Java és UML feladatok

Készítette: Simon Balázs, BME IIT, 2015.

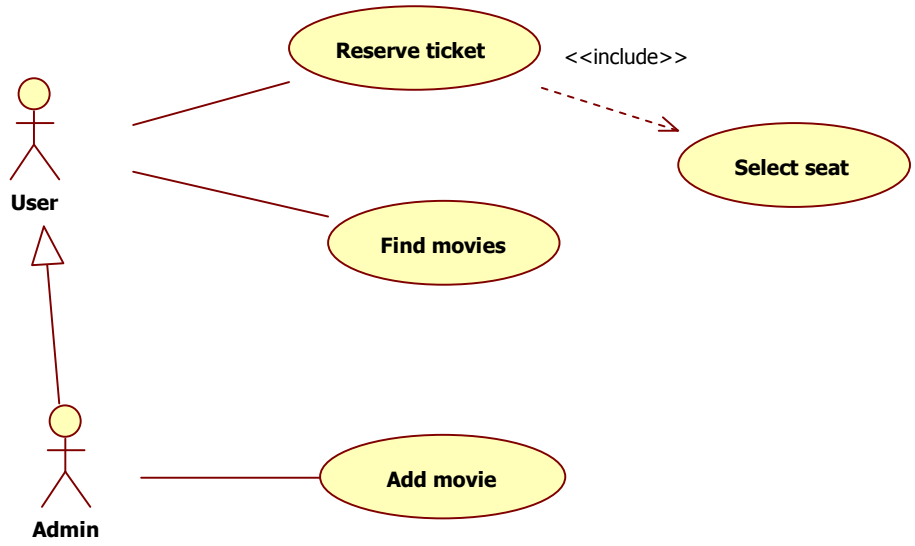
Bevezetés

A feladatok megoldásához a WhiteStarUML eszköz szükséges. Az eszköz a HSZK laborgépein már telepítve van.

Az első feladat független a többi feladattól. A másodiktól az ötödikig egymással összefüggőek a feladatok, és együtt kell értelmezni őket.

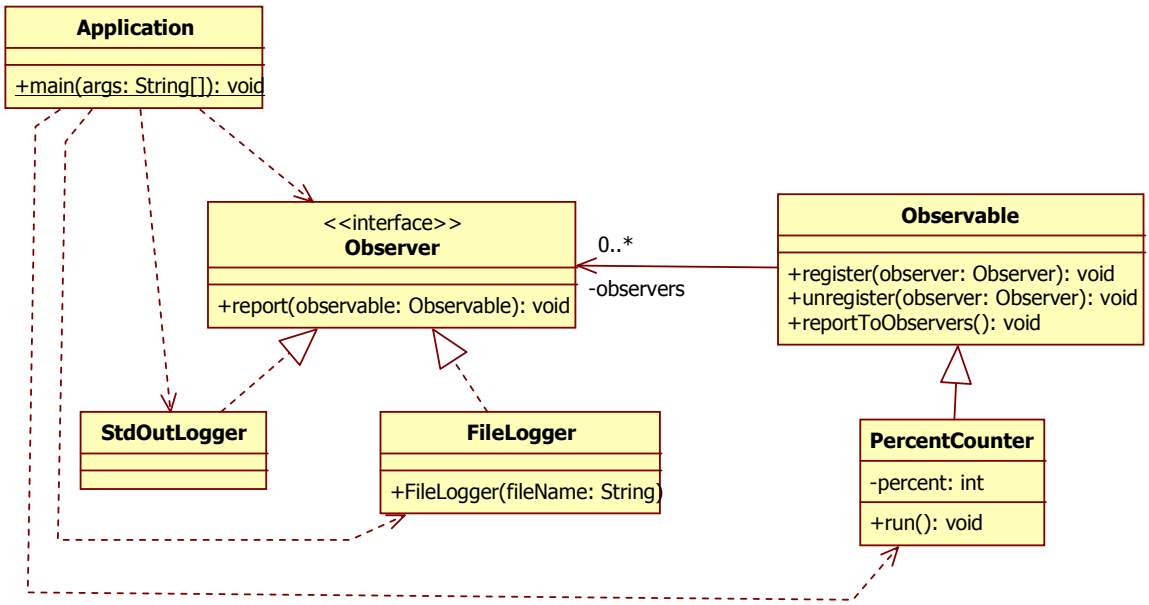
1 Use-case diagram

Rajzoljuk meg a következő use-case diagramot:



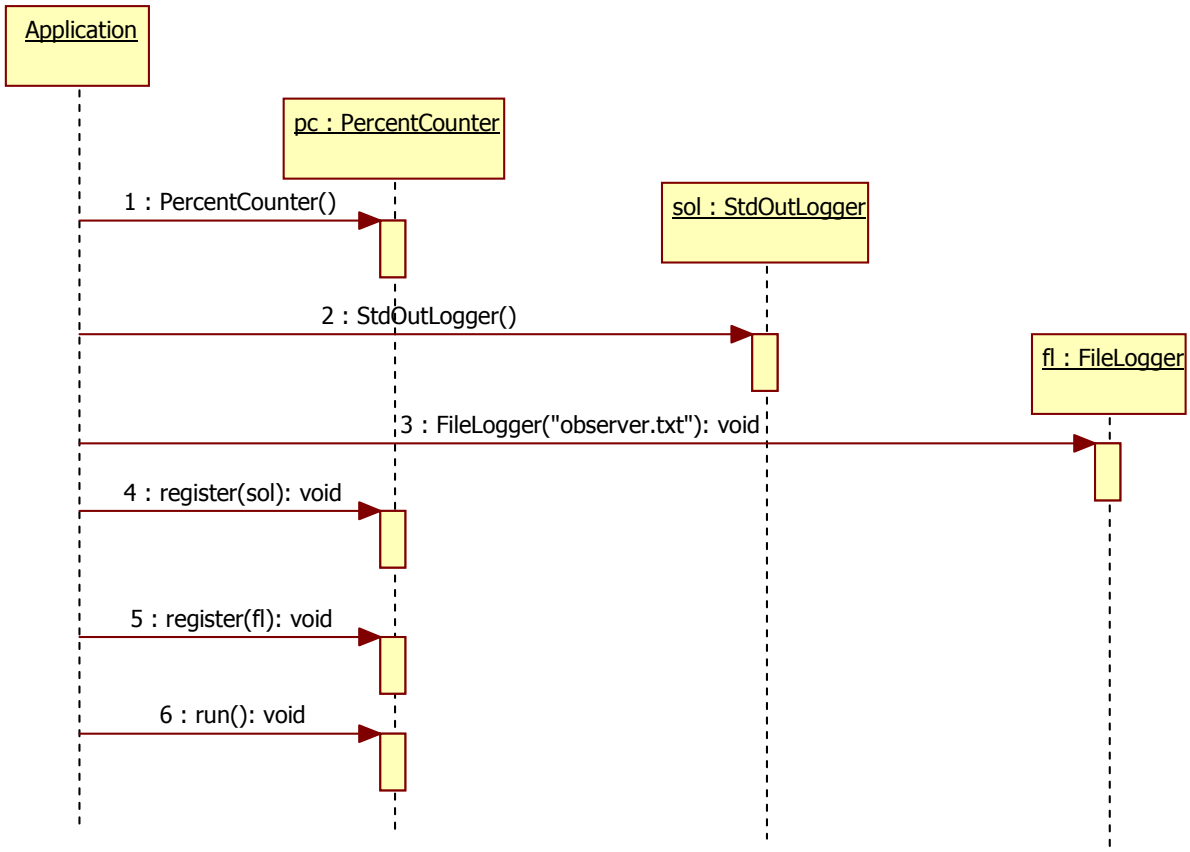
2 Osztálydiagram

Rajzoljuk meg a következő osztálydiagramot:



3 Szekvencia diagram

Rajzoljuk meg a következő szekvencia diagramot:



A konstruktorhívásokat sajnos csak hagyományos metódushívásként lehet ábrázolni.

A nyilak megrajzolásakor választhatunk az osztálydiagramon már szereplő metódusok közül is, így kisebb a hibázás lehetősége. Például a **register()** metódusnál:

A világoskék egyenlőségjel megnyomáskor kiválaszthatjuk egy listából a **register()** metódust.

A paraméterek és típusaik feltüntetéséhez a szekvencia diagram **MessageSignature** tulajdonságát állítsuk **NAMEANDTYPE**-ra. Ezt a **Model Explorer** és a **Properties** ablakok segítségével tehetjük meg.

4 Kódgenerálás

Hozzunk létre egy üres Java projektet az Eclipse-ben, és az src könyvtárába generáljuk a Java forrásokat a WhiteStarUML segítségével. (Segítségképpen lásd a tutorial utolsó fejezetét.)

5 Implementáció

Implementáljuk a keletkezett metódusokat úgy, hogy a működés megfeleljen az alábbiaknak! Sajnos a WhiteStarUML generátora nem túl jó, így szükség esetén módosítsuk a generált kódot!

Az ábrán látható alkalmazás az Observer tervezési mintát valósítja meg, amely lényege, hogy a megfigyelő (**Observer**) objektumok értesítést kapnak a megfigyelt (**Observable**) objektumok állapotváltozásairól.

Az **Observable** osztály metódusainak jelentése a következő:

- **register(observer:Observer)** – egy megfigyelő hozzáadása az értesítendő objektumok listájába (**observers** asszociációvég)
- **unregister(observer:Observer)** – egy megfigyelő törlése az értesítendő objektumok listájából (**observers** asszociációvég)
- **reportToObservers()** – az összes beregisztrált megfigyelő objektum értesítése az **Observer** interfész **report(observable:Observable)** metódusán keresztül (paraméterként a **this**-t kell átadni)

Az **Observer** interfész **report(observable:Observable)** metódusának megvalósításaiban kell reagálni a megfigyelt objektumok állapotváltozásaira. A megfigyelt objektumot a metódus paraméterként kapja meg.

Jelen esetben a **PercentCounter** osztály példányait kell megfigyelni. Ennek az osztálynak a feladata, hogy a **run()** metódusban a **percent** attribútumban elszámoljon 0-tól 100-ig egyesével, és amikor az attribútum értéke 10-zel osztható, akkor értesítse erről az ő megfigyelőit. A **toString()** függvény visszatérési értéke a percent attribútum tartalma és a százelékjel konkatenációja, pl. “10%”.

Kétfajta Observer-t kell megvalósítani. Az egyik az **StdOutLogger**, amely a **report(observable:Observable)** metódusban kapott objektum **toString()** metódusának eredményét kiírja a sztenderd outputra. A másik a **FileLogger**, amely a kapott objektum **toString()** metódusának eredményét a konstruktorban megadott fájlba írja. Mindkét megfigyelő esetén az eredményeket soronként kell kiírni. Fájl kimenet esetén a stream-et soronként flush-ölni kell. A fájlt reprezentáló attribútumot nem kell felvenni a diagramon, ezt majd elegendő kézzel beleírni a generált kódba.

Az **Application** osztály a főprogram. Ennek a **main(args:String[])** statikus függvénye a szekvencia diagramnak megfelelően működik: létrehoz egy-egy példányt a **PercentCounter**, **StdOutLogger** és **FileLogger** osztályokból, a megfigyelőket beregisztrálja a **PercentCounter** objektumba, végül pedig meghívja annak **run()** metódusát.