OOT összefoglaló

1. Java ismétlés

1.1. osztályok, interfészek

Object ősosztály

- `equals, hashCode, getClass, toString, clone` (deep vs. shallow)
- szálkezelés (ld ott)

Boxing

Interfész

Belső osztályok

- tagosztály: (akár) elérhető kívülről
- lokális osztály: blokkon belül
- anonim osztály:

```
addWindowListener(new WindowListener() {
        public void windowClosing(WindowEvent e) {
        System.ext(0);}
});
```

1.2. Szálkezelés

Thread (osztály)

- `run()` futtatható kód
- `start`() szál indítása
- `yield` lemond a futásról
- `sleep(long millis)` vár
- `interrupt`
- `join([timeout])` hívó bevárja a megfelelő szálat
- `setDaemon, getID, set/getName, isDaemon, isAlive`
- `static currentThread, static activeCount, static getThreadGroup`

Runnable interfész

- `run()`

Kölcsönös kizárás

- objektumonként monitor + várakozási sor

```
Hashtable<String, Integer> ht = ...;
public void increment(String s) {
        synchronized (ht) {
                int i = ht.get(s);
                i++;
                ht.put(s,i);
}}
```

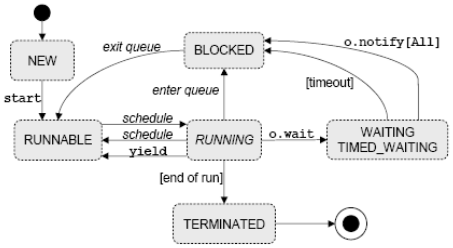
- `synchronized`: blokk vagy metódus előtt

volatile

- sorrend nem optimalizálható
- double, long atomi

szinkronizálás (Object osztály metódusai)

- `wait([timeout])` várakozási sorba kerül
- `notify()` egyet felébreszt
- `notifyAll()` összeset felébreszti



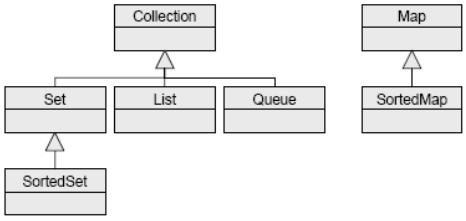
1.3. Genericitás

- <E>
- <?>
- <? extends E>
- <? super E>

1.4. Kollekciók

Collection<E> (általánosan)

- `add(E e), addAll(Collection<? extends E> c)`
- `remove(E e), removeAll(Col<? ext E> c)`
- `contains(E e), containsAll(Col<? ext E> c)`
- `size(), isEmpty()`
- `clear(), retainAll(Col<? ext E> c)`
- `equals(E e)`
- `Object[] toArray(), <T> T[] toArray(T[] ta), Iterator<E> iterator()`



Iterator

`hasNext(), next(), remove()`

Set, SortedSet, List, ListIterator, Queue, Map

for ciklus

```
Collection<Integer> c = ...;
for (Integer i : c) {
	System.out.println(i);
}
```

2. Eclipse

??? fákkjú?

3. Perzisztencia

3.1. Szerializálás

Serializable interfész

- formális
- Tömbök
- nem szérializálható: `transient` és `static`

ObjectInput interfész

- `throws IOException`
- `writeObject(Object obj)`
- `write(int b)`
- `write(byte b[])`
- `write(byte b[], int off, int len)`
- `flush()`
- `close()`

ObjectOutput interfész

- `throws ClassNotFoundException, IOException`
- `Object readObject()`
- …

Externalizable interfész

- kiírás/beolvasás felüldefiniálása

- `writeExternal(ObjectOutput out)`
- `readExternal(ObjectInput in)`

3.2. Hibernate

Alkalmazás átalakítása

- ID attribútumok (jól jöhet)
- konfigurációs fájl (xml)
- HSQL DB indítás

Leképezés

- `<hibernate-mapping>` gyökér
- `<class>` perzisztens osztály → tábla
- `<id>, <generator>` azonosító, generátor algoritmus (pl `native`)
- `<property>` attribútum → oszlop
- `<many-to-one>, <one-to-one>` reláció

```
<hibernate-mapping>
    <class name="auto.Person" table="PERSON">
        <id name="id" column="PERSON_ID">
            <generator class="native"/>
        </id>
        <property name="name"/>
        <set name="cars" inverse="true" cascade="persist">
            <key column="PERSON ID" not-null="true"/>
            <one-to-many class="auto.Car"/>
        </set>
    </class>
    <class name="auto.Car" table="CAR">
        <id name="id" column="CAR ID">
            <generator class="native"/>
        </id>
        <property name="platenr"/>
        <many-to-one   name="owner"   class="auto.Person"   column="PERSON_ID"   not-
null="true" cascade="persist"/>
    </class>
</hibernate-mapping>
```

Kollekciók

`<set>, <list>, <map>, <bag>, <array>, <<p>-array>`

```
<class name="Product">
       <id name="serialNumber" column="SN"/>
       <set name="parts">
              <key column="SN" not-null="true"/>
              <one-to-many class="Part"/>
       </set>
</class>
```

Asszociáció

- referencia más osztályokra
- lásd reláció

Tranzakciók

- csak így érhető el a DB
- Session osztály

Lekérdezések

- `Query q = session.createQuery(…);`
- visszatérés: skalár vagy tömb

```
X x = (X)session.createQuery(..).uniqueResult();
List l = session.createQuery(..).list();
Iterator i = session.createQuery(..).iterate();
```

- paraméterek: név (:xname) vagy sorszám (? ? ?)

```
q.setString(”x”, ”param”);
q.setString(1, ”param1”);
```

HQL

- From
- Join (inner, left outer, right outer, full outer)
- Select
- Aggregáló funkciók (avg(), sum(), min(), max(), count())
- Where
- order by, group by

3.3. PSEPro

Szálak és Sessionök

- egyidőben egy adatbázishoz kapcsolódhat
- tetszőleges read-only tranzakciója lehet
- egyetlen update tranzakciója lehet
- `public static Session create(String host, Properties props)`

Adatbázisok

- nyitás, zárás
- `public static Database create(String name, int fileMod)`

Tranzakciók

- egyszerre egy sessionhöz kapcsolódhat
- `public static Transaction begin(int type)`
- `public void commit(int retain)`
- `public void abort(int retain)`

Objektumok

- root-nak hívják

```
db.createRoot("foo", new Integer(5));
int x = (int) db.getRoot("foo");
```

Query

- paraméteres lekérdezés (`fv` opcionális)

```
FreeVariables fv = new FreeVariables();
fv.put("IS", Integer.TYPE);
Query q = new Query(Person.class,"salary>=IS",fv);
FreeVariableBindings fvb = new FreeVariableBindings();
fvb.put("IS", new Integer(20000));
Collection employees =(Collection)db.getRoot("employees");
Set result = q.select(employees);
```

Példa

```
private String dbName = "cardb.odb";
private Session session;
private Database db;
private Set carOwners;

public void initDB() {
        session = Session.create(null, null);
        session.join();
        try {
                db = Database.open(dbName, ObjectStore.UPDATE);
        }
        catch (DatabaseNotFoundException e) {
                db      =      Database.create(dbName,      ObjectStore.ALL_READ      |
ObjectStore.ALL WRITE);
        }
        Transaction tr = Transaction.begin(ObjectStore.UPDATE);
        try     {
                carOwners = (Set)db.getRoot("OwnersRoot");
        }
        catch (DatabaseRootNotFoundException ex) {
                carOwners = new OSHashSet();
                db.createRoot("OwnersRoot", carOwners);
        }
        tr.commit(ObjectStore.RETAIN_HOLLOW);
```

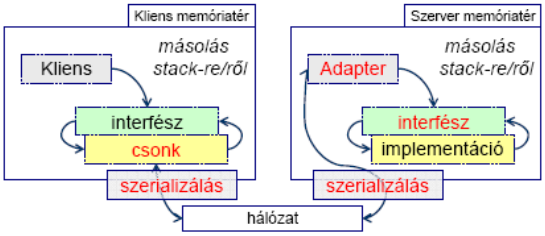
4. Elosztott rendszerek

4.1. Általános

- Socket vs keretrendszer (lábbalhajtós vs kézzeltekerős)
- Szerializálni kell

4.2. Távoli eljáráshívás

Függvényhívás



Csonk

- sorosítva elküldi a paramétereket
- meghívja a távoli függvényt
- visszaveszi a visszatérési értéket

Adapter

- beolvassa a paramétereket
- meghívja az implementációt
- visszaküldi a visszatérési értéket

Memóriakezelés

- primitív típusok
- deep copy

Hálózati hibák

- idempotensnél nincs gond
- at_most_once, at_least_once

Szerver megtalálása

- bedrótozva
- NameService, TradeService

4.3. Remote Method Invocation

Szerializálás

4.3.1. Biztonság

SecurityManager osztály

- `SecurityManager System.getSecurityManager()`
- `System.setSecurityManager(SecurityManager sm)`
- SecurityException
- `void checkAccess(Thread t)`
- `stop, suspend, resume, setPriority, setName, setDaemon`
- `void checkRead(String file)`
- `void checkWrite(String file)`
- `void checkDelete(String file)`
- `void checkConnect(String host, int port)`

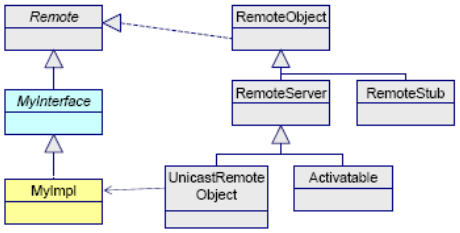
- `void checkAccept(String host, int port)`
- `void checkListen(int port)`
- `void checkExit(int status)`
- `void checkExec(String cmd)`
- `void checkPermission(java.security.Permission)`

Policy fájl

Távoli interfész

- java.rmi.Remote leszármazott
- java.rmi.RemoteException dob
- paraméter: Serializable interfészt megvalósító objektum

4.3.2. Szerver oldal



RemoteObject

- ősosztály
- `boolean equals(Object o)`
- `RemoteRef getRef()`
- `int hashCode()`
- `String toString()`
- `static Remote toStub(Remote obj)`

RemoteServer

- távoli objektum exportálása
- `static String getClientHost()`
- `static void setLog(OutputStream os)`
- `static PrintStream getLog()`

UnicastRemoteObject

- távoli objektum exportálása, stub elérése
- `Object clone()`
- `static RemoteStub exportObject(Remote r)`
- `static Remote exportObject(Remote r, int port [,RMIClientSocketFactory csf, RMIServerSocketFactory ssf])`
- `static boolean unexportObject(Remote r, boolean force)`
- `protected UnicastRemoteObject(int port, RMIClientSocketFactory csf, RMIServerSocketFactory ssf)`

Remote objektum megvalósítás

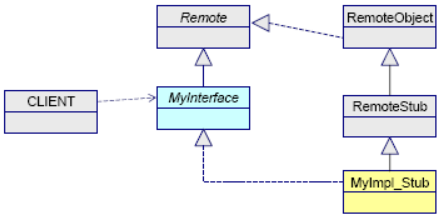
- MyImp implements MyInterface

```
try {
        MyImp imp = new MyImp();
        // exportálás, stub létrehozása explicit!
        MyInterface stub =
                (Hello)UnicastRemoteObject.exportObject(imp,0);
        Registry registry = LocateRegistry.getRegistry();
        registry.rebind("MyInterface", stub);
} catch (Exception e) {e.printStackTrace();}
```

- MyImp extends UnicastRemoteObject implements MyInterface

```
try {
        MyImpl imp = new MyImp();
        Registry registry = LocateRegistry.getRegistry();
        // paraméterátadáskor létrejön a stub
        // és csak ő adódik át
        registry.rebind("MyInterface", imp);
} catch (Exception e) {e.printStackTrace();}
```

4.3.3. Kliens oldal




```
try {
        Registry registry = LocateRegistry.getRegistry();
        MyInterface intf = (MyInterface)registry.lookup("MyInterface");
        intf.myMethod("Arg1");
} catch (Exception e) {e.printStackTrace();}
```

RMIRegistry

- `void bind(String name, Remote obj)`
- `String[] list()`
- `Remote lookup(String name)`
- `void rebind(String name, Remote obj)`
- `void unbind(String name)`

LocateRegistry

- `static Registry createRegistry(int port, RMIClientSocketFactory csf, RMIServerSocketFactory ssf)`
- `static Registry getRegistry(String host, int port, RMIClientSocketFactory csf)`

4.4. Tervezési minták

4.4.1. Callback

- kliens-szerver szerepek keverednek

4.4.2. Factory

- kliens hívásonként új szervantot készít a szerver

4.4.3. Mobil ügynök

- adat és kód együtt utazik

kódmigráció

- codebase=<URL>
- osztálybetöltődés

ügynök

- autonóm
- kapcsolatképes
- közlekedhet
- `Itinerary` minta szerint mozog (`getNextAgency`())

problémák

- osztálybetöltés
- biztonság
- hálózati hibák

tipikus használat

- erőforráskezelés
- információgyűjtés

- hálózati felügyelet

4.5. CORBA (OMG szabvány)

Interface Description Language

- *primitív típusok*: char, octet, boolean, short, long, double
- *összetett típusok*: enum, string, struct, union, sequence, array, valuetype, exception, …
- *paraméterátadás*: in, out, inout
- *interfészek IDL-ben*, implementáció natív nyelven

IDL → Java

- *xxxHelper*: marshallinghoz
- *xxxHolder*: inout-hoz
- *xxxPOA* (Portable Object Adapter): Servant-ból származtatva
- *kliens*: natív típusokkal dolgozhatunk

Szabványos szolgáltatások

- szabványos megoldások gyakori problémákra
- naming/tradingService

5. Ablakkezelés

5.1. AWT

Container

- tartalmazás felelőssége
- komponensek megtalálása: pozíció, sorszám, összes
- fókusz továbbadás
- elhelyezés: `LayoutManager` (lásd ott)

5.2. Eseménykezelés

- kezdeti nehézségek
- Observer minta

xxxEventListener interfész

- `addXxxEventListener(XEventListener el)`

xxxEventAdapter osztály

`xxxEventListener megvalósítása üres metódusokkal`

5.3. Fókusz-kezelés

KeyEventDispatcher interfész

- más is hozzáférhet a leütött billentyűhöz, nem csak a *focus owner*…
- `boolean dispatchKeyEvent(KeyEvent e) //továbbadódjon-e`
- `KeyboardFocusManager`

KeyEventPostProcessor interfész

- feldolgozás után más is hozzáfér
- `boolean postProcessKeyEvent(KeyEvent e)`
- `KeyboardFocusManager`

WindowEvent

- `WINDOW_ACTIVATED / WINDOW_DEACTIVATED`
- `WINDOW_GAINED_FOCUS / WINDOW_LOST_FOCUS`

FocusEvent

- `FOCUS_GAINED / FOCUS_LOST`
- `setFocusTraversalKeys(int id, Set<? extends AWTKeyStroke> keystrokes)`
- `KeyboardFocusManager.XXX_TRAVERSAL_KEYS:`
- `FORWARD, BACKWARD, UP_CYCLE`

Fókusz továbbadás

- FocusTraversalPolicy
- ContainerOrderFocusTraversalPolicy
- DefaultFocusTraversalPolicy
- SortingFocusTraversalPolicy
- LayoutFocusTraversalPolicy

KeyboardFocusManager

- `focusNextComponent(Component)`
- `focusPreviousComponent(Component)`
- `upFocusCycle(Component)`
- `downFocusCycle(Container)`

Component

- `transferFocus()`
- `transferFocusBackward()`
- `transferFocusUpCycle()`
- `transferFocusDownCycle()`
- `void requestFocus()`
- `boolean requestFocusInWindow()`

5.4. Layout managerek

Container

- `void setLayout(LayoutManager mgr)`
- `LayoutManager getLayout()`
- `void validate()`
- `Component add(Component comp [,int index])`
- `void add(Component c, Object constraint, int index)`

LayoutManager

- `void addLayoutComponent(String name, Component`
- `comp)`
- `void removeLayoutComponent(Component comp)`
- `void layoutContainer(Container parent)`
- `Dimension minimumLayoutSize(Container parent)`
- `Dimension preferredLayoutSize(Container parent)`
- `void addLayoutComponent(Component comp, Object constraints)`
- `float getLayoutAlignmentX(Container target)`
- `float getLayoutAlignmentY(Container target)`
- `void invalidateLayout(Container target)`
- `Dimension maximumLayoutSize(Container target)`

BorderLayout

- öt mező

FlowLayout

- sorban egymás mellé
- új sor, ha megtelt

CardLayout

- mindig a legfelső látszik

GridLayout

- táblázatos

GridBagLayout

- táblázat összevont cellákkal
- `GridBagConstraint` segít az elrendezésben (`gridx/y, gridwidth/height, weightx/y, ipadx/y, insets, fill, anchor`)

(swing) BoxLayout, SpringLayout, GroupLayout

5.5. Swing

- Java-ban megírt
- MVC minta
- testreszabható
- egyszerű komponensek: +J betű

JList

- nem méretezhető: `JScrollPane`-be kell tenni (Decorator minta)
- modell: `ListModel`

ListModel interfész

- `Object getElementAt(int index)`
- `int getSize()`
- `void removeListDataListener(ListDataListener l)`
- `void addListDataListener(ListDataListener l)`

ListDataListener

- `void intervalAdded(ListDataEvent e)`
- `void intervalRemoved(ListDataEvent e)`
- `void contentsChanged(ListDataEvent e)`

DefaultListModel

- ListModel megvalósítás
- `void add(int index, Object o)`
- `int size()`
- `Object get(int index)`
- `Object remove(int index)`

JTable

- modell: TableModel

TableModel

- esemény: TableModelListener
- `boolean isCellEditable(int r, int c)`
- `String getColumnName(int col)`
- `void setValueAt(Object aValue, int rowIndex, int columnIndex)`
- `Object getValueAt(int r, int c)`
- `int getColumnCount()`
- `int getRowCount()`

JTree

- BinTreeModel

5.6. Drag and Drop

- `JComponent.setDragEnabled(boolean b)`
- `setDropMode(DropMode dm)`
- `INSERT, INSERT_COLS / INSERT_ROWS, ON, USE_SELECTION, ON_OR_INSERT / ON_OR_INSERT_COLS / ON_OR_INSERT_ROWS`

TransferHandler

- segítsével testre szabhatjuk a komponens viselkedését
- `JComponent.setTransferHandler(TransferHandler th)`
- `int getSourceActions(JComponent): COPY, MOVE, LINK`
- `Transferable createTransferable(JComponent)`
- `void exportDone(JComponent c, Transferable t, int action)`
- `boolean canImport(TransferHandler.TransferSupport ts)`
- `boolean importData(TransferHandler.TransferSupport ts)`

TransferSupport

- segít Drag and Drop lebonyolításában
- `Component getComponent()`
- `int getDropAction()`
- `int getSourceDropActions()`
- `DataFlavor[] getDataFlavors()`
- `boolean isDataFlavorSupported(DataFlavor)`
- `Transferable getTransferable()`
- `DropLocation getDropLocation()`

Transferable

- interfész adatok fogadására
- `Object getTransferData(DataFlavor flavor)`
- `DataFlavor[] getTransferDataFlavors()`
- `boolean isDataFlavorSupported(DataFlavor flavor)`

DataFlavor

- adatformátumokat tárol, ami megjelenhet a drag and drop-ban (és a vágólap és fájrendszerben)
- `DataFlavor(Class<?> representationClass, String humanPresentableName)`

5.7. Szálkezelés

SwingWorker<T,V>

- szálként viselkedik
- visszatérési értéke van
- képes kódot futtatni az eseménykezelő szálban
- `protected abstract T doInBackground()`
- `void execute()`
- `protected void done()`
- `boolean isDone()`
- `T get(long timeout, TimeUnit unit)`
- `void setProgress(int i)`
- `int getProgress()`
- `void cancel(boolean mayInterruptIfRunning)`
- `boolean isCancelled()`
- `protected final void publish(V... chunks)`
- `protected void process(List<V> chunks)`
- `public final SwingWorker.StateValue getState(): PENDING, STARTED, DONE`

5.8. Beágyazott

- kevés erőforrás
- folyamatos VM futás
- listener eseménykezelő nem nyerő, Whiteboard kell

Whiteboard minta

- közös registry
- ide regisztrálnak az érdeklődök
- forrás ezt értesíti

6. OO metrikák

6.1. Tervezési elvek

Csatolás

- függőségek minimalizálása
- package, osztály, objektum
- *D(ARP)C* = Direct (Attribute, Reference, Parameter) Based Coupling (azon különbözı osztályok száma, amelyeknek (attribútumát, metódusát, paraméterként) érjük el)
- *DCC* = Direct Class Coupling
- *CBO* = Coupling Between Objects
- **Instability (RMI) = kimenő_csatolás / (bejövő_csatolás + kimenő_csatolás)**

Osztály csatolás

- *NUCD* = Number of used classes by dependency relation
- *TNUCD* = Total number of evidences for 'Used classes by dependency relation'
- *RNUCD* = NUCD / TNUCD
- öröklés: mélysége, gyerekek száma
- asszociáció: *NAC* = Number of associated classes with a class
- TACU = Total associated class Usages

Kohézió

- egységen (osztályon) belül
- **LCOM: |P| - |Q|**
- |P| = függvények száma, nincs közös attribútum, |Q| = függvények száma, közös attribútummal

Egyéb

- *CC* = Cyclomatic Complexity: metódus bonyolultsága
- *WMC* = Weighted methods per class = ∑ CC (minden metódusra)
- *RFC* = Response for a class: metódusok száma
- *RMA* = Abstractness: absztrakt / nem absztrakt (package)

6.2. Cocomo

Effort

- Effort = 2.94 * EAF * $(KSLOC)^E$
- Effort (PersonMonth, PM)
- EAF = Effort Adjustment Factor
- KSLOC = kilo source LOC
- E = exponens

Duration

- Duration = 3,67 * $(Effort)^{SE}$
- Duration – (PM-ben)
- Effort (PersonMonth, PM)
- SE = schedule equation exponent

6.3. CDP

6.3.1. Locking minták

Lock

- probléma: biztonságos lock felszabadítás

- megoldás: saját guard osztály

Double check
- ágymásba ágyazot dupla ellenőrzése

Thread-safe lock
- mo: két interfész: külső ellenőriz, belső (privát) elfogad

Bedrótozott lock
- mo: run-time szinkronizáció

6.3.2. Konkurencia minták

Monitor objektum
- több kliens esetén
- mo: minden objektumnak van monitora + várakozási sora (lásd Objektum): synchronized blokk

Aktív objektum
- metódushívás nem blokkolhat
- mo: hívás és végrehajtás szétcsatolása, végrehajtás saját szálon (kell valami üzenetsor)

Reactor
- 1 szerver, több kliens
- mo: szinkron események demultiplexelése

Vezető-követő
- események konkurrensen
- mo: szálcsokor, szinkron események demultiplexelése

6.3.3. Esemény minták

Proactor
- mint a Reactor, csak itt esemény
- mo: mint a Reactor (szinkron események demultiplexelése)

Csatlakozó (acceptor-connector)
- összefonódik a kapcsolódás és kommunikáció
- mo: külön szedni (szinkron vagy aszinkron kapcsolódás)

6.4. Visitor Combinator

Fákkjú

7. XML

7.1. Általános

jól formált
- opcionális fejléc, gyökérelem, nyitó-záró tag
- valid: megfelel a sémának

7.2. XSD

gyökérelem
- `<xs:schema>…</xs:schema>`

- `xmlns:xs="http://www.w3.org/2001/XMLSchema"` adattípusok definíciójának névtere
- `targetNamespace="http://www.w3schools.com"` elemek névtere
- `xmlns="http://www.w3schools.com"` alapértelmezett névtér
- `elementFormDefault="qualified"` kötelező névteret tartalmaznia

egyszerű elem
- `<xs:element name="xxx" type="string"/>`

beépített típusok
- xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time

kezdő érték
- `<xs:element name="xxx" type="yyy" default="zzz"/>`

fix érték
- `<xs:element name="xxx" type="yyy" fix="qqq"/>`

attribútum
- `<xs:attribute name="xxx" type="yyy" [use="required"]/>`

típusdefiníció
```
<xs:simpleType>
      <xs:restriction base="xs:string">…</xs:restriction>
</xs:simpleType>
```

megkötések
- totalDigits, fractionDigits, enumeration, length, minLength, maxLength…

összetett elemek
- `<xs:complexType name="personinfo">…</xs:complexType>`
- öröklés: `<xs:extension base="personinfo">…</xs:extension>`

Indikátorok
- *Sorrend*: `all, choice, sequence`
- *Gyakoriság*: `maxOccurs, minOccurs`
- *Csoportosítás*: `group <név>, attributeGroup`

helyettesítés
- `<xs:element name="name" type="xs:string"/>`
- `<xs:element name="név" substitutionGroup="name"/>`

hivatkozás XML-ből
```
<note xmlns=http://www.w3schools.com
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://www.w3schools.com note.xsd">
```

7.3. XPath

csomópontnév
- adott nevű csomópont gyerekei

/
- gyökérelem

//
- az aktuális elemtől választ, mindegy, hogy milyen mélyen

.
- aktuális elem

..
- az aktuális elem szülője

@<név>
- attribútum

[feltétel]
- megfelelő elem után
- index, vagy feltétel elemre, attribútumra, vagy függvény

*
- bármely elem

ágak (hosszú név)
- `self`
- `parent`
- `ancestor / ancestor-or-self`
- `child`
- `descendant / descendant-or-self`
- `following`
- `following-sibling`
- `preceding`
- `preceding-sibling`
- `namespace`
- `attribute`

rövidítés (kettő megegyezik)
- `child::A/descendant-orself::node()/child::B[position()=1]`
- `A//B/*[1]`

operátorok
- aritmetikai: + - / * div mod
- logikai: and or
- relációs: = != < <= > >=
- únió: |

függvények (csak pl)
- `fn:round(num)`
- `fn:substring(string,start,len)`
- `fn:current-time()`

7.4. XSLT

sablon definíció
- `<xsl:template match="/">`

felsorolás
- `<xsl:for-each select="neptun/student">`
- feltétellel: `<xsl:for-each select="neptun/student[average&gt;'3.0']">`

elem érték
- `<xsl:value-of select="name"/>`

rendezés
- `<xsl:sort select="name"/>`

hozzárendelés az XML-ben
- `<?xml-stylesheet type="text/xsl" href="test1.xsl"?>`

feltétel

- `<xsl:if test="average &lt;2.0">` csak egy feltétel (nincs else ág)!

többszörös elágazás

```
<xsl:choose>
	<xsl:when test="…">…</xsl:when>
	<xsl:when test="…">…</xsl:when>
	<xsl:otherwise>…</xsl:otherwise>
</xsl:choose>
```

sablon alkalmazása

- `<xsl:apply-templates select="id"/>`
- korábban definiálni kell: `<xsl:template match="id">…</xsl:template>`

7.5. Simple API for XML

- soros
- állapotfüggetlen
- parser

org.xml.sax.ContentHandler interfész

- throws SAXException
- `void startDocument() – void endDocument()`
- `void startElement(String namespaceURI, String sName, String qName, Attributes attrs) – void endElement()`
- `void startPrefixMapping(…) – void endPrefixMapping(…)`
- `void characters(char[] ch, int start, int length)`
- `void ignorableWhiteSpace(…)`
- `void processingInstruction(…)`
- `void skippedEntity(…)`
- `void setDocumentLocator(…)`
- üres megvalósítás: org.xml.sax.helpers.DefaultHandler

hibakezelés

- fatal error: syntax error
- error: nem érvényes (nem illeszkedik a sémára)
- warning: egyéb gányság (kétszeres típus)

7.6. Document Object Model

7.6.1. DOM általánosan

Document

- dokumentumot reprezentálja
- lekérdezhető adatok: doctype, verzió, kódolás, stb
- elemeket tud gyártani
- maga is egy `Node`

Node

- egy elemet reprezentál
- lekérdezhető, módosítható adatok: érték, attribútum, típus
- navigálás: lefelé (gyerekek), felfelé (szülők), oldalra (testvérek)
- gyerekeket módosíthatjuk

NodeList

- amikor lekérjük egy `Node` gyerekeit: `NodeList Node.getChildNodes()`
- `int getLength()`
- `Node item(int index)`

NamedNodeMap

- Node elérésében segít
- `NamedNodeMap Node.getAttributes()`
- `int getLength()`
- `Node item(int index)`
- `Node getNamedItem(String name)`
- `Node getNamedItemNS(String namespaceURI, String localName)`
- `Node removeNamedItem(String name)`
- `Node removeNamedItemNS(String namespaceURI, String localName)`
- `Node setNamedItem(Node arg)`
- `Node setNamedItemNS(Node arg)`

7.6.2. JDOM

- Java-s szemlélet
- `List`-et használ `NamedNodeMap`, `NodeList` helyett
- szűrők: `org.jdom.filter.Filter`
- XSL: `org.jdom.transform.XSLTranformer`
- XPath: `org.jdom.xpath.XPath`
- beolvasás külső segítséggel: `DOMBuilder` vagy `SAXBuilder`