Kiss Barnabás 2021.07.02.

barnabacska99@gmail.com

Módosított dokumentáció

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 a fájlban jelölt évek 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:

```
50
3
1000 4000 6000
2000 3000 8000
6500 5000 3000
```

Egyéb elvárások a programmal kapcsolatban

Csak azt a funkcionálisan működő (a kitűzött feladatot megoldó), felhasználó barát, bolond biztos input-output felületű alkalmazást fogadjuk el, amelyhez a megadott feltételeket kielégítő terv-dokumentáció tartozik, és amelynek kódja a tervnek megfelel, és automatikus tesztkörnyezettel rendelkezik.
A feladatok megoldásához több olyan osztályt kell használni, amelyek egy közös ősosztályból származnak és felüldefiniálják az ősosztály virtuális metódusait. Ezen osztályok objektumait egy gyűjteménybe kell elhelyezni, majd ezt a gyűjteményt kell bejárni, a benne levő objektumok megfelelő metódusait meghívni. Ez a bejárás a futásidejű polimorfizmusra támaszkodik. Használjon legalább két tervezési mintát is.
A gyűjtemények bejárását továbbra is a tanult programozási tételek alapján végezze.
A tesztkörnyezet biztosítson egységenkénti (osztály, metódus) tesztelési eseteket is a végfelhasználói teszteseteken kívül.

Elemzés

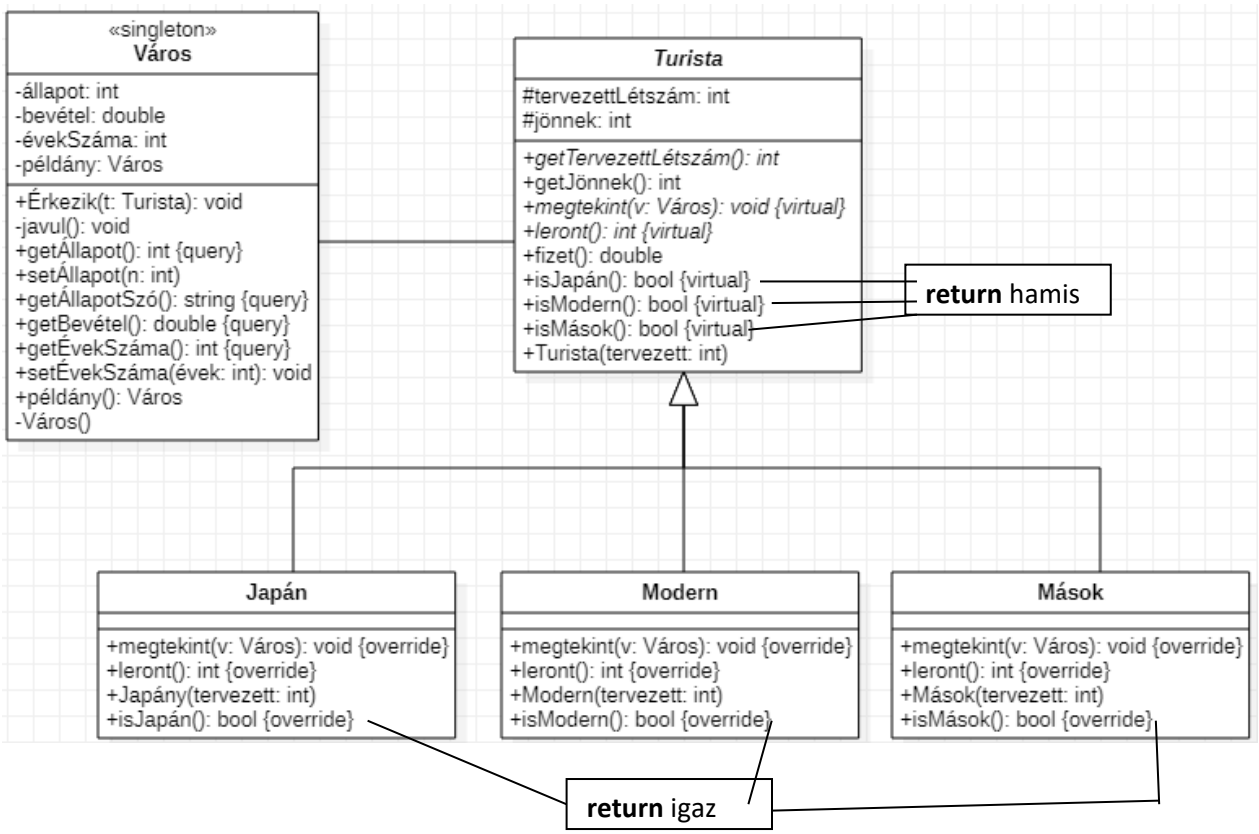
A feladat önálló objektumai egy **város**, és három féle **turista**: **japánok**, **modernek** és a **többiek**.

A város rendelkezik egy 1 és 100 pont közötti **állapotponttal** és valamennyi **bevétellel**.

A turisták három csoportja különböző befolyással van a város állapotára, és a város állapota befolyással van a turistacsoportokra:

Turistacsoport	Turistacsoport hatása a városra	**Lepusztult** állapotú város hatása a turistacsoportra (1-33 pont között)	**Átlagos** állapotú város hatása a turistacsoportra (34-67 pont között)	**Jó** állapotú város hatása a turistacsoportra (68-100 pont között)
Japánok	*Nem változtat*	Egyáltalán nem jönnek	*Nem változtat*	20%-kal többen jönnek
Modernek	100-asával rontanak egy pontot	*Nem változtat*	10%-kal többen jönnek	30%-kal többen jönnek
Mások	50-esével rontanak egy pontot	*Nem változtat*	10%-kal többen jönnek	*Nem változtat*

Terv



Mivel az általam használt szerkesztőprogram nem tette lehetővé az osztályok metódusainak, algoritmusának a vizuális megjelenítését a diagramon, ezek alább kerülnek szemléltetésre.

A Turistából származtatott Japán, Modern, Mások osztályok között a **Sablonfüggvény** tervezési minta látható, illetve a Város osztály **Singleton** tervezési mintával került megvalósításra.

Város osztály

Érkezik(&t: Turista): void
állapot := állapot - t->leront()
if(állapot < 1**), then:** állapot := 1 **endif**
bevétel := bevétel + t->fizet()

Javul(): void
if(bevétel > 1000**), then:**
    bevétel := bevétel – 1000
    állapot := állapot + (bevétel *div* 20)
endif
bevétel := 0
if(állapot > 100**), then:** állapot := 100 **endif**

getÁllapotSzó(): string
if(állapot < 34**), then: return** „rossz”
else if(állapot < 67**), then: return** „átlagos”
else: return „jó”
endif

példány(): Város
if(példány = NULL**), then:** példány := **new** Város() **endif**
return példány

Turista osztály

fizet(): double
return jönnek * 0,1

Turista(tervezett: int)
tervezettLétszám := tervezett

Japán

megtekint(v: Város): void {override}
állapot := v->getÁllapot()
if(állapot <= 33**), then:** jönnek:=0
else if(állapot >= 67**), then:** jönnek := tervezettLétszám*1,2
else: jönnek:=tervezettLétszám
endif

leront():int {override}
return 0

Modern

megtekint(v: Város): void {override}
állapot := v->getÁllapot()
if(állapot >= 67**), then:** jönnek:= tervezettLétszám * 1,3
else if(állapot >= 37**), then:** jönnek := tervezettLétszám*1,1
else: jönnek:=tervezettLétszám
endif

leront():int {override}
return [jönnek / 100]

Mások

megtekint(v: Város): void {override}
állapot := v->getÁllapot()
if(állapot >= 34 && állapot <= 67**), then:** jönnek:= tervezettLétszám * 1,1
else: jönnek:=tervezettLétszám
endif

leront():int {override}
return [jönnek / 50]

Fő:

t := <>
város := getPéldány()
generál(város, t)
szimulál(város, t)

generál(&város: Város, t: Turista*):

város->setÁllapot(n)	
város->setÉvekSzáma(n)	
t.resize(város->getÉvekSzáma())	
i=0..város->getÉvekSzáma	
	t[i*3] := new Japán(n)
	t[i*3+1] := new Modern(n)
	t[i*3+2] := new Mások(n)

szimulál(&város: Város, t: Turista*):

i=0..város->getÉvekSzáma	
	t[i*3]->megtekint(város)
	t[i*3+1]->megtekint(város)
	t[i*3+2]->megtekint(város)
	város->érkezik(t[i*3])
	város->érkezik(t[i*3+1])
	város->érkezik(t[i*3+2])

Ahol **n** egy tetszőleges egész számot jelöl.

Specifikáció

A = (város: Város, t: Turista*, file: infile(N))

Ef = (város = város', t = t', file = file', file.size()!=0)

Uf = (város.állapot' = file első száma, város.évekszáma' = file második száma, t' = file maradék számai, város.állapot = =város.állapot + $\sum_{i=0}^{|t|/3}$ (- t[3*i].leront - t[3*i+1].leront - t[3*i+2].leront + (város.javít() = (város.bevétel = t[3*i].fizet + t[3*i+1].fizet + t[3*i+2].fizet))))

város.javít() = (város.bevétel-1000)/20, ha város.bevétel > 1000

ha t[i].isJapán igaz; t[i].leront = 0, t[i].jönnek = 0, ha város.állapot < 34
t[i].tervezett, ha 33 < város.állapot < 67
t[i].tervezett*1,2, ha város.állapot > 68

ha t[i].isModern igaz; t[i].leront = t[i].jönnek/100,
t[i].jönnek = t[i].tervezett, ha város.állapot < 34
t[i].tervezett*1,1, ha 33 < város.állapot < 67
t[i].tervezett*1,3, ha város.állapot > 68

ha t[i].isMások igaz; t[i].leront = t[i].jönnek/50,
t[i].jönnek = t[i].tervezett, ha város.állapot < 34
t[i].tervezett*1,1, ha 33 < város.állapot < 67
t[i].tervezett, ha város.állapot > 68