIntent.getBroadcast(this, requestcode,
    intent, flags);

locationManager.addProximityAlert(lat, long, radius, timeout, pIntent);

……..

public class ProximityIntentReceiver extends BroadcastReceiver {
    @Override
    public void onReceive (Context context, Intent intent) {
        String key = LocationManager.KEY_PROXIMITY_ENTERING;
        Boolean entering = intent.getBooleanExtra(key, false);
        ....
    }
}
```

Geocoding

- Postacímről – földrajzi koordinátákra
- Inverz geokóding: fordítva
- Locale: a nyelv, illetve szokásos terület (ország) beállításához
  - Locale.getDefault();
- Visszatérésük: Address objektumok listájával
  - Lat, long, esetleg ország, város, utca, házszám…
- Szinkron hívások!
  - Tegyük külön szálba

Geocoding

- Címből koordináták
- A cím lehet irányítószám, város, utca, házszám, esetleg landmark-ok (pl. múzeum, vasútállomás, …)

```
Geocoder geocoder = new Geocoder(this, Locale.HU);
String streetAddress = “Magyar tudósok körútja 2, Budapest”;
List<Address> locations = null;
locations = Geocoder.getFromLocationName(streetAddress,
    maxNumOfResults);
```

- Megadható egy keresési „téglalap” is (pl. ami a képernyőn épp látható)

Reverse geocoding

- Koordinátákból postacím

```
location = locationManager.getLastKnownLocation(
    LocationManager.GPS_PROVIDER);

double latitude = location.getLatitude();

double longitude = location.getLongitude();

Geocoder gc = new Geocoder(this, Locale.getDefault());

List<Address> addresses = null;

addresses = gc.getFromLocation(latitude, longitude, 10);
```

MapView

- Földrajzi pozíciók megjelenítése térkép-szerűen
- Teljes kontroll a megjelenítés felett
  - Helyszín, nagyítási szint
  - Térkép, műhold, traffic
- Ráhelyezhetőek overlay-ek
- Megjeleníthetőek rajta tetszőleges POIk


15:25
Map
Messages
Friends
Search
Settings
Integration
Logged in!
COPI Android | BME AAIT 2011
Petőfi híd
Egyetemisták parkja
Infopark
Október 23. utca
Bocskai út
Lágymányosi híd
Palace of Arts
Könyves Kálmán körút
Haller utca
Dombóvári út

Térkép nézet készítése

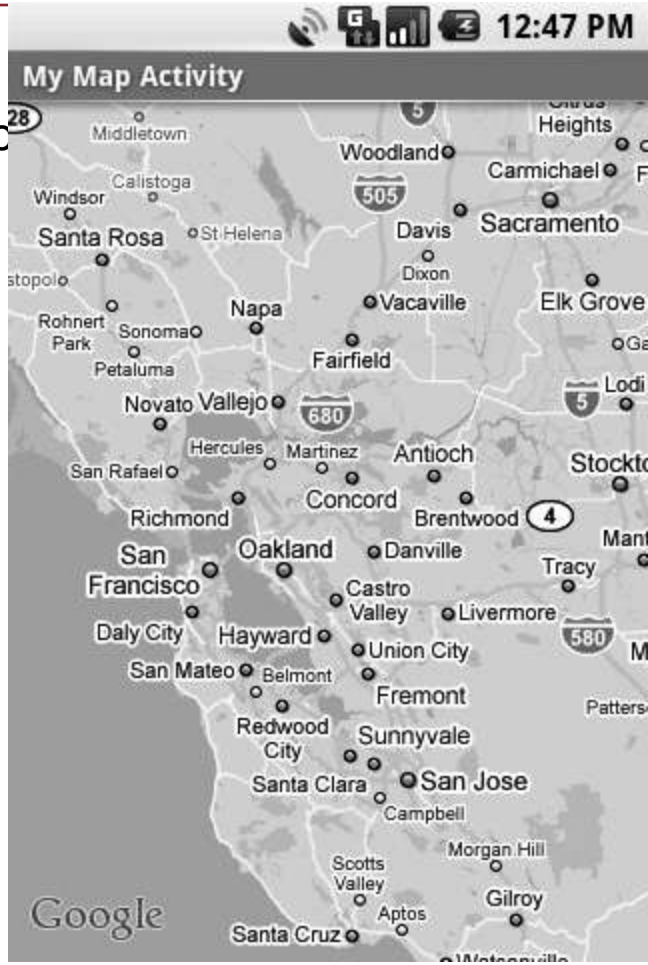
- Az Activity-nk a MapActivity-ből származzon
  - Életciklus kezelés, pl. mivel a netről töltődik a kép
- A layout-ba helyezzünk egy MapView-t
  - Lehetőleg alkalmazásonként egy MapActivity és MapView legyen csak
- A térkép letöltése darabonként, on-demand történik, tehát Internet permission-re szükség van.
- protected boolean isRouteDisplayed() felüldefiniálása

layout

```
<LinearLayout
xmlns:android=”http://schemas.android.com/apk/res/android”
android:orientation=”vertical”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent”>
    <com.google.android.maps.MapView
    android:id=”@+id/map_view”
    android:layout_width=”fill_parent”
    android:layout_height=”fill_parent”
    android:enabled=”true”
    android:clickable=”true”
    android:apiKey=”mymapapikey”
    />
</LinearLayout>
```

layout

```
<LinearLayout
xmlns:android=”http://schemas.android.com/apk/res/andrc
android:orientation=”vertical”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent”>
    <com.google.android.maps.MapView
    android:id=”@+id/map_view”
    android:layout_width=”fill_parent”
    android:layout_height=”fill_parent”
    android:enabled=”true”
    android:clickable=”true”
    android:apiKey=”mymapapikey”
    />
</LinearLayout>
```



MapView használata

- Az egyes rétegek beállítása
  - mapView.setSatellite(true);
  - mapView.setStreetView(true);
  - mapView.setTraffic(true);
- Lekérdezhető:
  - Középpont
  - Aktuális és max zoom
  - A képernyőn mutatott terület adatai (pl. geocodinghoz hasznos)

Zoom kontrol hozzáadás

```
MapView.LayoutParams lp;

lp = new MapView.LayoutParams(
    MapView.LayoutParams.WRAP_CONTENT,
    MapView.LayoutParams.WRAP_CONTENT,
    hovaX, hovaY, MapView.LayoutParams.TOP_LEFT);

View zoomControls = mapView.getZoomControls();

mapView.addView(zoomControls, lp);

mapView.displayZoomControls(true);
```

Térkép vezérlése

- MapController osztállyal
  - MapController mapController = myMapView.getController();
- A térkép pozíciók GeoPoint objektumokkal reprezentálva, mikrodegree
  - mapController.setCenter(myGeoPoint); //ugrik
  - mapController.animateTo(point); //repül
  - mapController.setZoom(1); //1-21, 1 a legtávolabbi

Overlay-ek készítése

- Egy térképhez akárhány adható
- Bármit rajzolhatunk rájuk, kezelik a klikkeléseket
- Felüldefiniálandó a rajzolást és az érintés eseményeket kezelő réteg
- A földrajzi koordinátákról a képernyő koordinátáira történő leképzést a Projection osztály segíti
  - GeoPoint ←→Point
- Saját pozícióhoz külön MyLocationOverlay

Overlay-ek készítése

```
@Override
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
}
@Override
public boolean onTap(GeoPoint point, MapView mapView) {
// Return true if screen tap is handled by this overlay
return false;
}
//---------- szinkronizált műveletek...
List<Overlay> overlays = mapView.getOverlays();
MyOverlay myOverlay = new MyOverlay();
overlays.add(myOverlay);
mapView.postInvalidate();
```

ItemizedOverlay

- POI-k automatikus megjelenítésére
- A POI-kat OverlayItem-leszármazottak reprezentálják
  - Az ItemizedOverlay OverlayItem-ekkel specializált template osztály
- A konstruktorában meg kell hívni a populate() függvényt, ha minden pont rendelkezésre áll
- A POIk számát a size() függvényben kell visszaadnunk
- Majd a createItem függvény készíti el a POIkat

CreateItem…

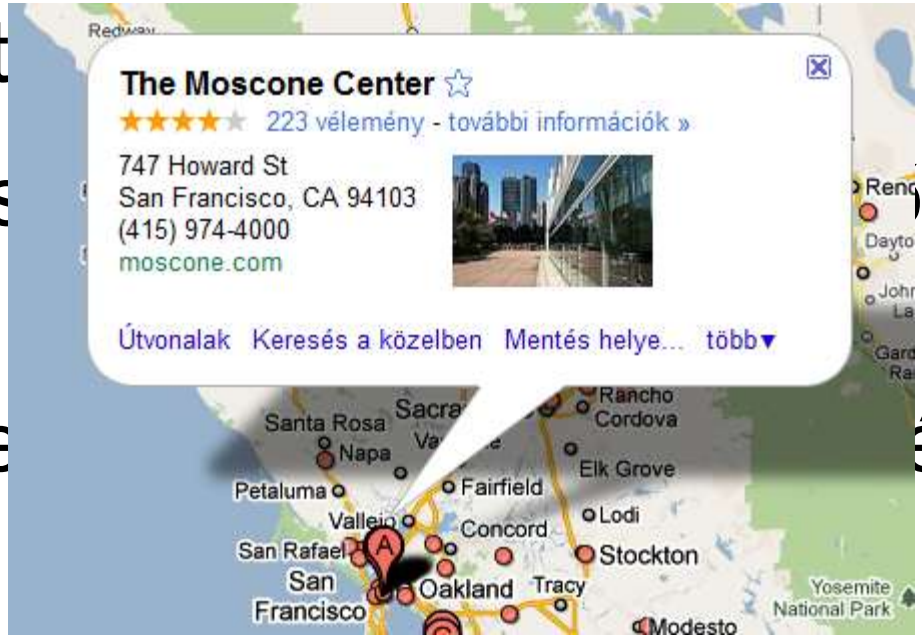
```
@Override
protected OverlayItem createItem(int index) {
      switch (index) {
          case 0:
          Double lat = 42.144578*1E6;
          Double lng = -80.477443*1E6;
          GeoPoint point = new GeoPoint(lat.intValue(), lng.intValue());
          OverlayItem oi;
          oi = new OverlayItem(point, “POI”, “POI szövege”);
          return oi;
      }
return null;
}
```

View rögzítése

- Képernyőponthoz vagy földrajzi ponthoz rögzíthető bármilyen View
- Erőforrásigényesebb, mint az előző techikák
- Pl. kifejtett POI buborékok rögzítésére

View rögzítése

- Képernyőponthoz vagy földrajzi ponthoz rögzíthet
- Erőforrás ző techikák
- Pl. kifejte ésére



View rögzítése - példa

```
Layoutparams = new MapView.LayoutParams(
    MapView.LayoutParams.WRAP_CONTENT,
    MapView.LayoutParams.WRAP_CONTENT,
    x, y, //képernyőponthoz kötve
    MapView.LayoutParams.TOP_LEFT);
```

- Hozzáadás a mapView-hoz (pl. onCreate()-ben):

```
mapView.addView(editText1, screenLP);
```

- Eltűntetés removeView-val

View rögzítése - példa

```
Layoutparams = new MapView.LayoutParams(
    MapView.LayoutParams.WRAP_CONTENT,
    MapView.LayoutParams.WRAP_CONTENT,
    myGeoPoint //földrajzi ponthoz kötve
    MapView.LayoutParams.TOP_LEFT);
```

- Hozzáadás a mapView-hoz (pl. onCreate()-ben):

```
mapView.addView(editText1, screenLP);
```

- Eltűntetés removeView-val

Emlékeztek-e még?

- Mondjatok 3 lehetőséget, hogy a térkép nézeten egy földrajzi pontot megjelöljünk!
  - Overlay rajzolása
  - ItemizedOverlay-re OverlayItem készítés
  - View rögzítése adott GeoPoint-ra

Tegnapi bejelentések

- Honeycomb 3.1 update
  - USB host
    - Kameráról fényképek közvetlen másolása
    - Billentyűzet, egér, játék kontroller, stb…
  - Átméretezhető widgetek
  - Google TV-re is ez jön még idén nyáron
- Android 2.4: Ice Cream Sandwich
- Ugyanaz a UI lesz mindenhol, telefonon és tableten is

Tegnapi bejelentések

- Honeycomb 3.1 update
  - USB host
    - Kamerár
    - Billentyű
  - Átméretezl
  - Google TV
- Android 2.4:

- Ugyanaz a UI lesz mindenhol, telefonon és tableten is