

# Szoftvertchnológia és -technikák

## 4. Előadás

### *Aktivitásdiagram, Állapotgép*



# Osztálydiagram és szekvenciadiagram

# Osztályok és interfészek

- Osztályok és interfészek

- > Fejléc

- > Tagváltozók

- [Láthatóság] [Név]: [Típus][Multiplicitás] [= kezdőérték]
    - Statikus: aláhúzás

- > Metódusok/Műveletek

- [Láthatóság] [Név]([paraméterek])([: visszatérési érték])

- > Láthatóság

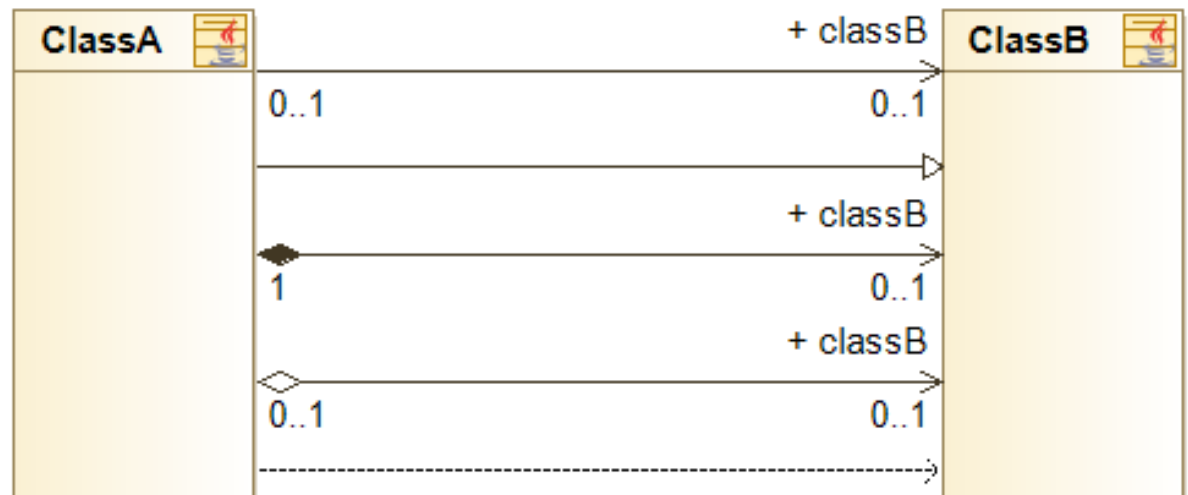
- Public: +
    - Private: -
    - Protected: #
    - Package: ~



# Kapcsolatok

- Kapcsolat típusok

- > Asszociáció
- > Általánosítás/  
Öröklés
- > Kompozíció
- > Aggregáció
- > Függőség
- > Interfész  
megvalósítás



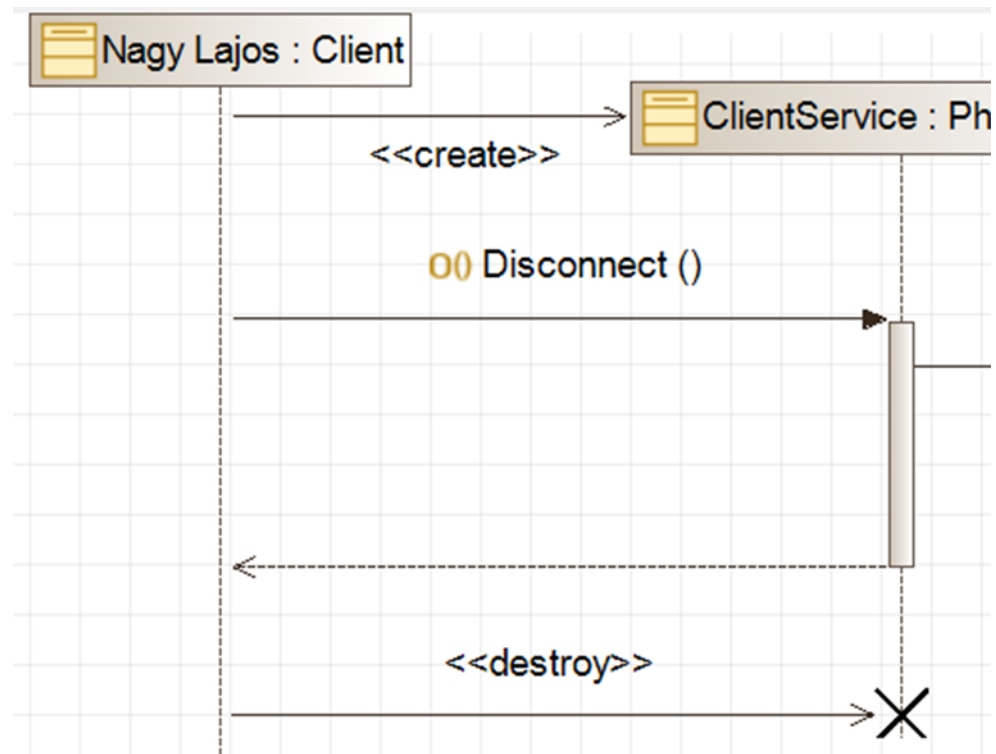
- Multiplicitás

- Navigálás

- > Szerepnév
- > Navigálhatóság

# Életvonalak és üzenetek

- Életvonal
  - > [Név]: [típus]
- Üzenet
  - > Név([params])
  - > Szinkron/aszinkron
  - > Létrehozás/  
megszűntetés
- Futási/végrehajtási  
specifikáció (execution specification)



# Blokkok

- Kombinált részletek (Combined fragments)
  - > Alt: Feltételes elágazás (if – else if – else szerkezet)
  - > Loop: Feltétel szerinti előltesztelő ciklus
  - > Opt: Opcionális utasítások (else ág nélküli if)
  - > Par: Párhuzamosan végrehajtandó utasítások
  - > Seq, Critical, Assert, ... (nem tanuljuk)



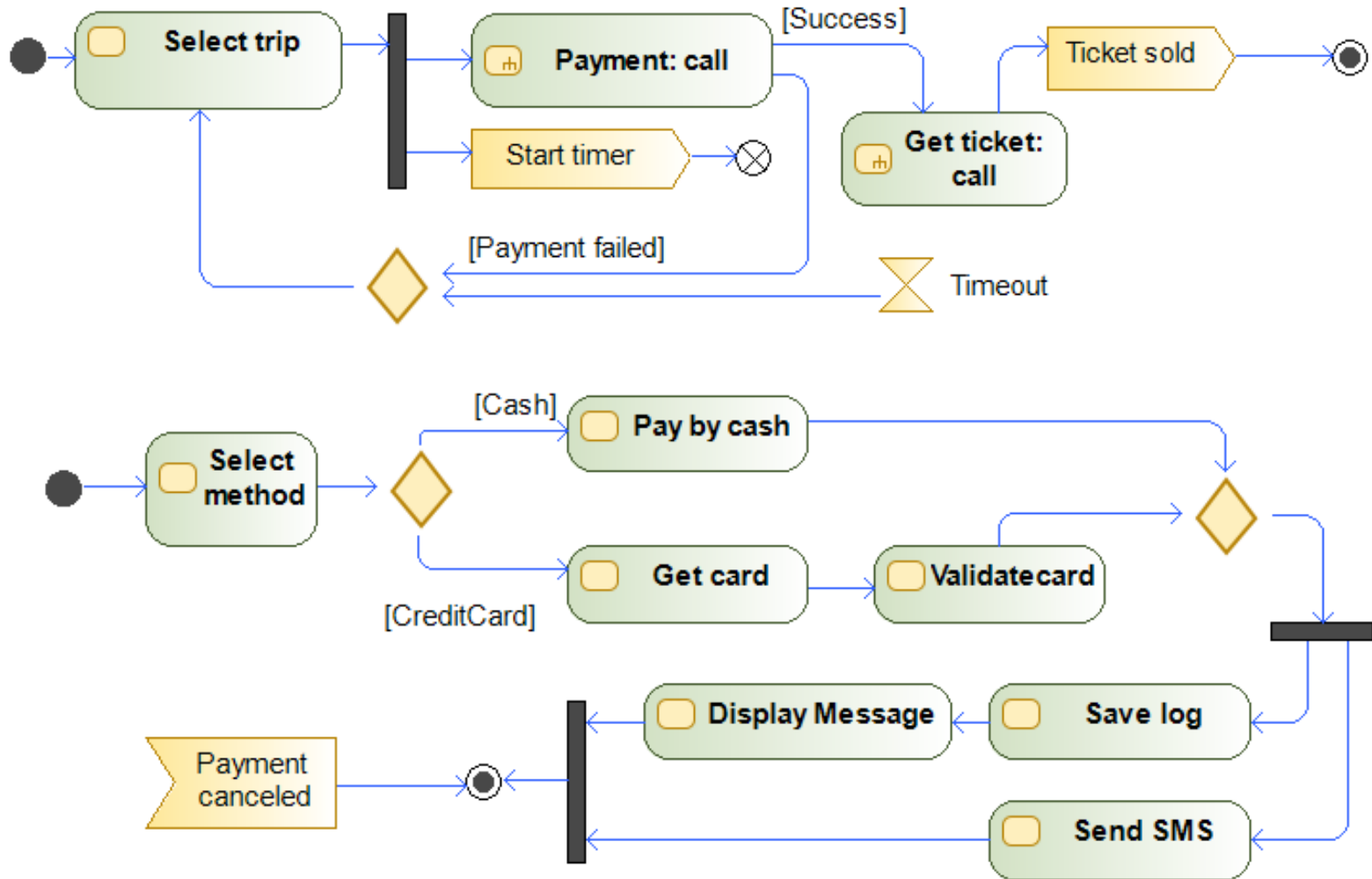
# Aktivitás diagram

# Aktivitásdiagram

- Struktúra – osztálydiagram
- Műveletek sorrendje – szekvenciadiagram
  - > Szekvenciális végrehajtás
  - > Jellemzően függvényhívások
- Munkafolyamatok leírása – Aktivitás diagram
  - > Workflow, control flow
  - > Elágazások, párhuzamos ágak, feltételek
  - > Fókusz a funkciókon, a folyamatokon
  - > Magasabb szintű, több használati esetet is összefoghat
  - > <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

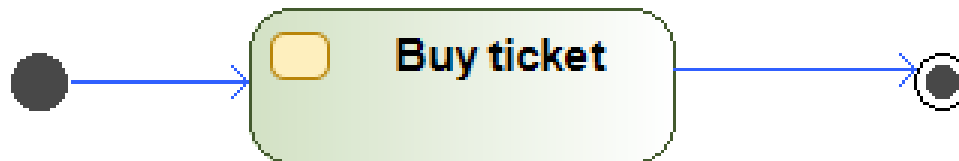


# Aktivitásdiagram: BKV jegyvásárlás



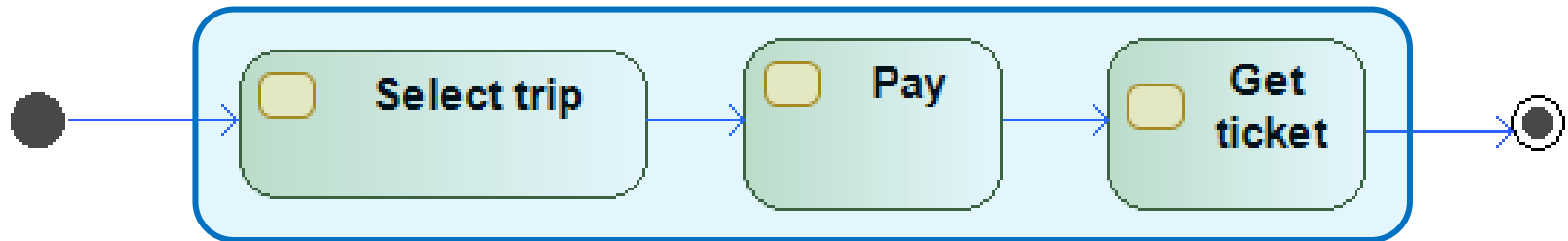
# Jegyvásárlás

- Modellezzük a jegyvásárlást!
  - > Kezdő csomópont (initial node)
    - Folyamat kezdése
    - Nincs bemenő él
    - Több kezdőpont: párhuzamos kezdés
  - > Aktivitás (Activity)
    - Folyamat befejezése
    - Nincs kimenő él
  - > Végző csomópont (final node)
    - Folyamat befejezése
    - Nincs kimenő él
  - > Vezérlési él (control flow)
    - Elindítja a cél aktivitás futtatását
    - Forrás : előfeltétel ; él – külső akció ; nyelő : a rendszer lépése



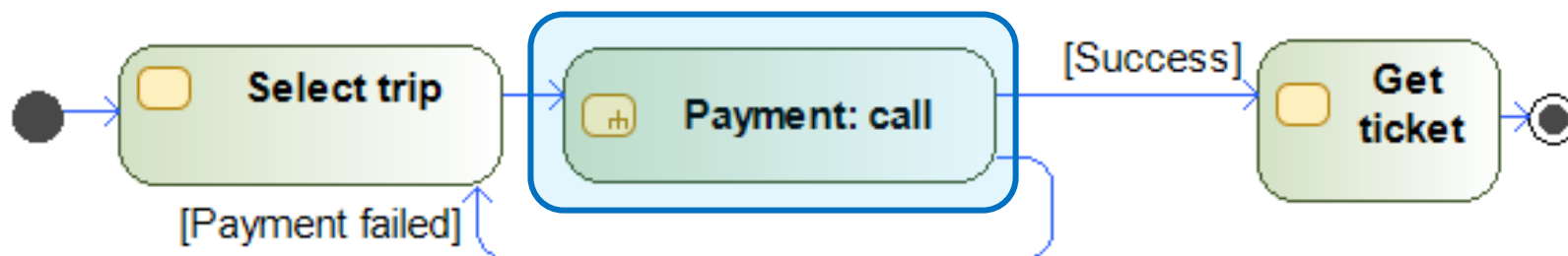
# Több lépésben

- Bontsuk több lépésre a vásárlást!
  - > Lépésről-lépésre finomíthatjuk a folyamat leírását



# Örfeltétel és dekompozíció

- Csak sikeres fizetés esetén adjuk ki a jegyet!
  - > Sikeres – sikertelen fizetés
    - Örfeltétel (guard condition):  
Szabályozza melyik vezérlési él válhat aktívá
  - > A fizetési folyamatot bontsuk lépésekre!
    - Hívás aktivitás (call activity):  
Hierarchikus dekompozíció



# Fizetés

- Lehesse kártyával és készpénzzel fizetni!



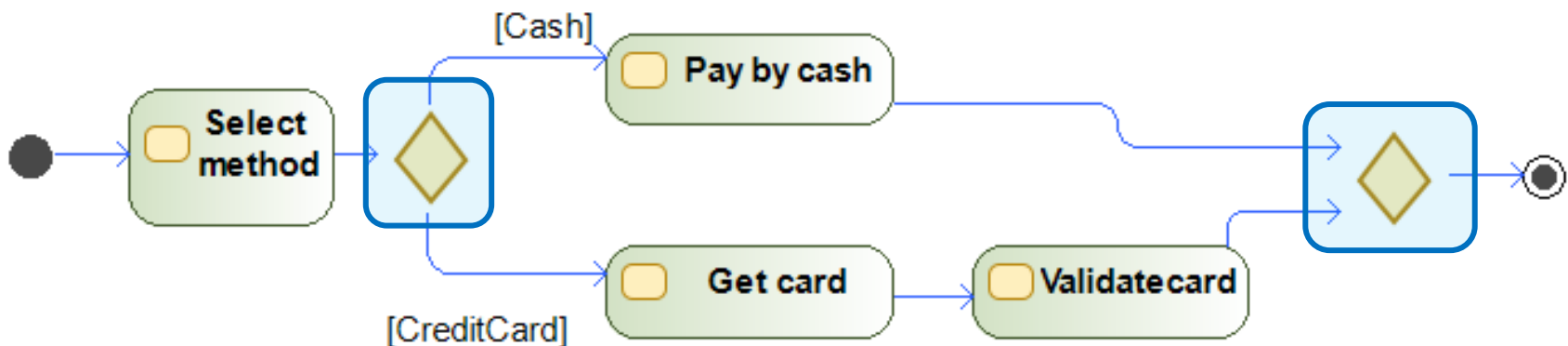
- > Döntés (decision):

Legfeljebb egy kimenő él aktiválható

- > Összevonás (merge):

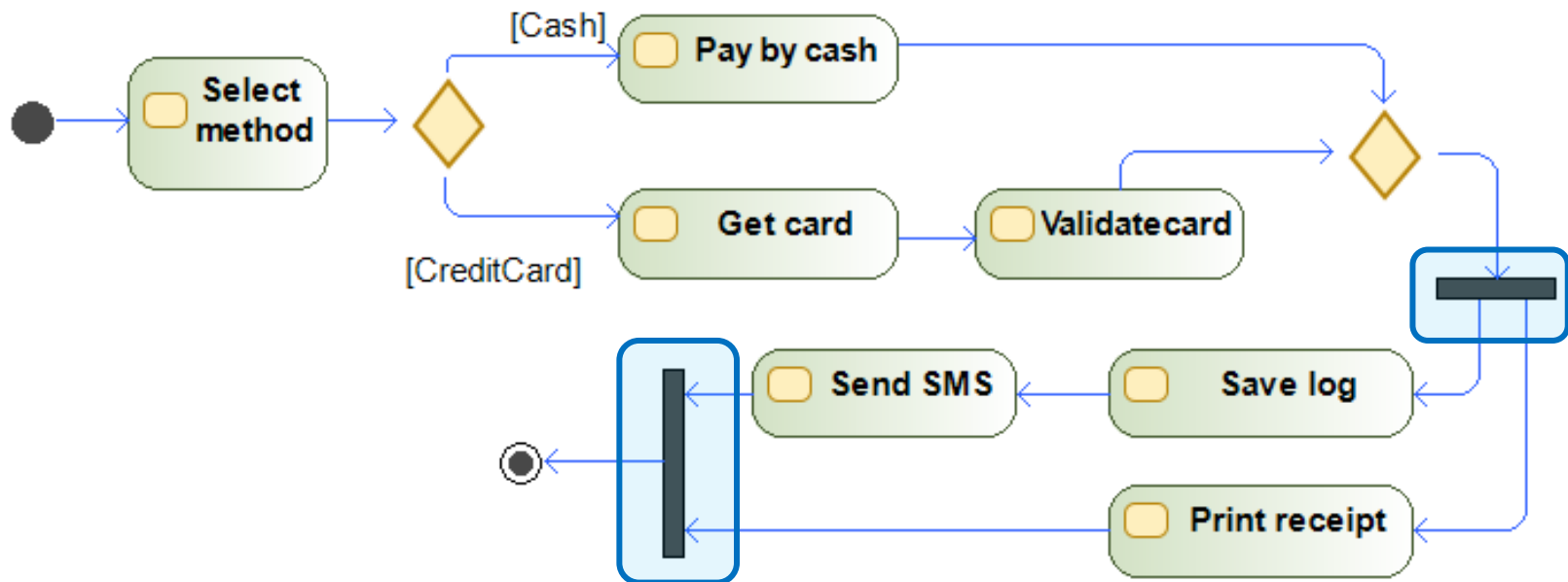
Összefog több ágat, minden beérkező ágra lefut újra

- > Kártyás fizetésnél ellenőrizzük is le a kártyát!



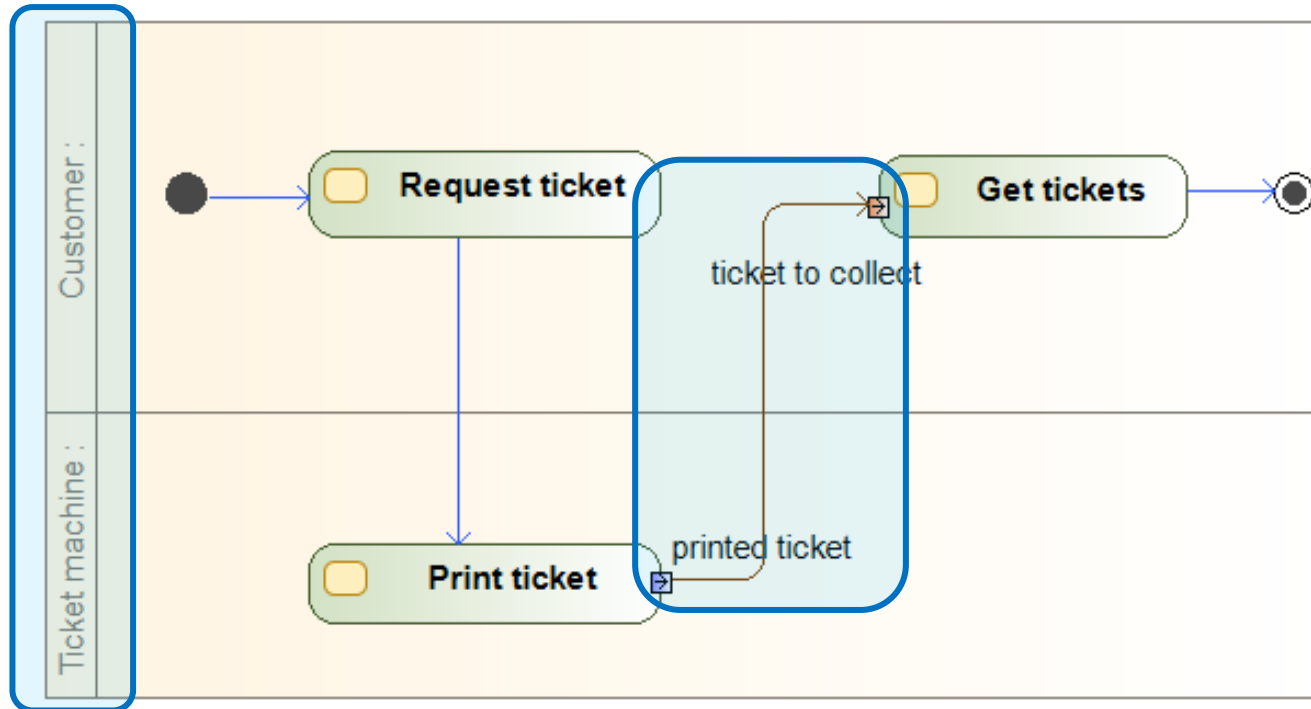
# Fizetés

- A fizetés jóváhagyása után
  - > Párhuzamosan:
    - Jegyezzük be a fizetést, küldjünk sms-t!
    - Nyomtassunk ki bizonylatot!
  - > Elágazás (fork), ill. Egyesítés (join)
    - Párhuzamos futás, szinkronizáció



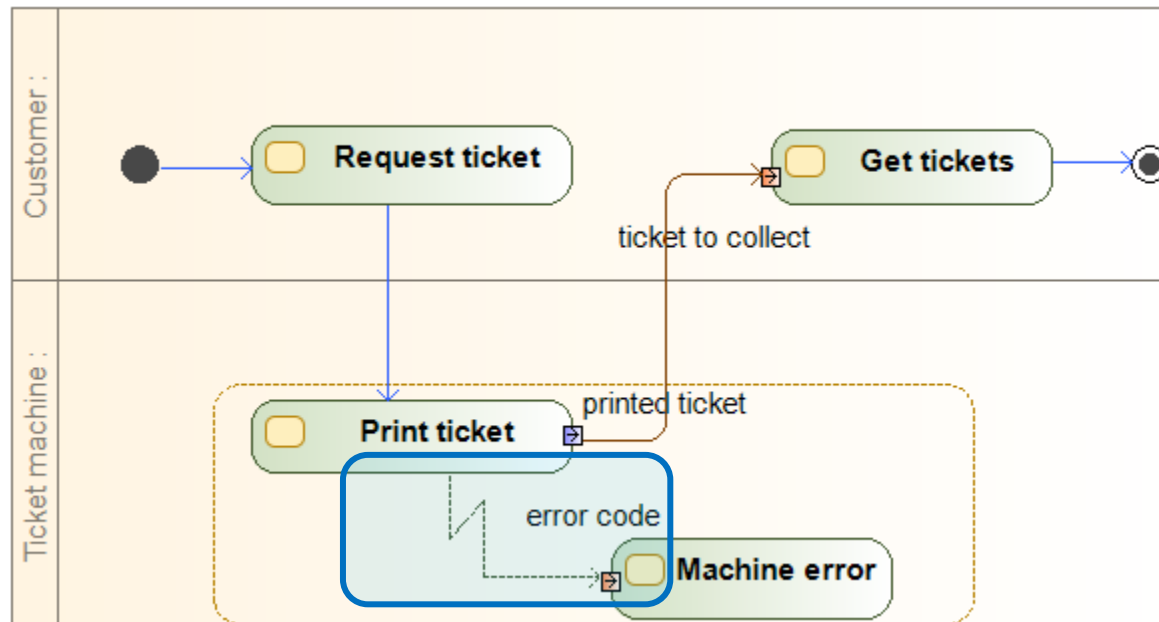
# Jegy átvétele

- A jegyet az automata nyomtatja
  - > Két résztvevő rendszer (szerepkör) van a folyamatban!
    - Partíció (partition)
  - > A jegy átadása adat átadás, nem vezérlés!
    - Objektum él (object flow)
    - Objektum csomópont (object node)
    - Tüske (pin) – az objektum csomópont átadásának jelölésére, egy fajta objektum



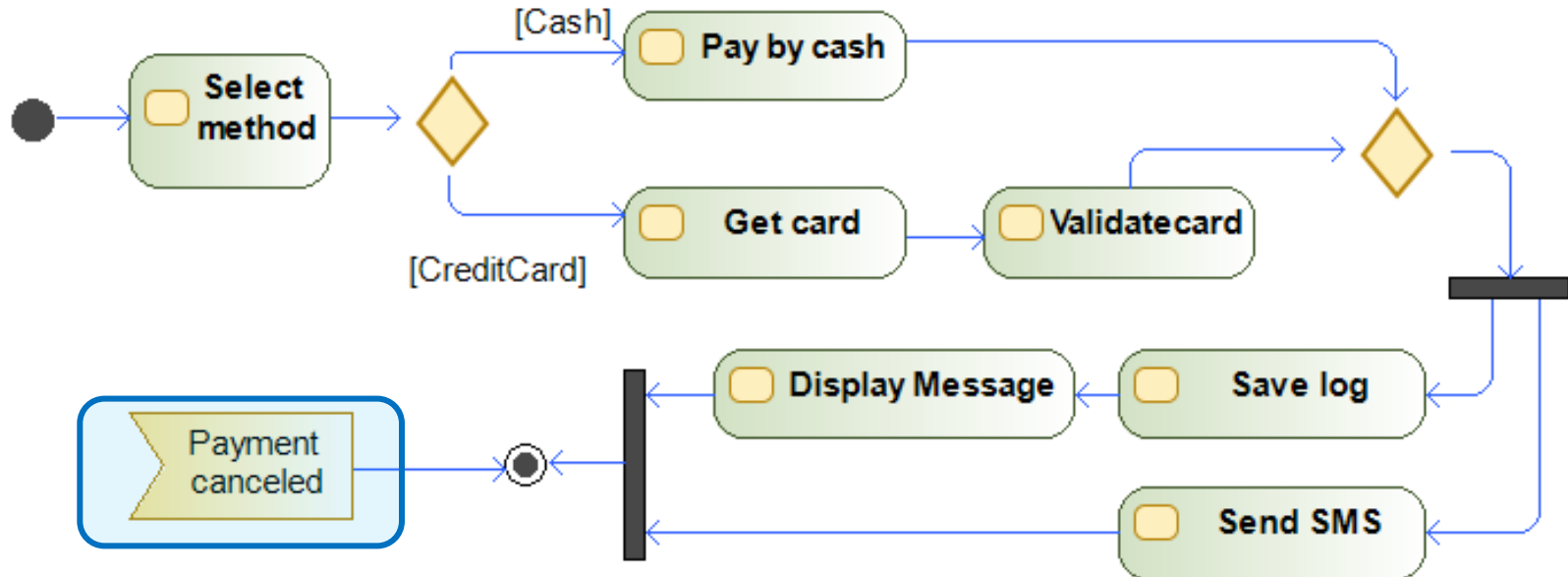
# Hibakezelés

- Ha a nyomtatás során hiba adódik, kezelni kell
  - > Kivételkezelés, adatátadással
  - > Kivétel (exception)
    - Adott régióban történő váratlan megszakításhoz
    - (Modelio nem támogatja a „villám” jelölést)



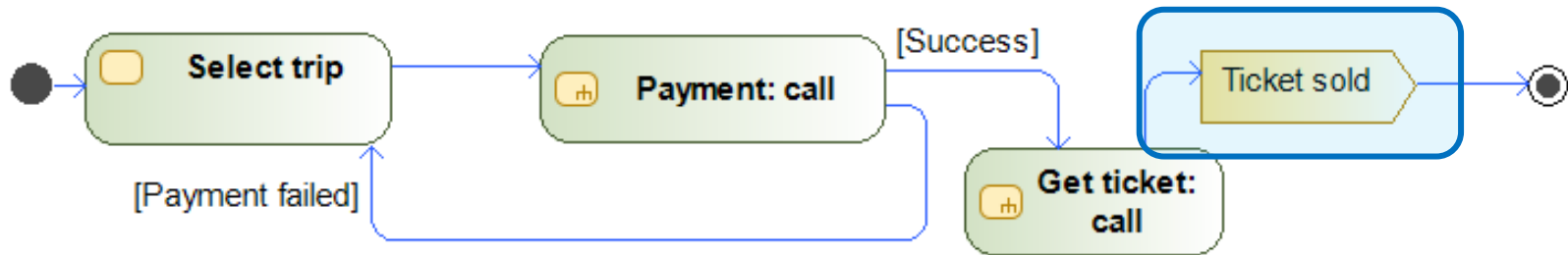
# Események

- A fizetés legyen bármikor megszakítható!
  - > Esemény alapú vezérlés: akkor folytatódik a vezérlés, ha az esemény bekövetkezett
  - > Az esemény bárhol kiváltható (mi most nem modellezzük)
  - > Esemény fogadása akció (accept event action)



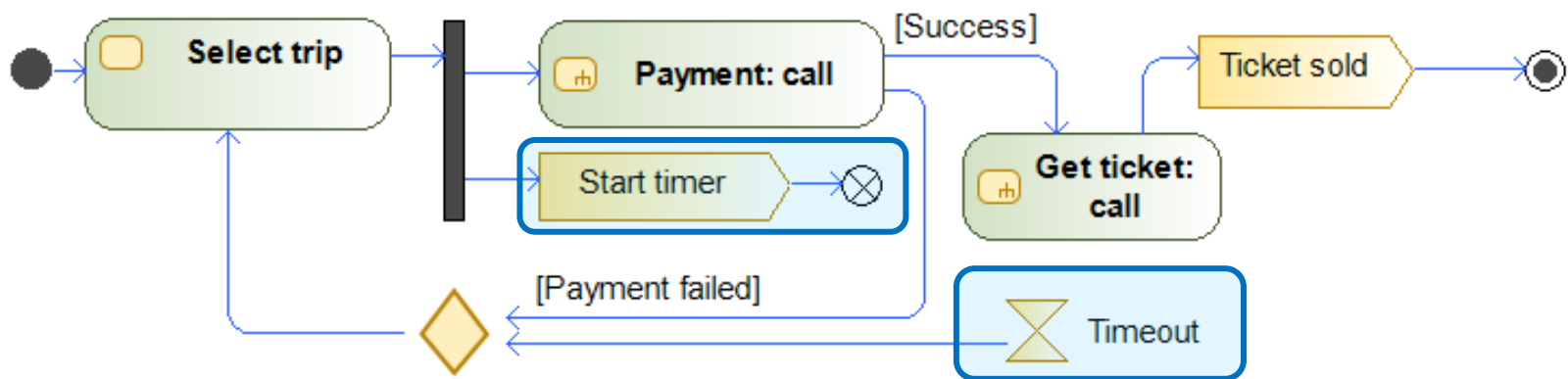
# Események

- A sikeres tranzakciót eseményen keresztül publikáljuk!
  - > Esemény küldése akció (send signal action)



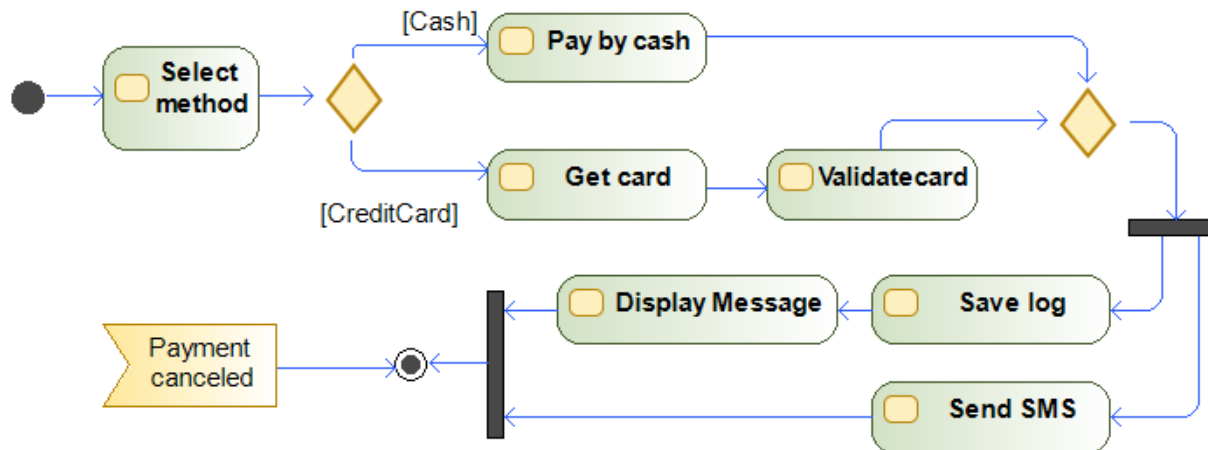
# Események

- A tranzakciónak legyen időkerete! Ha az idő lejár, automatikusan váljon sikertelenné a tranzakció!
  - > Időesemény fogadása akció (accept time event action)
- Indítsuk el az időzítőt az elején!
  - > Az időzítő elindítása után az a folyamat ág véget ér, de a teljes folyamat nem!
  - > Folyam vége csomópont (flow final node)



# Összefoglalás

- Aktivitás
- Kezdő
- Végcsomópont
  - > Folyam vége
  - > Végző csomópont
- Vezérlési (folyam) él
  - > Őrfeltétel
- Hívás aktivitás
- Döntés – összevonás
- Elágazás – egyesítés
- Partíció
- Objektum, objektum él
- Esemény – fogadás és küldés
- Időesemény

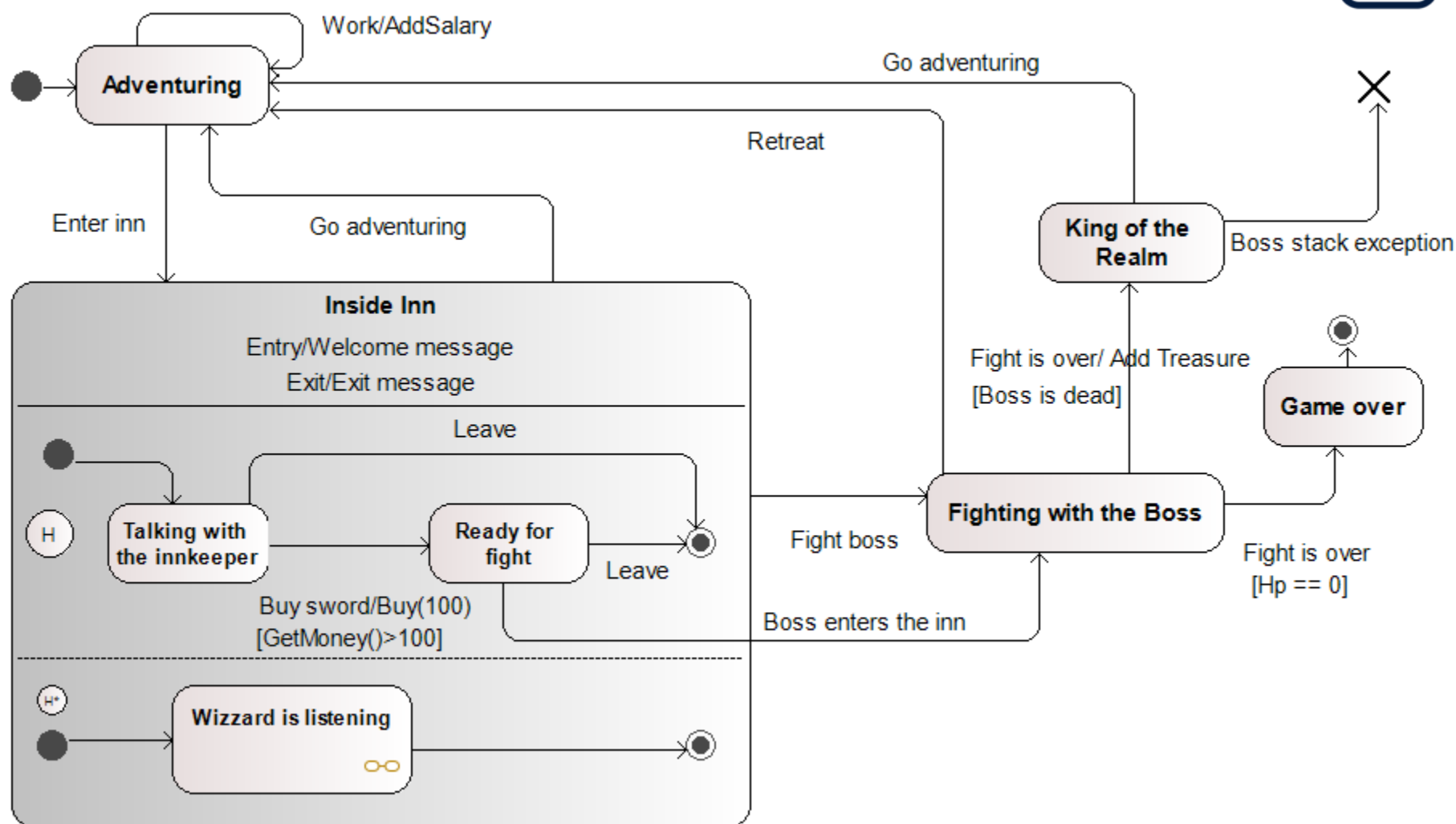


# Állapotgép

# Állapotgépek

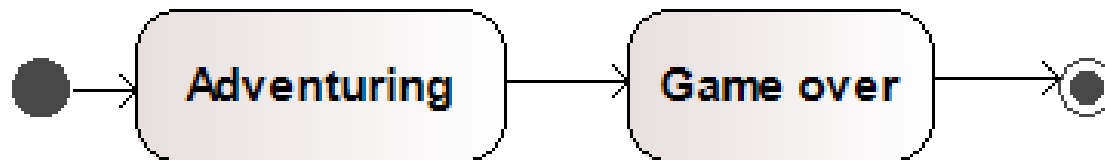
- Állapotgép (Statemachine, statechart diagram)
  - > Objektum-orientált, eseményvezérelt “véges automata”
  - > Fókusz:
    - A rendszer, vagy egy komponens állapotai
    - Állapotátmentek és feltételek
  - > Két fajta
    - Viselkedést leíró (Behavioral state machines)
    - Protokoll leíró (Protocol state machines)

# Állapotgép: Kalandjáték



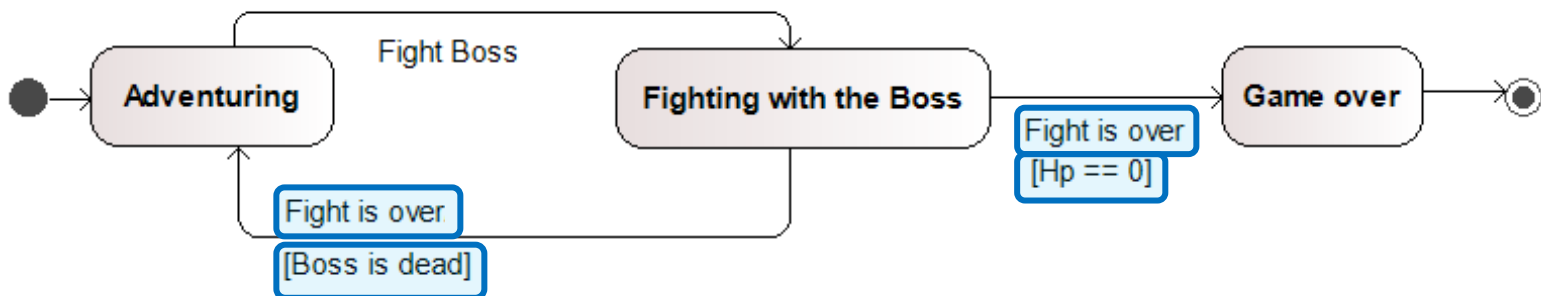
# A kezdet és a vég

- Modellezzük egy kalandjátékot!
  - > A játék elkezdődik, kalandozunk, majd véget ér
  - > Kezdő állapot (initial state)
  - > Állapot (state)
  - > Átmenet (transition)
    - Ha több lehetséges átmenet van, nem-determinisztikusan választunk egyet
  - > Végző állapot (final state)



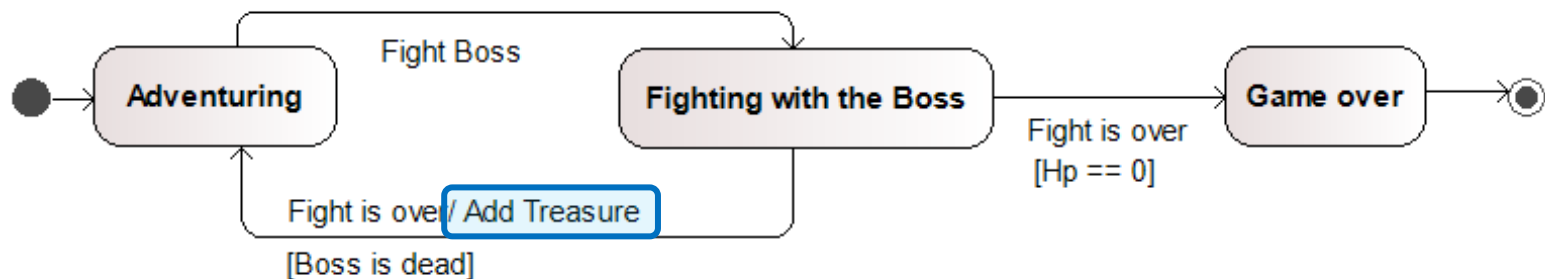
# Főellenség

- Legyen egy főellenség, akit le kell győzni!
  - > Trigger (trigger)
    - Az állapotátmenet oka, leggyakrabban egy esemény
    - Kódban általában az objektumon kívülről hívott metódus
- Ne legyen automatikus a győzelem/vesztés!
  - > Őrfeltétel (guard condition)
    - Lehetséges átmenetnél a feltételnek is teljesülnie kell
    - Triggerek finomhangolása



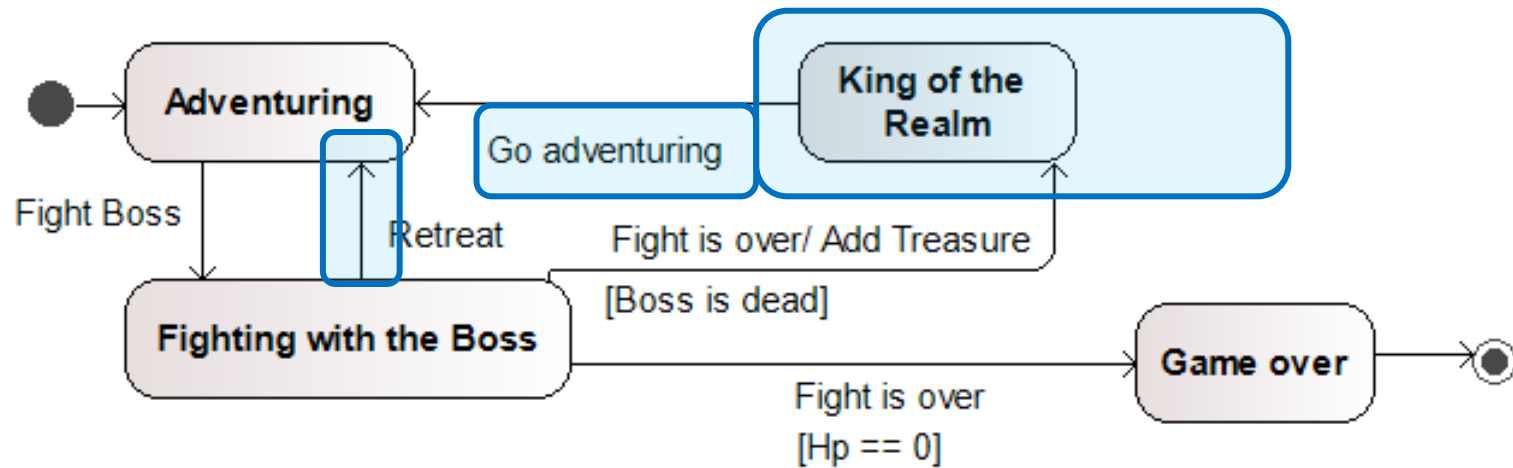
# Jutalom

- Ha győztünk, kapjunk jutalmat!
  - > Akciók (action, hivatalosan: behavior expression)
    - Az átmenet során végrehajtandó akció
    - Kódban általában belső függvényhívás



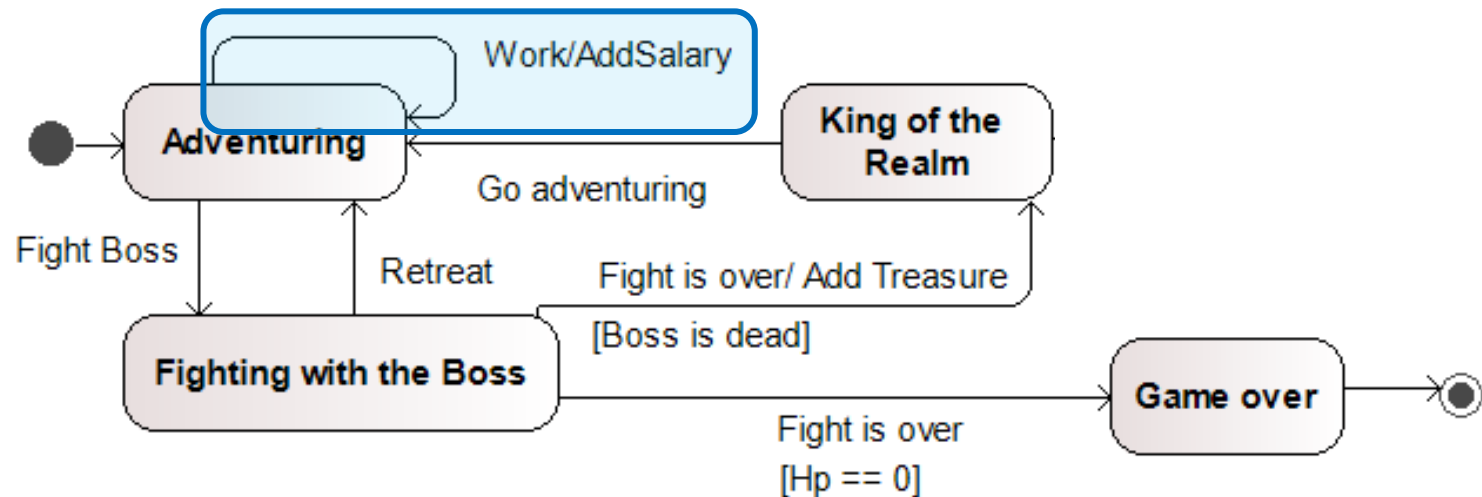
# Visszavonulás és királyság

- Lehesse visszavonulni a csatából!
- Ha győztünk, kapjuk meg a királyságot!
  - > Utána még lehesse kalandozni!



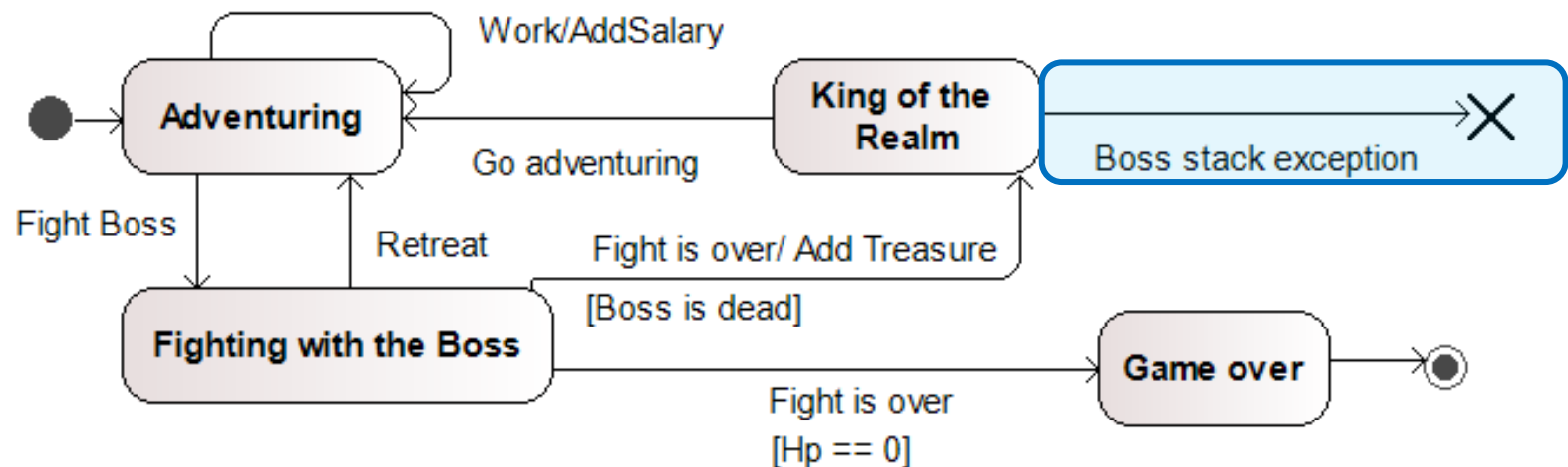
# Munka, munka, munka

- Lehessen dolgozni (kalandozóként)!
  - > A munkáért kapjunk pénzt!



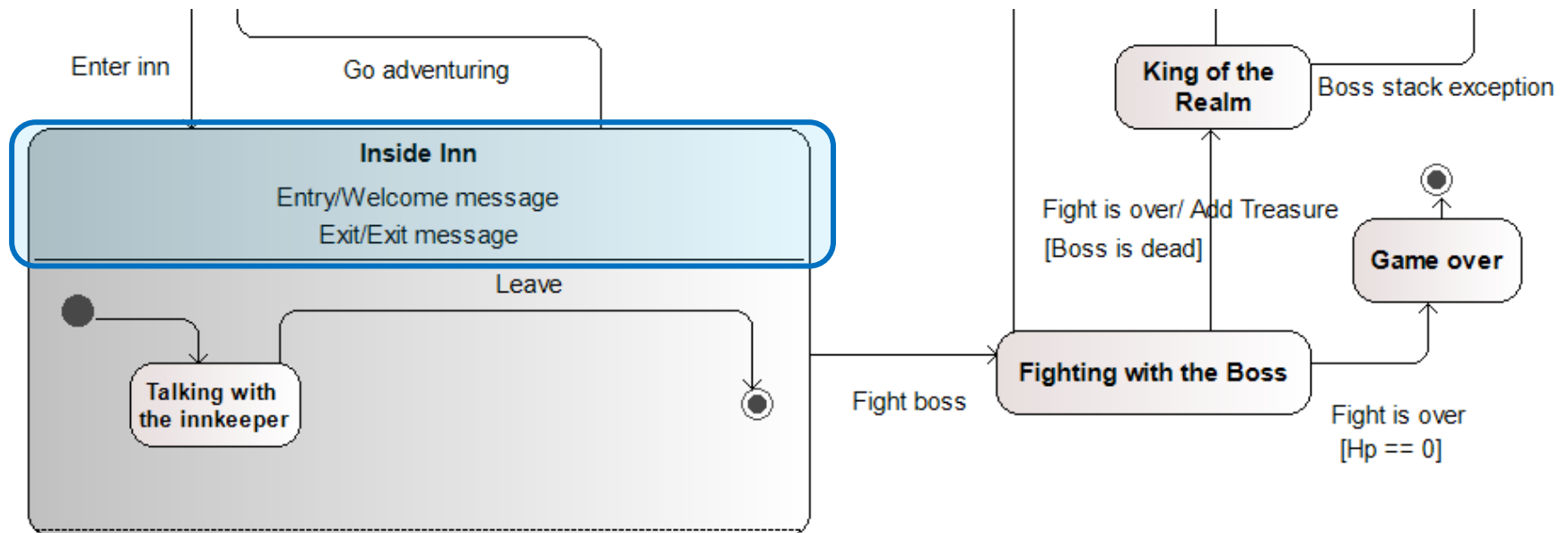
# Hibakezelés

- Ha nincs több főellenség, kapjunk hibát!
  - > Terminálási állapot (terminate pseudo state)
  - > Kilépési akciók, állapotváltások nélkül véget ér a végrehajtás



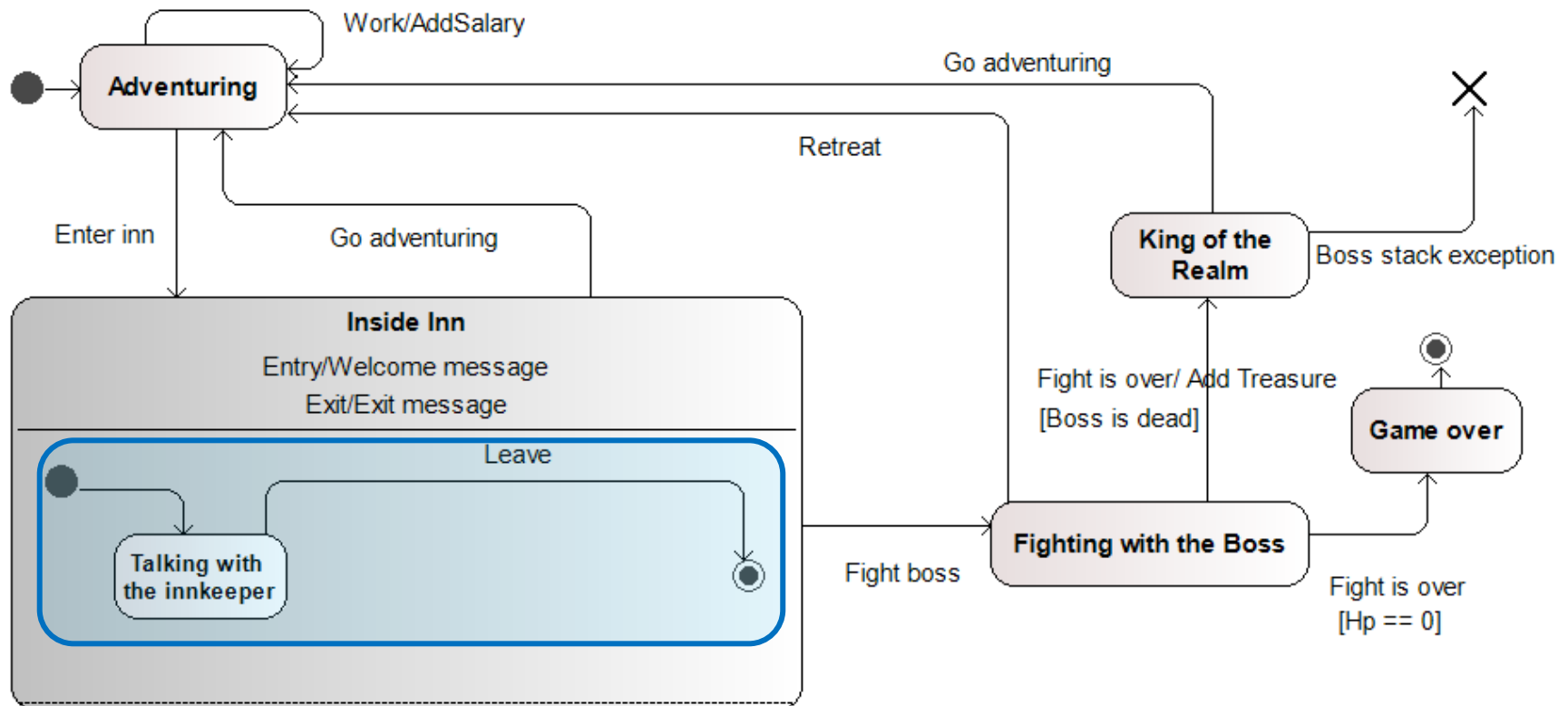
# Fogadó

- Minden kaland egy fogadóban kezdődik...
  - > Összetett állapot (composite state)
  - > Kapjunk üdvözlést az elején, elköszönést a végén!
    - Belépési akció (entry action): lefut, ha külső állapotból belépünk
    - Kilépési akció (exit action): lefut, ha külső állapotba kilépünk
    - Cselekvési akció (do action): folyamatosan fut a belépéstől a kilépésig (a példában ez nem szerepel)



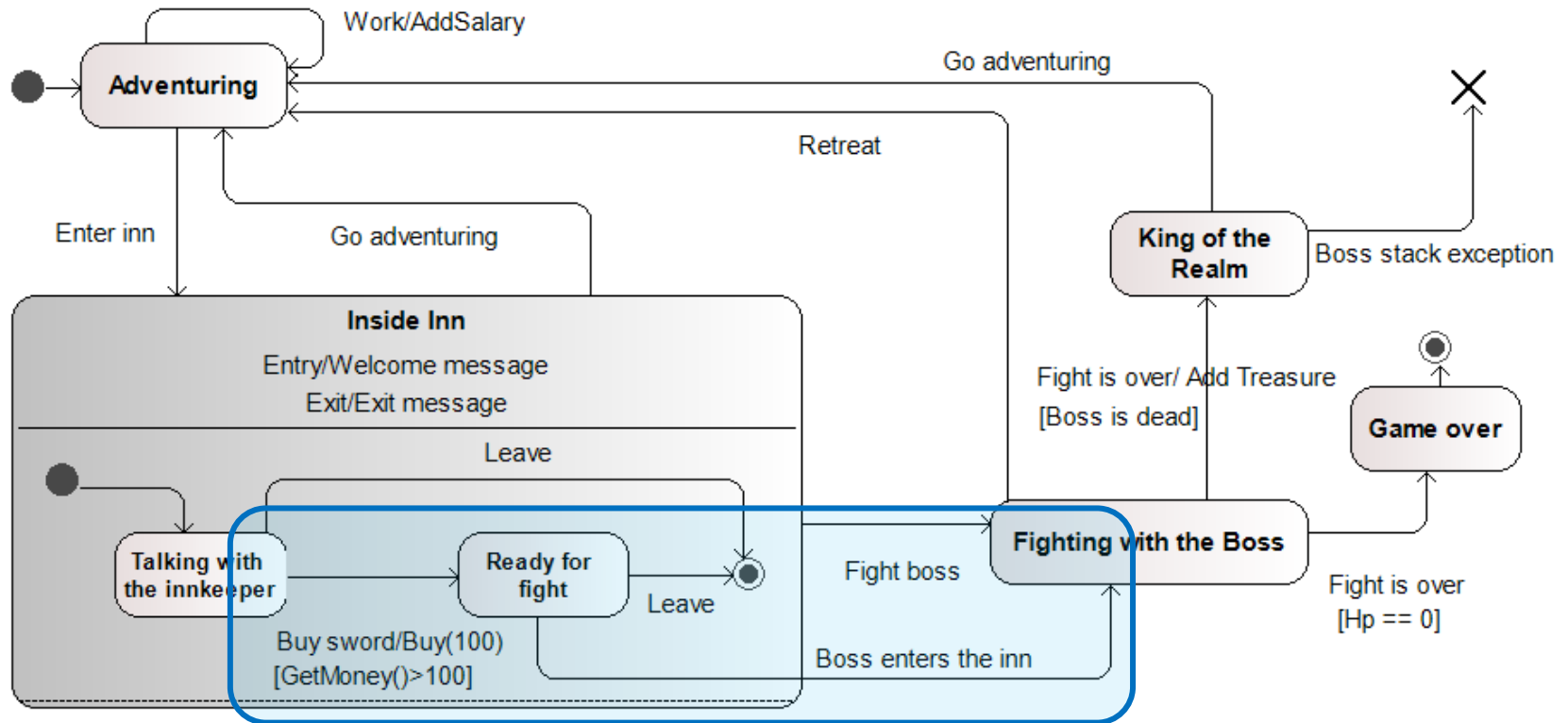
# Fogadó (folytatás)

- Lehesse beszélgetni a fogadóval!
  - > Belső állapotok/hierarchikus állapotgép



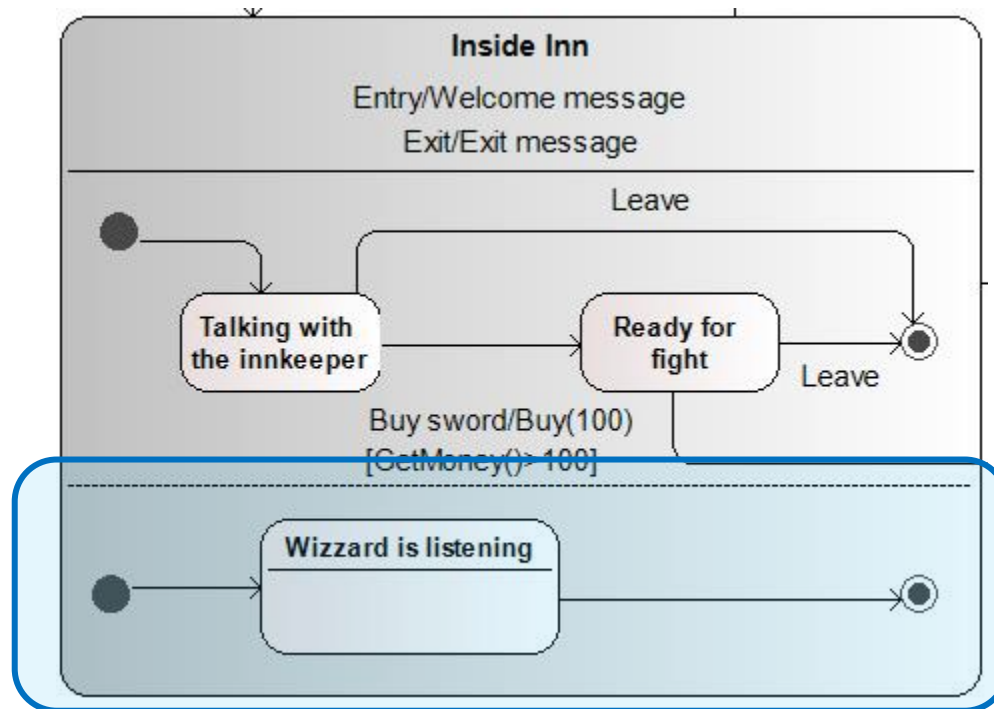
# Kard

- Lehesen venni kardot a fogadóstól!
  - > A kard kerüljön pénzbe!
  - > Ha megvettük, készen vagyunk a csatára
  - > Ha a főellenség ilyenkor betoppan, azonnal harcolni kell
    - Közvetlen kilépés



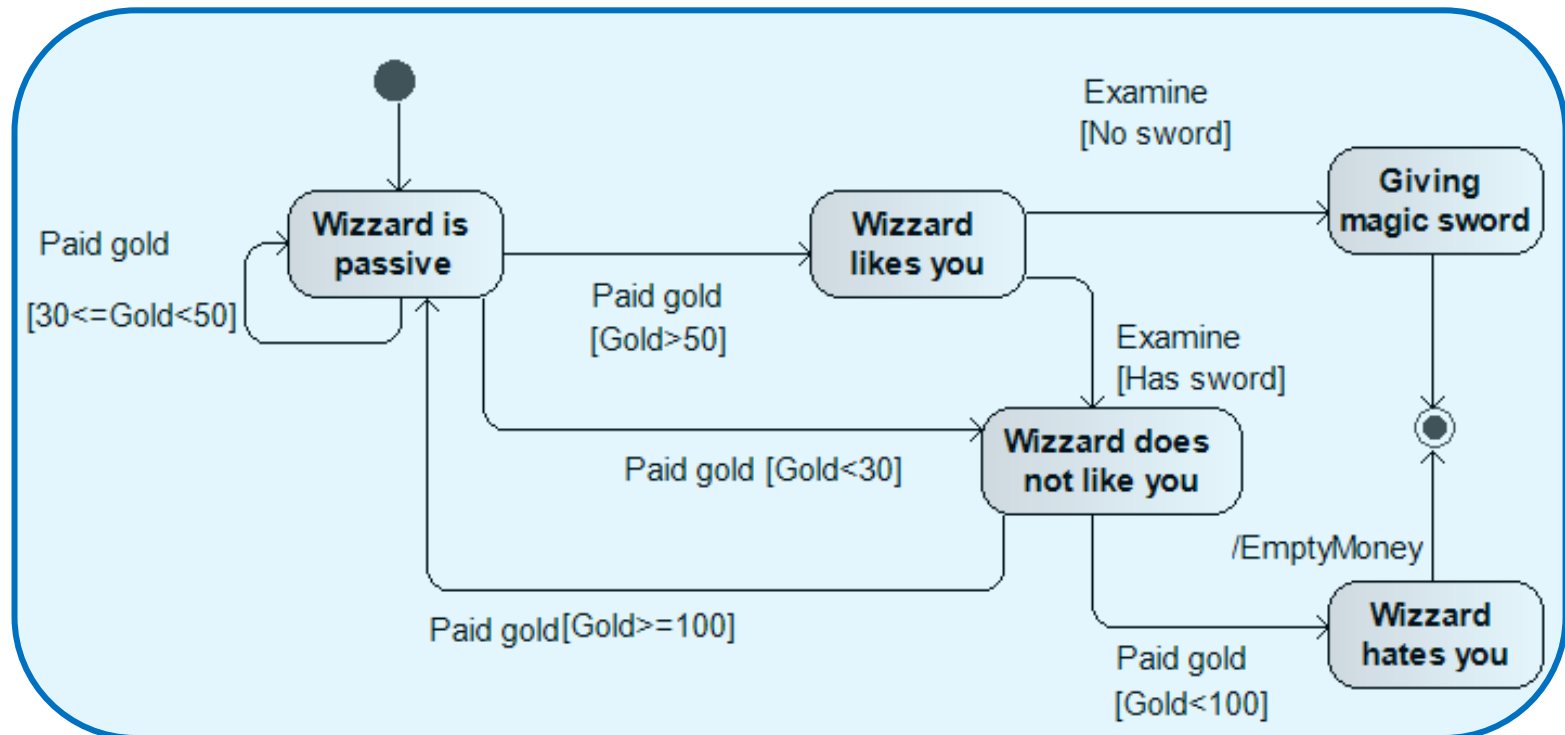
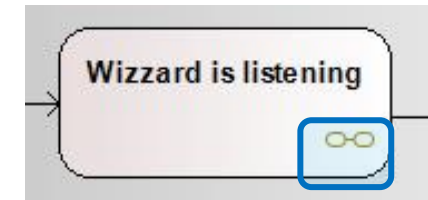
# Varázsló

- Minden kalandban van egy varázsló...
  - > Az állapotváltás a fogadós szállal *párhuzamosan* zajlik
  - > Régió (region)
    - Összetett állapot ortogonális szétbontása



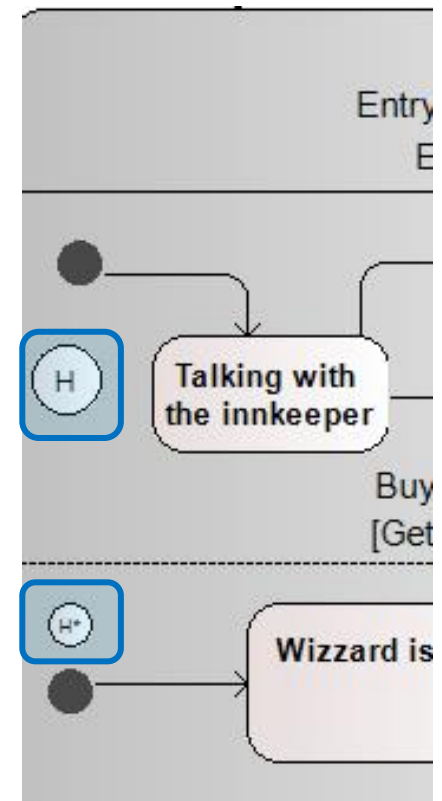
# Varázsló

- Lehesse lefizetni a varázslót... de csak okosan!
  - > Állapoton belüli állapotgép
  - > **Állapotgép állapot (submachine state)**
  - > Hivatkozás a másik összetett állapotra



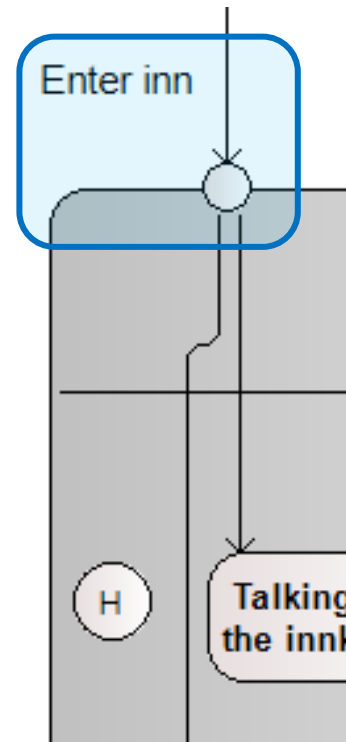
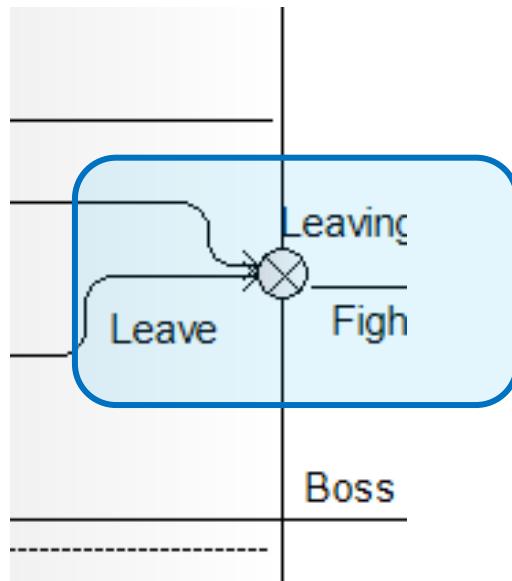
# Emlékezet

- Emlékezzon a rendszer, hol tartottunk a fogadóban, ha kimentünk és visszamentünk!
  - > **Sekély történet (shallow history) (H)**
    - Visszaállítja a közvetlen tartalmazó állapotgép/régió állapotát visszatéréskor
    - Csak egy hierarchia szinten keresztül működik
  - > **Mély történet (deep history) (H\*)**
    - Visszaállítja a tartalmazó állapotgép/régió állapotát visszatéréskor
    - Bármilyen mély hierarchiában működik



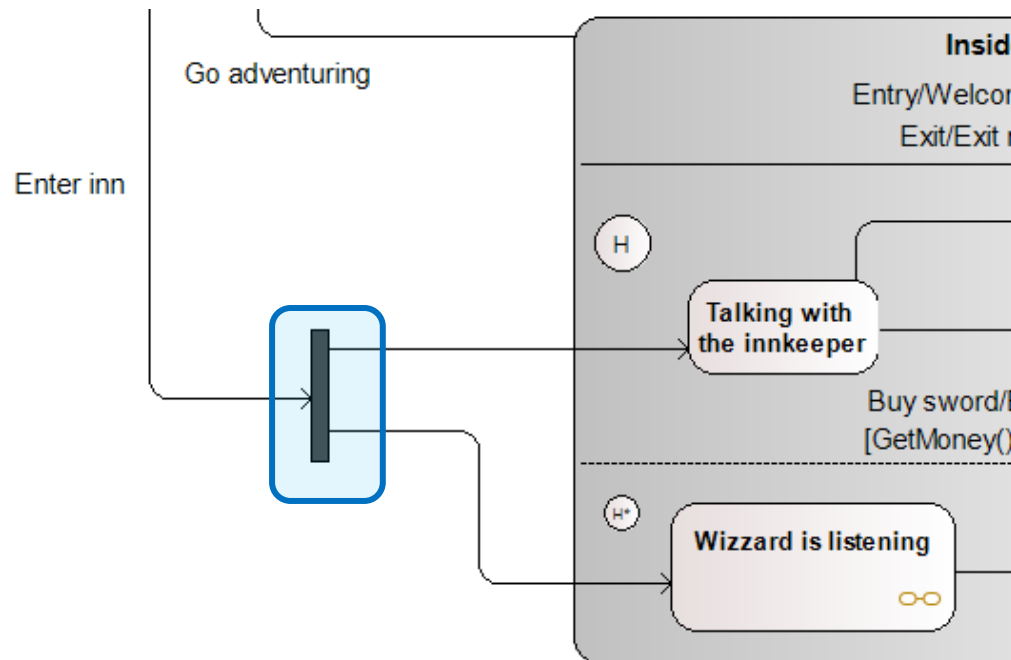
# Belépési és kilépési pont

- Megadható külön nevesített be-, és kilépési pont
  - > Belépési/kilépési pont (entry/exit point)
  - > Segíthet, ha különböző forgató- könyvekben különböző lépések kellenek
  - > Elrejtja a belső szerkezetet



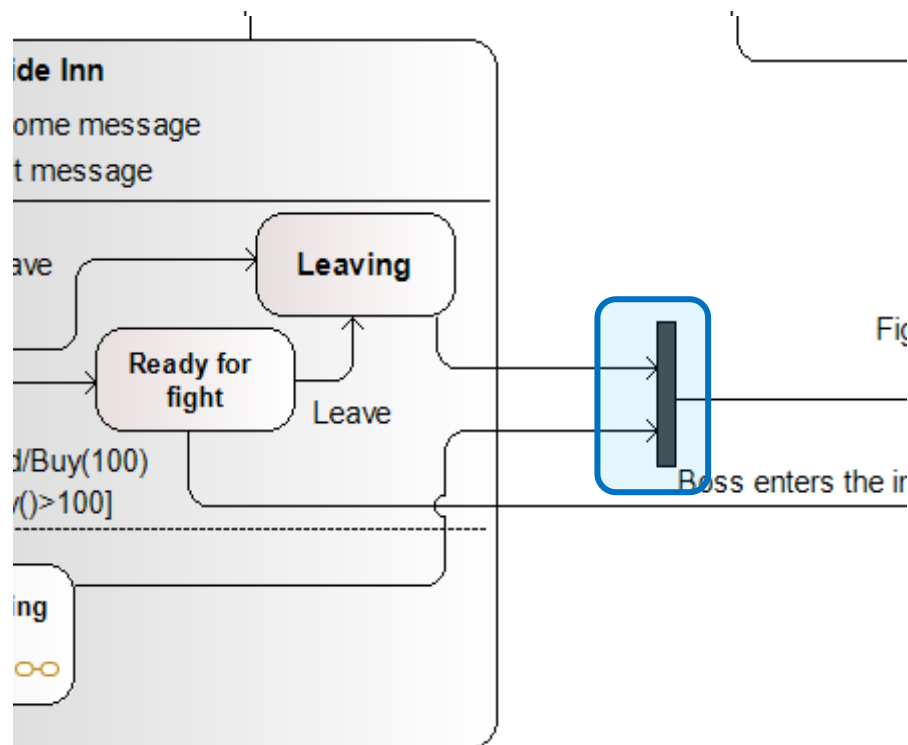
# Elágazás és egyesítés

- Elágazó útvonalak
  - > Elágazás (fork)
  - > Összetett állapothoz, több régióval
  - > Egyszerre minden állapot aktiválódik
  - > Kimenő élek kötelezően esemény (trigger) és őrfeltétel nélkül!



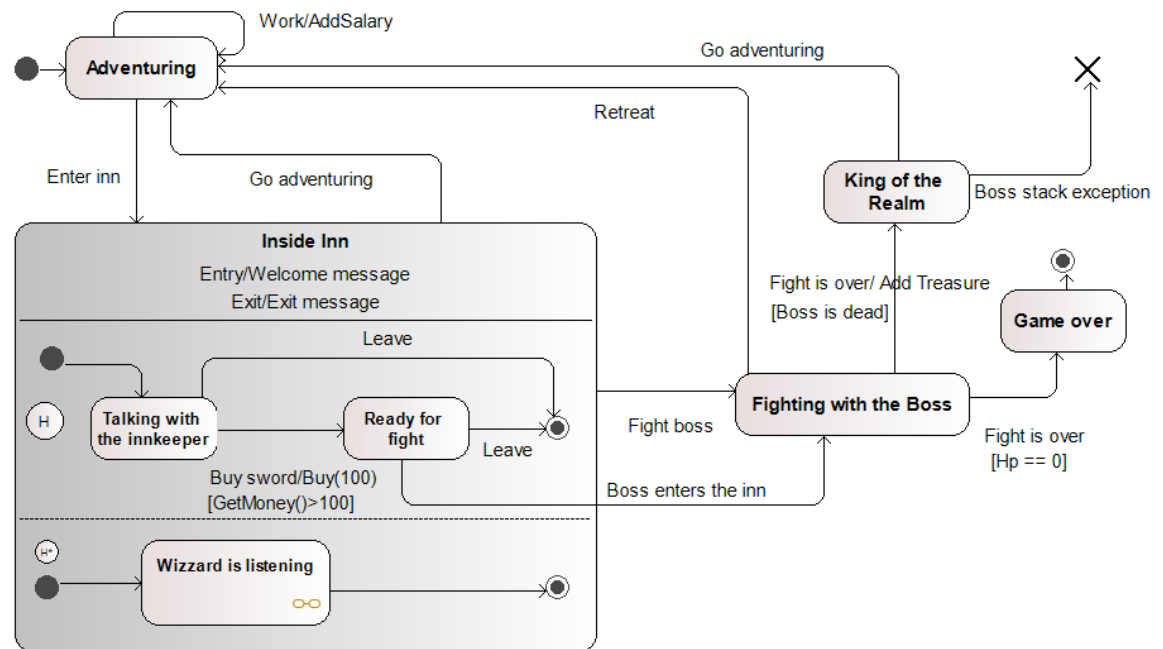
# Elágazás és egyesítés

- Összefutó útvonalak
  - > Egyesítés (join)
  - > Megvárja a részállapotokat és egységesíti őket
  - > Bemenő élek kötelezően esemény (trigger) és őrfeltétel nélkül!



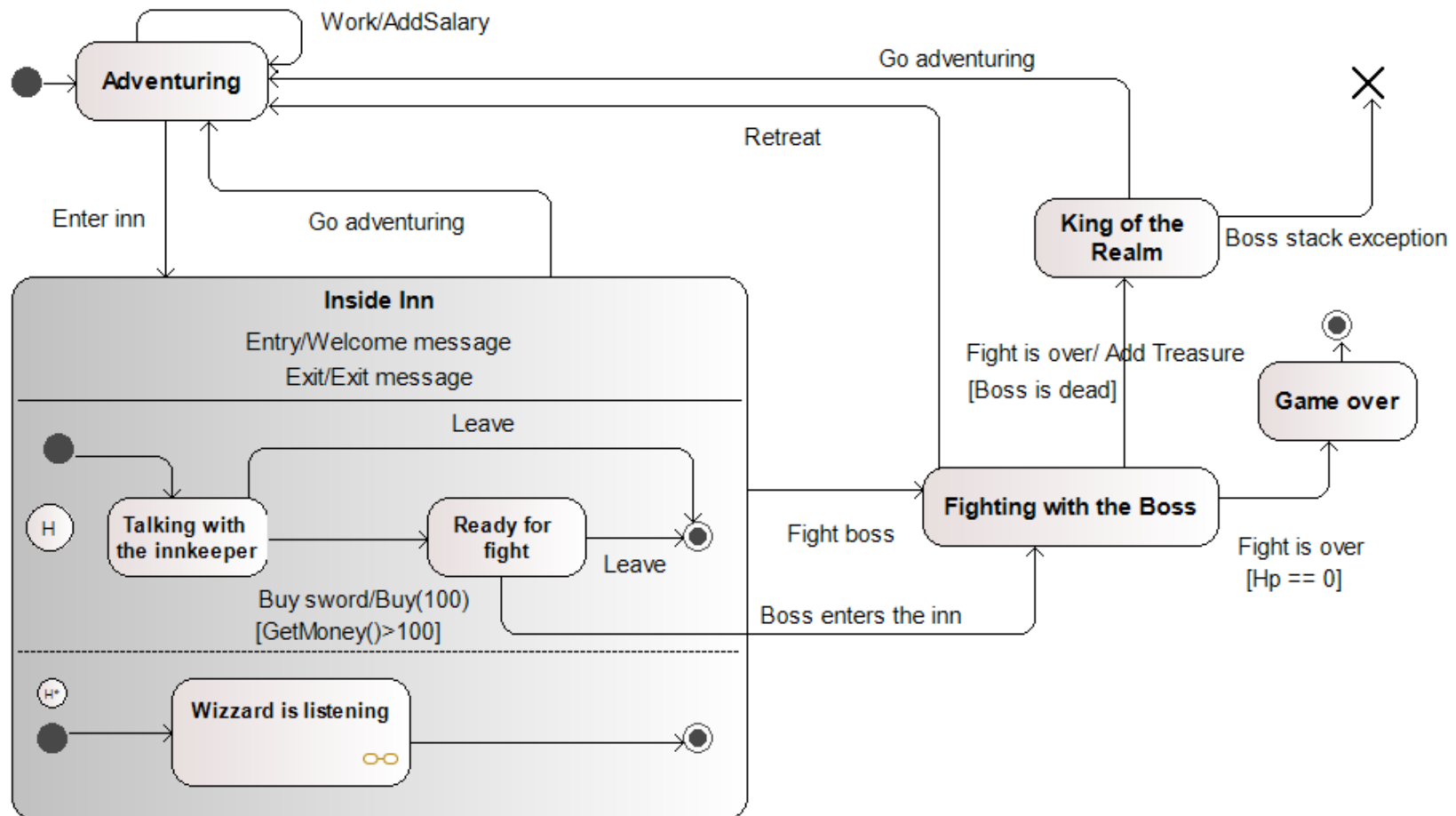
# Állapotgép

- Állapotok
  - Kezdő és végállapot
  - Terminálás
  - Átmenetek
    - > Esemény
    - > Örfeltétel
    - > Akció
  - Összetett állapotok
    - > Régiók
- > Történet elem
  - > Explicit be-, és kilépési pont
  - > Elágazás, egyesítés



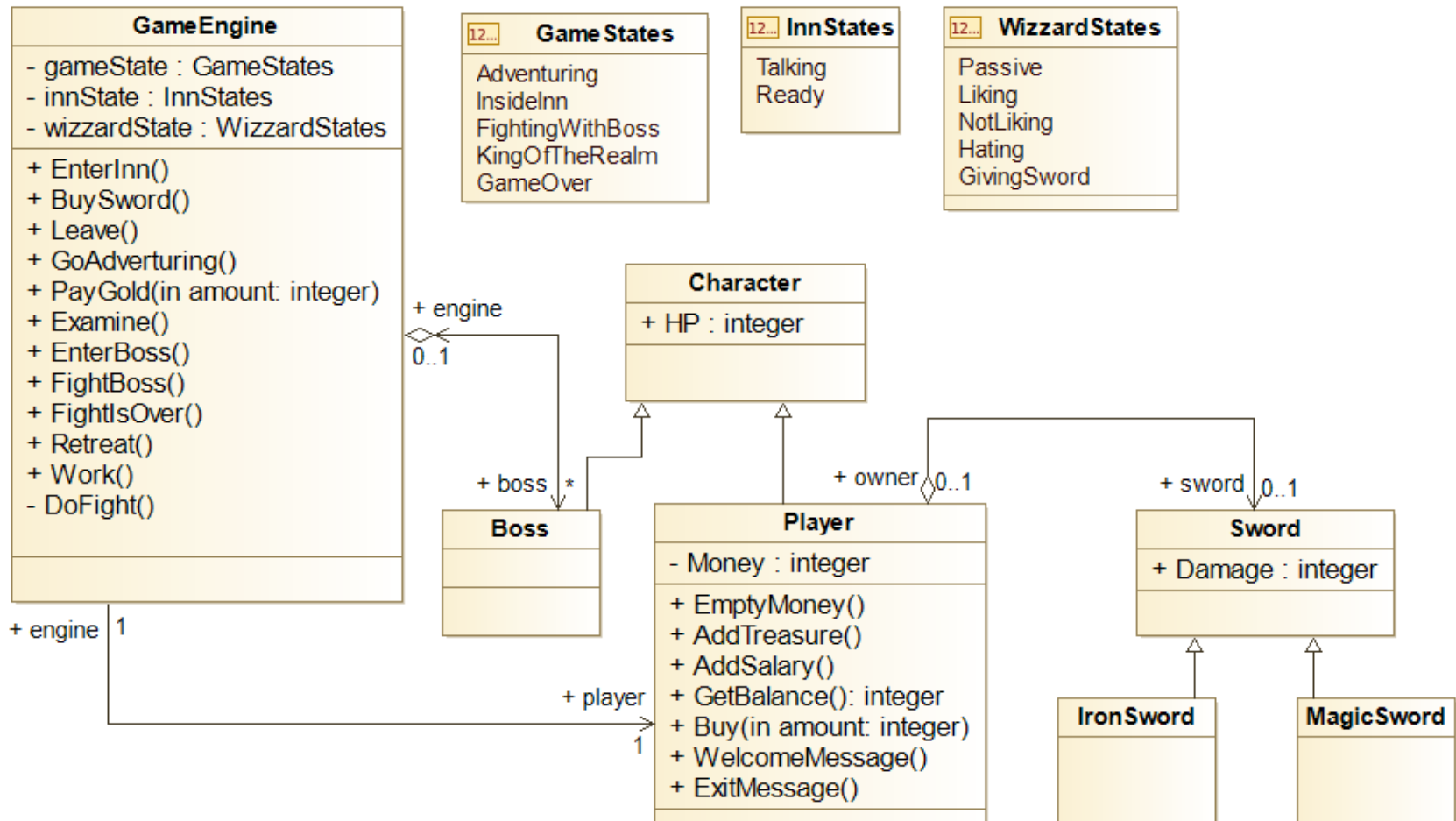
# Osztálydiagram

- Rajzoljuk meg az állapotgéphez tartozó osztálydiagrammot!



# Osztálydiagram

- Rajzoljuk meg az állapotgéphez tartozó osztálydiagrammot!



# Szekvencia-aktivitás-állapot?

- **Szekvenciadiagram**

- > Szekvenciális interakció objektumok közt
- > Metódushívások, üzenetek tényleges időbelisége

- **Aktivitásdiagram**

- > Vezérlési folyam: aktivitások/tevékenységek
- > Lehet szekvenciális, szétágazó, vagy párhuzamos
- > A vezérlés (flow) fontosabb az őrfeltételeknél

- **Állapotgép**

- > Állapotokat és állapotátmeneteket ír le
- > Események vezérlik az állapotátmenetet

# Köszönöm a figyelmet!

# Automata működése (+ pontért)

- Modellezzük egy megrendelés kezelő automata aktivitásait!
- A folyamat a megrendelés beérkezésétől indul.
- Elindítunk egy időzítőt, ami 3 nappal később lejár. Ha a folyamat további lépéseivel nem végzünk ez alatt az idő alatt, automatikusan hibát adunk és leállunk.
- Ha a megrendelt termék nincs raktáron, a folyamat végére ugrunk.
- A termék raktáron van, akkor behozzuk a raktárból, majd párhuzamosan (1) feladjuk a megrendelést és ezzel egyidőben (2) a számlaintézést is elindítjuk
- A számla kiállítását a számlázórendszer végzi, aminek át kell adnunk a szükséges adatokat, onnantól mindent elvégez nekünk. A számlázórendszer egy “Számlakész” eseménnyel jelzi, ha végzett.
- A sikeres megrendelés kezelése mindig egy „Köszönjük, hogy minket választott” üzenettel ér véget