Feladat

A **turisták** látogatása bevételt hoz egy városnak, miközben kis mértékben rontja annak állapotát. Egy város, ami jó állapotban van, vonzza a turistákat, míg a rossz állapotú város taszítja az odalátogatni készülőket.

Egy város állapotát 1-100-ig értékeljük: 1 – 33: lepusztult; 34 – 67: átlagos; 67 – 100: jó. A turistáknak 3 fajtája van: japánok, akik nem rontanak a város állapotán (rendet raknak maguk után); a nyugati országokból érkező turisták, akik minden 100 fő esetén egy-egy pontot rontanak a város állapotán (kevésbé ügyelnek a környezetükre), és a többiek, akik minden 50 fő esetén rontanak egyegy pontot a város állapotán (a szemetelés kulturális szokásnak tekinthető).

Egy turista látogatása 100.000 Ft bevételt hoz a városnak. Ha a város ebből származó összes bevétele egy évben meghaladja az 20 milliárd forintot, akkor a többletet a város javítására és szépítésére fordítják: ez ötvenmillió forintonként egy pont állapotjavulást eredményez.

Ha a város jó állapotban van, akkor 20%-kal több japánt és 30%-kal több nyugatit vonz, mint ahányan azt az év elején jelezték. Átlagos állapotban 10%-kal több nyugati, és 10%-kal több egyéb turista jön az előzetes várakozáshoz képest. Lepusztult állapotban a japánok egyáltalán nem jönnek, a többiek pedig csak annyian, ahányan azt az év elején jelezték.

Készítsen használati eset diagramot, ahol a turisták és a város szempontjából lényeges eseteket, valamint ezek kapcsolatát ábrázolja. Adjon meg olyan szekvencia diagramot, amely a városvezetés és a város közötti kommunikációt: a város metódusai hívásainak sorrendjét jeleníti meg. Rajzolja fel a város állapotgép diagramját! Készítse el az osztály diagramot! Használjon állapot és látogató tervezési mintákat.

Implementálja a modellt, és oldja meg az alábbi feladatot: ***Adja meg, hogy hányadik évben volt a legjobb a város állapota, de írja ki évenként a turisták számát (a tervezett és a tényleges számot) kategóriák szerint, az éves bevételt, és a város új állapotát (szám és kategória) is!***

A program egy szövegfájlból olvassa be az adatokat! Az első sorban a város kezdeti állapotát mutató pontszám (egész szám) szerepel. A többi sor azt tartalmazza, hogy az egymás utáni években hány turista tervezte, hogy eljön a városba. Minden sor 3 darab egész számból áll: az utazást tervező japán, nyugati, és egyéb turisták számait mutatja. A program kérje be a fájl nevét, majd jelenítse is meg a tartalmát. (Feltehetjük, hogy a fájl formátuma helyes.) Egy lehetséges bemenet:

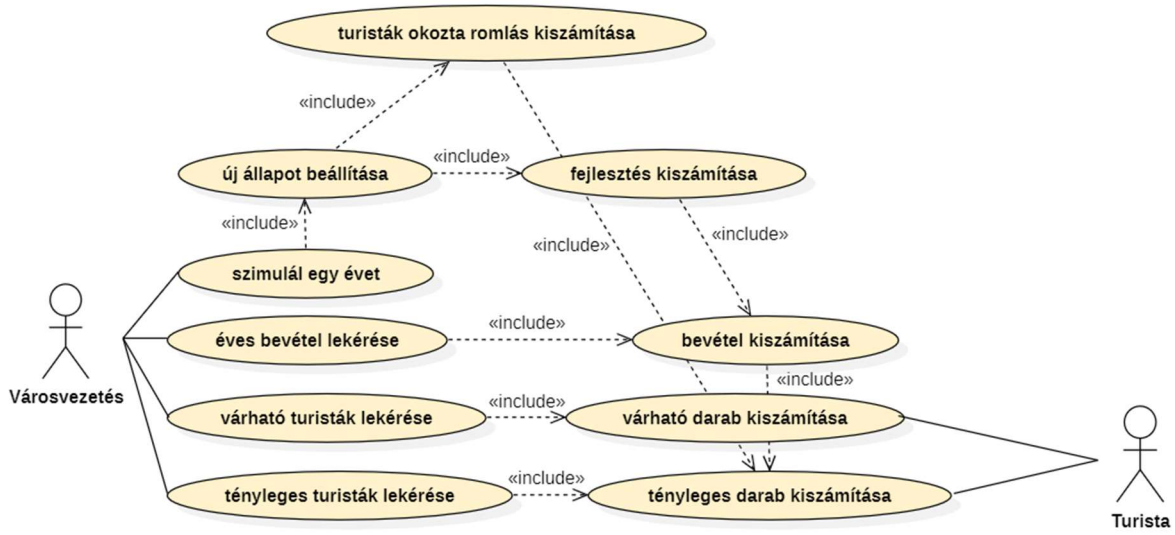
```
50
1000 4000 6000
2000 3000 8000
6500 5000 3000
```

Készítsen teszteseteket, és hozzon létre ezek kipróbálására automatikusan tesztkörnyezetet!

Terv

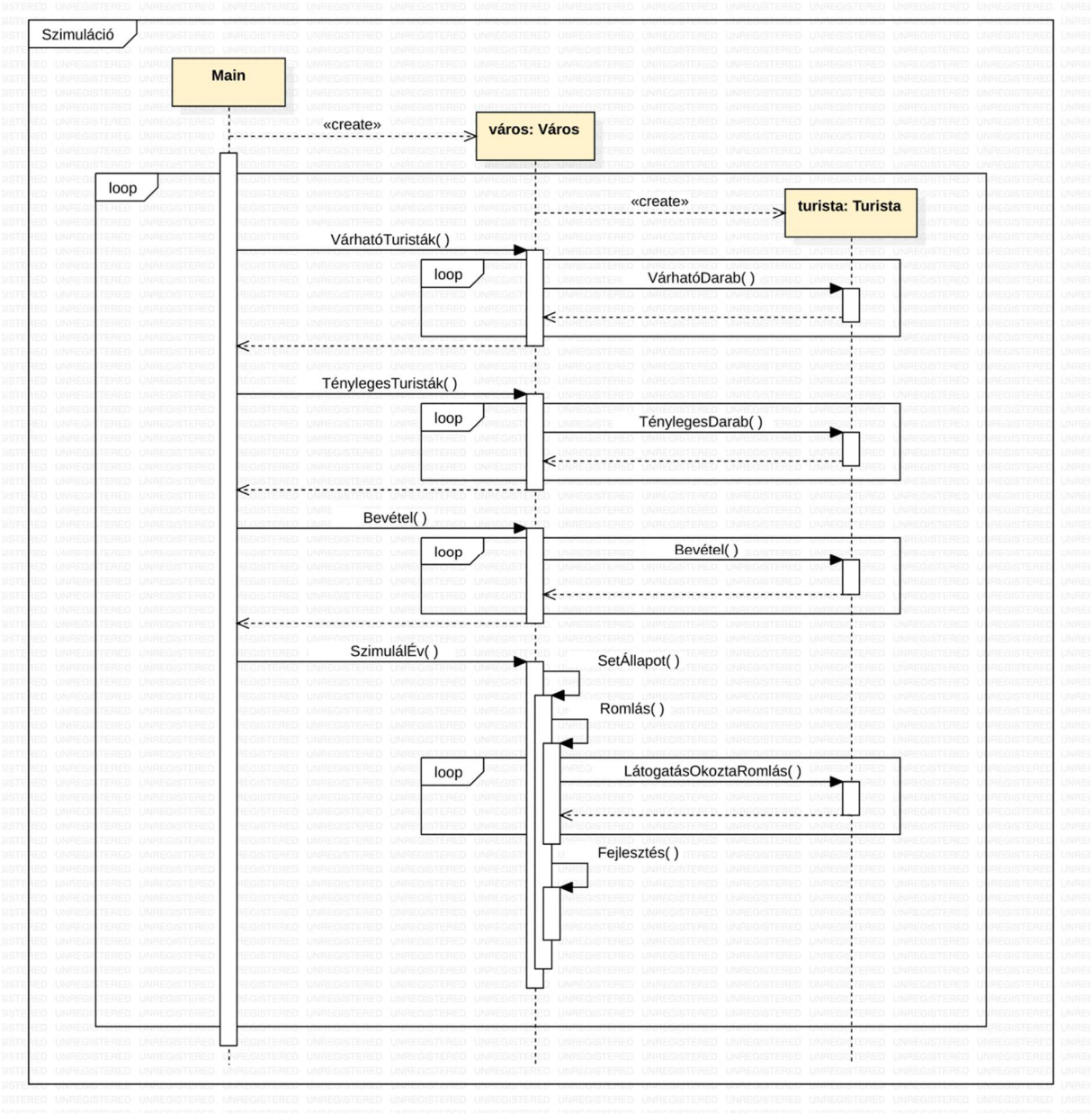
Használati eset diagram:

A feladat megoldásához szükséges megvalósítani azokat metódusokat, amivel a *városvezetés* lekérheti az adott évben várható és a ténylegesen érkező turisták számát továbbá az adott éves bevételt is. Ezenkívül szükséges egy metódus amivel a városvezetés leszimulálhatja az adott évet, így beállítva a város új állapotát.



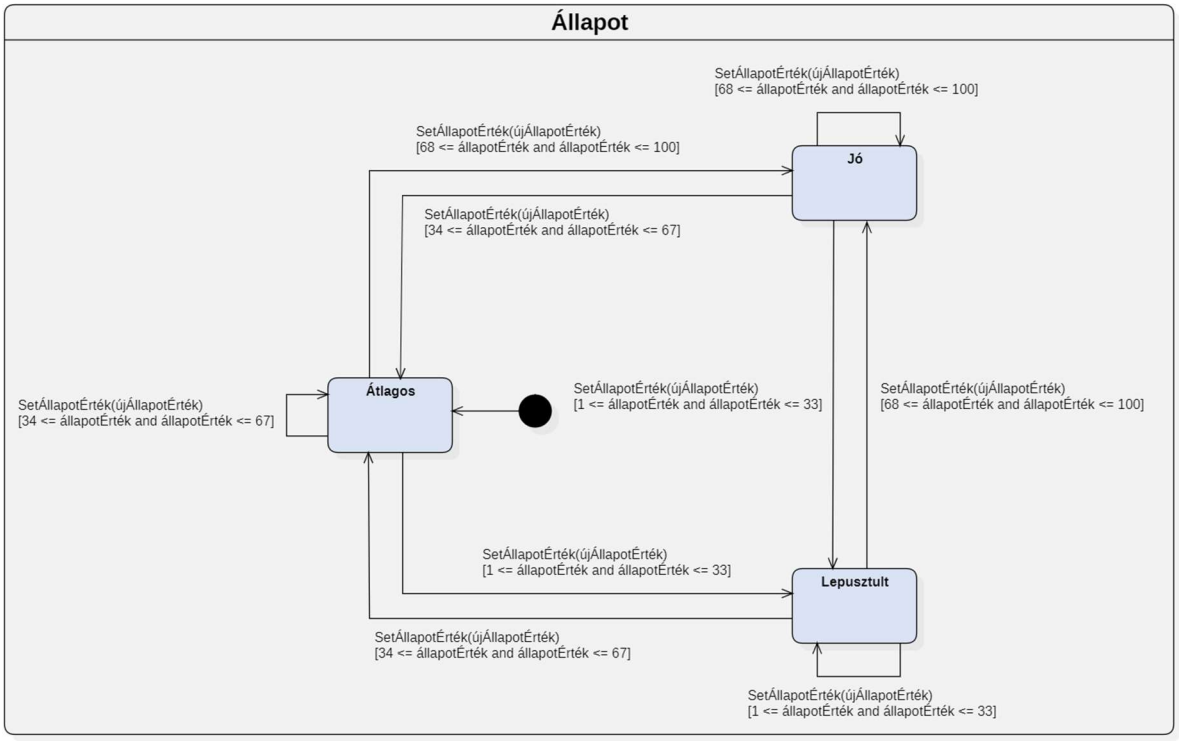
Szekvencia diagram:

A városvezetés egymás után ciklikusan lekéri az adott évi várható és tényleges turistákat illetve az éves bevételt. Ezek a metódusok az adott évben várható turisták és a város állapotából kiszámolják a szükséges adatokat és ezeket összegezve visszaadják a kapott értékeket a városvezetésnek. Végül pedig megtörténik az év szimulációja, így a városnak egy új állapot kerül beállításra. Ezeket ciklikusan ismételve megkapjuk a város több éves szimulációját. Ezt írja le az alábbi szekvencia diagram.



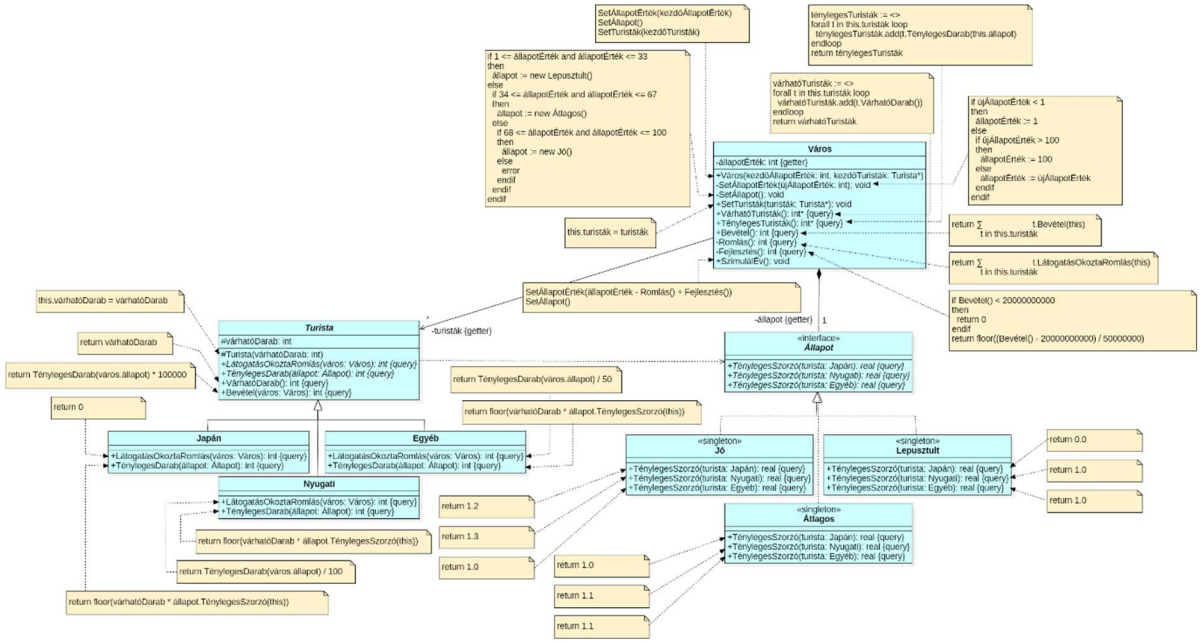
Állapotgép diagram:

A városnak három állapota lehet: lepusztult, átlagos, jó. Ez az állapot minden egyes év végén változik az adott évben látogató turisták számától és a város év eleji állapotától függően. Változás akkor történik ha a város állapota *(állapotÉrték)* egy másik állapothoz tartozó intervallumba kerül.



Osztály diagram:

A város különböző állapotait az *Állapot* interface megvalósításával kapjuk meg. Emellett a különböző turistákat a *Turista* absztrakt osztályból való származtatásával hozzuk létre. A különböző állapotok metódusai az állapot tervezési mintát használják, míg a turisták *TénylegesDarab()* metódusának megvalósításához látogató tervezési mintát alkalmazzuk. A különböző állapotok nem tartalmaznak adattagokat, így lehetnek az egyke tervmintának megfelelőek. A *Város* osztály tárolja a városunk állapotát és tartalmazza azokat a különböző metódusokat, amik a feladatok megoldásához szükségesek.



Tesztelési terv

1. *Turista* osztályok tesztelése
    - *LátogatásOkoztaRomlás()* metódus tesztelése különböző állapotú városokkal, mindegyik *Turista* osztályon
    - *TénylegesDarab()* metódus tesztelése különböző állapotokkal, mindegyik *Turista* osztályon
    - *VárhatóDarab()* metódus tesztelése mindegyik *Turista* osztályon
    - *Bevétel()* metódus tesztelése különböző állapotú városokkal, mindegyik *Turista* osztályon
2. *Állapot* osztályok tesztelése
    - *TénylegesSzorzó()* metódus tesztelése az összes *Állapot* interface-t megvalósító osztályban paraméterként mindegyik *Turista* osztállyal
3. *Város* osztály tesztelése
    - *Kontstruktor* tesztelése különböző értékekkel, köztük kritikusakkal is (pl.: -1, 0, 33, 34, 100, 101)
    - *VárhatóTuristák()* metódus tesztelése különböző *turisták* listákkal (pl.: üres lista, egy elemű lista, több elemű lista)
    - *TénylegesTuristák()* metódus tesztelése különböző *turisták* listákkal (pl.: üres lista, egy elemű lista, több elemű lista)
    - *Bevétel()* metódus tesztelése különböző *turisták* listákkal (pl.: üres lista, egy elemű lista, több elemű lista)
    - *SzimulálÉv()* metódus tesztelése különböző *turisták* listákkal (pl.: üres lista, egy elemű lista, több elemű lista) és különböző *állapotÉrték* értékekkel