**Feladat**

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

Egy turista látogatása átlagosan 100.000 Ft bevételt hoz a városnak. Ha a város bevétele egy évben meghaladja az egy milliárd forintot, az egy milliárdon felüli részt a város javítására és szépítésére fordítják, hogy több látogató érkezzen a következő évben. A város állapota 1 és 100 pont között mozog (1 alá és 100 fölé sose megy, mert az állam elkölti a fölösleget és besegít, ha már nagyon vészes a helyzet). 1 és 33 közt számít lepusztultnak, 34 és 67 között átlagosnak és 67 fölött jó állapotúnak. Minden évben egy milliárd forint bevétel fölött minden húszmillió forint hoz egy pont állapotjavulást a városnak.

A turisták 3 fajtába sorolhatók: a japánok rendet raknak maguk után, így ők nem rontják a város állapotát. A modern országokból érkező turisták kevésbé ügyelnek a környezetükre: 100-asával rontanak egy-egy pontot a város állapotán. A harmadik csoportba sorolható turisták azon országok képviselői, ahol a szemetelés kulturális szokásnak tekinthető, ők 50-esével rontanak egy-egy pontot a város állapotán.

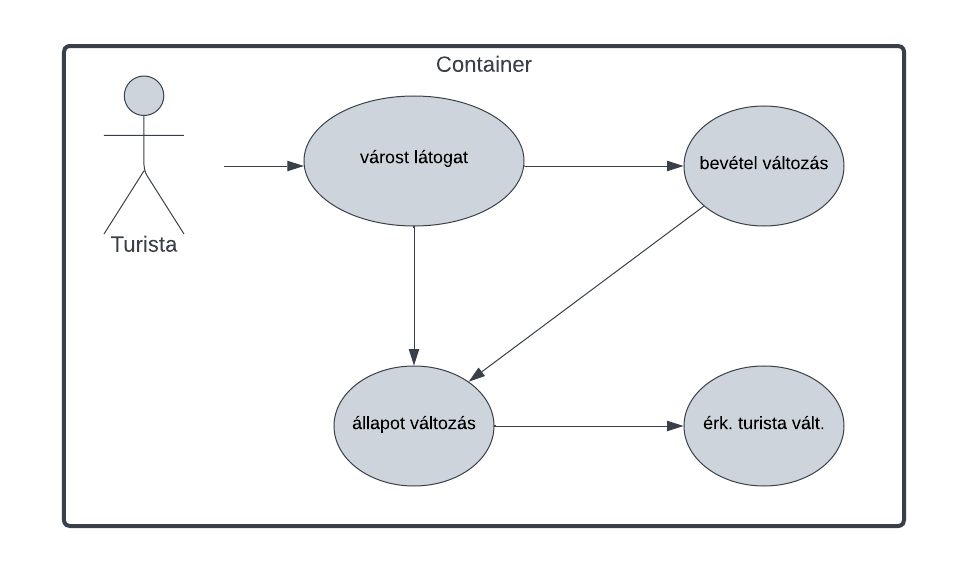
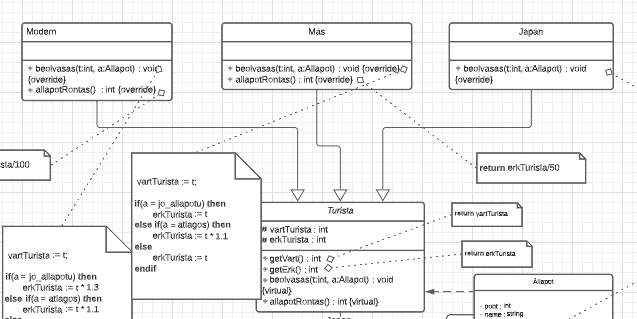
Ha a város jó állapotban van, abban az évben 20%-kal több japánt és 30%-kal több modernt vonz, mint ahány tervezte, hogy ellátogat oda. Átlagos állapotban 10%-kal több modernt és 10%-kal több harmadik típusú turistát vonz. Lepusztult állapot esetén a japánok egyáltalán nem jönnek, a többiek pedig annyian, amennyien tervezték.

**Adjuk meg, hogy 10 év letelte után milyen a város állapota! Körönként mutassuk meg az érkezett turisták számát (hány tervezett és hány jött) kategóriák szerint, az éves bevételt és a város felújítás előtti állapotát (szám és kategória)!**

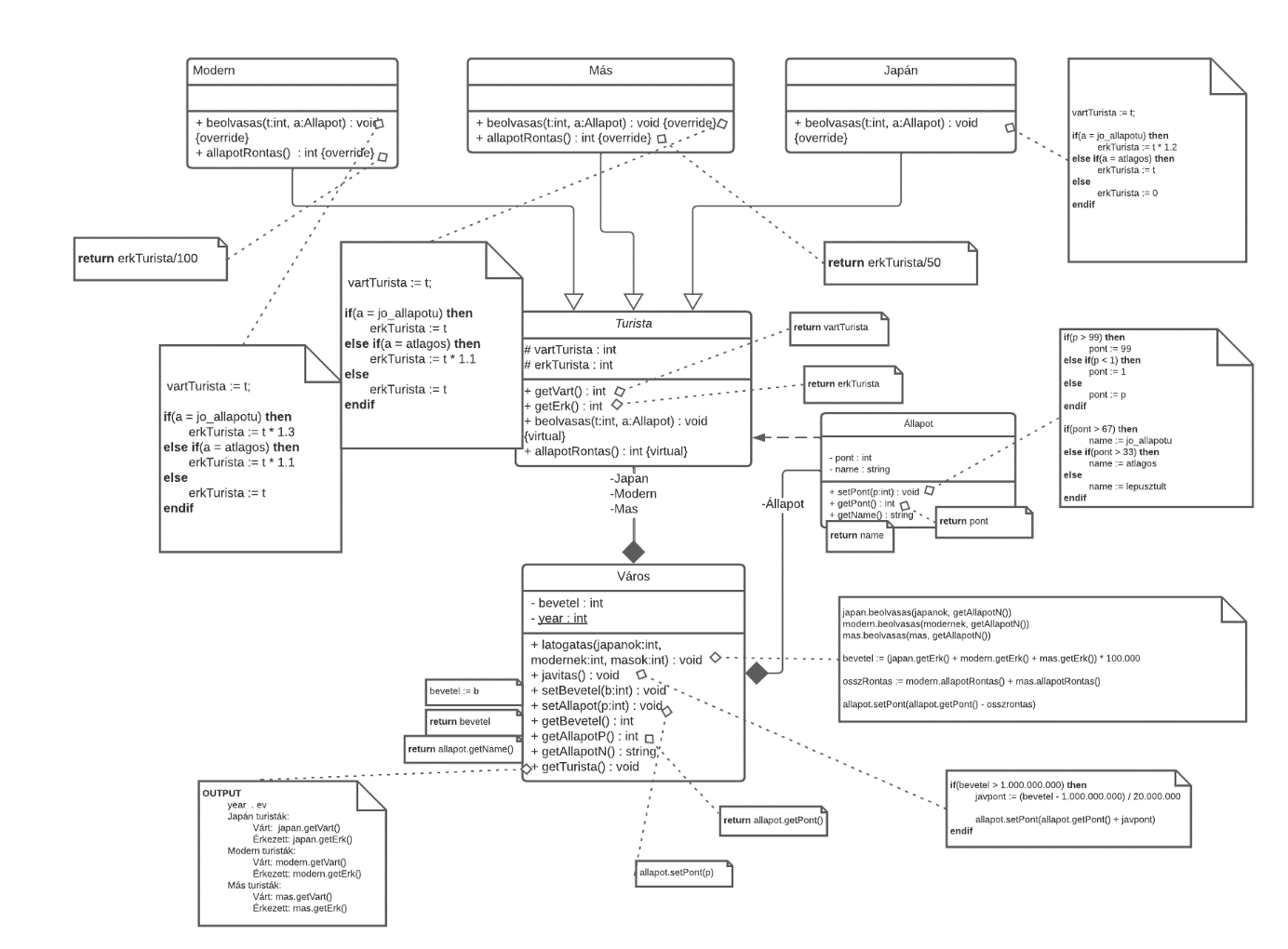
A program egy szövegfájlból olvassa be az adatokat! Az első sorban a város kezdeti állapota szerepel. A második sor jelöli a szimulált évek számát. A következő sorok tartalmazzák, hogy az egyes években hány turista tervezte, hogy eljön a városba: minden sor 3 darabszámot tartalmaz (japánok, modernek, többiek). A program kérje be a fájl nevét, majd jelenítse is meg a tartalmát. (Feltehetjük, hogy a fájl formátuma helyes.) Egy lehetséges bemenet:

**Terv**

A feladat alapja a lent látható képen kivehető. Egy turista (később részletezve) egy bizonyos várost látogat meg, mely látogatás hatására a város bevétele változáson esik át és a város állapotában is változás áll be. Amennyiben ez a bevétel változás olyan mértékű, hogy „fejlesztés”-re adhat okot, úgy maga a bevétel is változással lesz az állapotra. Ezen sorozatos állapotváltozások végeredményben arra lesznek hatással, hogy mennyi turista fog érkezni. A turisták érkezésére természetesen hatással van még az is, hogy alapvetően hányan érkeztek volna (továbbiakban: Várt turista).

A turisták között megkülönböztethetjük a feladat leírása alapján a **Japán** turistákat, a **Modern** turistákat, illetve a más/harmadik típusúként megnevezett turistákat (továbbiakban: **Más**). Az **Open-Close elv** betartása céljából ezeket származtatással definiáljuk és a **DRY (Don’t Repeat Yourself) elv** szerint a szülő (vagy Parent) osztályban a közös részeket, mint pl. a getVart() metódust definiálom. 

A Turista osztály beolvasás() metódusával rendelkezik a Japán, Modern és Más osztály is. Itt az első paramétereként kapott int típusú számmal feltölti a Turista osztályból leszármaztatott (protected) vártTurista adattagot a megfelelő értékekkel (ami minden leszármaztatott osztály esetében változó), majd a második paramatéreként kapott Állapot típusú objektumból lekéri a város állapotának szöveges reprezentációját az Állapot getName() metódusával, majd a szöveges reprezentáció szerint feltöltjük a turisták érkTurista adattagját. Itt a ***„látogató” tervmintát*** alkalmazzuk, mivel a Turista osztály és leszármaztottai beolvasás() metódusának az Állapot a „látogatója”, viszont ebben a kontextusban nem változik, ugyanis külön metódusok lettek írva az állapot változtatására (mint pl. a Város látogatás() metódusába ágyazott romlás vagy az éves javítás() metódus).

A város állapotát ebből következően egy külön osztályban valósítottam meg. Az **Érkező turisták x Állapot**, illetve a **Turisták látogatása x Romlás okozása** az alábbi táblázatból jól kivehető (a feladat szövegeből kinyerhető volt), lebontva a turista adott típusára. 

A várt jelen kontextusban azt jelenti, hogy éppen annyi turista érkezett, amennyit a beolvasás során paramétereként kapott az adott és leszármaztatott objektum, tehát lényegében számbeli változtatás nem történt, változatlan maradt. Ettől függetlenül (az egyszerűség kedvéért) mindegyik végleges értéke a érkTurista adattagban kerül rögzítésre.

**Érkező turisták táblázat**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Turista típus/ Állapot*** | **lepusztult** | **átlagos** | **jó állapotú** |
| **Japán** | 0 | várt | várt \* 1.2 |
| **Modern** | várt | várt \* 1.1 | várt \*1.3 |
| **Turista** | várt | várt \* 1.1 | várt |

**Állapot romlása táblázat**

|  |  |
| --- | --- |
| ***Turista típus*** | ***Emberenkénti romlás ( 1 pont )*** |
| **Japán** | nincs romlás |
| **Modern** | 100 |
| **Turista** | 50 |

**Az állapot változására több osztályban is irányul metódus**. A Turista leszármaztatott osztályaiban az allapotRontas(), ami kiszámolja, hogy adott típusú turista hány pontot rontana az állapoton (**ld. táblázat**). Ezt a metódust segédfüggvényként használja fel a Város latogatas() metódusán belüli mellékrész, ahol ezen rontó pontokat szummáza, majd az Állapot setPont() metódusával megváltoztatja az így kialakuló állapotot.

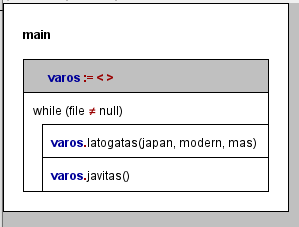
A másik főbb szerepet kapó metódus szintén a Város osztályban szerepet kapó javítás() nevet viselő metódus, aminek az a lényegi fontossága, hogy ha az 1 milliárdon felüli keretet átlépi a város bevétele, akkor 20 milliónként 1 pontot tudunk javítani a város állapotán. A romláshoz hasonlóan az Állapot setPont() metódusa végzi ezen állapot megváltoztatását.

A megoldandó feladat specifikációja az alábbi.

**Állapottér:** város: város1, turista: japán, modern, más  
**Előfeltétel:** város = város0 ^ ev > 0**Utófeltétel:** Ef ^ bevétel = japáni + moderni + mási \* 100’000, i = 1..ev ^ állapot = állapot – (moderni.allapotRontas() + mási.allapotRontas()), i = 1..ev

A turisták i-edik változatát a japán, modern, mási jelöli. Az i itt jelen esetben adott évet jelent. Az évenkénti lebontott turisták száma nem lehet több értelemszerűen, mint az évek száma, ezért lett kikötve az, hogy i = 1..ev. Mind az állapot romlása/javulása, mind a bevétel egy egy évenkénti összegzésre vezethetőek vissza.

A főprogram és a hozzá kapcsolódó lényegesebb metódusok algoritmusai a következőek.



Ahogy látható a főprogramban lényegében csak a fájlból beolvasás művelete van megvalósítva, illetve a város egy-egy metódusának meghívása, melyből a latogatas() metódus csupán inicializálja a megfelelő adattagokat, a javítás(), pedig esztétikai és átláthatósági okokat szolgál, hiszen ezt is meg lehetett volna valósítani a latogatas() metódusban.

A látogatás metódus végi a program központi részét és a következőképp épül fel.

A képen asztal látható

Automatikusan generált leírás

Mint látható a látogatás() metódus több dolgot is összefoglal. A turista osztályból leszármaztatottak beolvasas() metódusát hívja fel, melyek inicializálják várt és érkezettTurista adattagjaikat, de ezen felül a bevetélt és az állapotot is megváltoztatja. A látogatás metódus pontosan annyiszor hívódik meg, ahány év van.

Véleményem szerint ezekből már tisztán kivehető, hogy a Város elpusztulása után, a külön szervezett Állapot és a turisták sem maradnak „életben”.

Javíthatnánk a lényegi szerepet betöltő latogatas() algoritmusát azzal, ha a javítást is bele foglaltuk volna, nem pedig külön a javítás() metódusba, mivel ekkor az állapot setPont() metódusában a már javítás – rontás + alap-ot tudtuk volna átadni, amivel éppen egy hívással kevesebbet teszünk, valamint a bevételt sem kellene többször megváltoztatnunk, hanem elég lenne egyszer, de végeredményben a metódus műveletigényén nem változtattunk nagyságrendekkel.

**Tesztelési terv**

**Szürke doboz tesztesetek:**

* **Bevétel és állapot**
* Város setBevetel() és getBevetel() metódusainak tesztelése
* Állapot setAllapot() és getAllapot() metódusainak tesztelése
* Állapot setPont() metódus közvetett ellenőrzése, int típusú paraméterként kapott pontok szövegessé formálása intervallumok alapján
  1. Állapot < 34 (lepusztult)
  2. 33 < Állapot < 68 (átlagos)
  3. Állapot > 67 (jó állapotú)
* **Várt és érkezett turisták**
* Város latogatas() metódusának tesztelése az állapot függvényében
* Város getJapanErk(), getModernErk(), getMasErk() metódusainak tesztelése, ami a megnevezett objektumok érkezett turistáinak számát adja vissza (átalakított)
* Esetek:
  1. Lepuszult állapot (japánok: 0\*, modernek: 1\*, masok: 1\*)
  2. Átlagos állapot (japánok: 1\*, modernek: 1,1\*, masok: 1,1\*)
  3. Jó állapot (japánok: 1,2\*, modernek: 1,3\*, masok: 1\*)
* **Javítás és romlás**
* Város javítás() metódusának tesztelése elég nagy bevétel tekintetében
* Megváltozott állapot tesztelése, hogy valóban javult-e
* Város latogatas() metódusába ágyazott romlás tesztelése
* Megváltozott állapot tesztelése, hogy valóban romlott-e