Programozási nyelvek (Java) vizsga, 2017.05.25.

Általános tudnivalók

Ebben az ismertetésben az osztályok, valamint a minimálisan szükséges metódusok leírásai fognak szerepelni. A feladatmegoldás során fontos betartani az elnevezésekre és típusokra vonatkozó megszorításokat, illetve a szövegek formázási szabályait.

Segédfüggvények létrehozhatóak, a feladatban nem megkötött adattagok és elnevezéseik is a feladat megoldójára vannak bízva. Törekedjünk arra, hogy az osztályok belső reprezentációját a *lehető legjobban védjük*, tehát csak akkor engedjünk, és csak olyan hozzáférést, amelyre a feladat felszólít, vagy amit azt osztályt használó kódrészlet megkíván!

A beadott megoldásodnak működnie kell a mellékelt tesztprogramokkal, de ez nem elégséges feltétele az elfogadásnak. A megírt forráskód legyen kellően általános és újrafelhasználható!

Használható segédanyagok: Java dokumentáció, legfeljebb egy üres lap és toll. Ha bármilyen kérdés, észrevétel felmerül, azt a felügyelőknek kell jelezni, *NEM* a diáktársaknak!

Tesztelés

Az egyes részfeladatokhoz tartoznak külön tesztesetek, amelyeket a feladatok végén jelöltük meg. Ezek önállóan is fordítható és futtatható `.java` állományok a mellékelt `.jar`segítségével. Például Windows alatt az első feladathoz tartozó tesztesetek így fordíthatóak és futtathatóak:

```
> javac -cp .;tests-Render.jar tests/Part1.java
```

```
> java -cp .;tests-Render.jar tests/Part1
```

Ugyanezeket a teszteseteket használja a komplett feladathoz tartozó tesztelést végző `Test` osztály is. Ezt Windows alatt így lehet futtatni:

```
> java -cp .;tests-Render.jar Test
```

Linux alatt mindent ugyanúgy lehet fordítani és futtatni, csak a `-cp` paraméterében a pontosvesszőt kell kettőspontra cserélni.

A feladat összefoglaló leírása

A feladatban egy szémítógépes képelkotás egy erősen leegyszerűsített illusztrációját valósítjuk meg. Ebben a modellben a testek felületét a csúcspontok 3d-s a koordinátákival meghatározott háromszögek alkotják, melyeken különféle transzformációk hajthatóak végre. Az ily módon reprezentált teret aztán levetíthetjük egy, a képernyőt szimbolizáló két dimenziós síkra.

A programhoz tartozik egységtesztelő amely az egyes osztályok funkcionalitását teszteli.

A feladat részletes ismertetése

Part1 (11+2 pont)

render.shape.Point (3p)

Point - két dimenziós pontokat reprezentáló osztály - a pont koordinátái rejtett egészekben legyenek eltárolva - nyilvános konstruktor ami a koordinátáknak megfelelő számú egészet vár - írjunk a koordináták lekérdezésére szolgáló getX() és getY() metódust!

- nyilvános makeCopy metódus ami az objektum másolatát adja vissza
- toString: a következő formában"[x = 1, y = 2]"

- egy translate nyilvános metódusra, amely a koordináták számának megfelelő egészet vár, majd ezekkel eltolja a pont koordinátáit

render.shape.Point3D (3p)

Point3D - a Pointot terjeszti ki egy új koordináta hozzáadásával, implementáljuk eszerint a előbb megadott függvényeket! (toString: [x = 0, y = 0, z = 0]) - írjuk meg a hiányzó getZ() metódust! - ezen felül itt hozzunk létre egy project metódust amely visszaadja az adott térbeli pont (x,y) vetületét!

Comparable, equals, hashCode (4p)

Comparable - a két pont osztály valósítsa meg a Comparable interfészt! A compareTo metódus működjön a következőképpen: - tekintsük "nagyobbnak" a pontot ha x koordinátája nagyobb a másikénál. - ha egyenlő az x koordináta hasonlítsuk össze az y-okat és így tovább.. (a Point3D -be írjuk felül Pointban definiált összehasonlítást, mely így Pointot vár paraméterként, de figyeljünk rá hogy a harmadik koordinátát is helyesen kezeljük!) Equals, hashCode - írjuk meg a két osztály equals és hashSode metódusát, két pont egyenlő, ha minden koordinátája egyenlő

render.util.Color (1p)

Color - hozzuk létre a Color felsoroló típust a következő felvehető értékekkel: RED,GREEN,BLUE

Part2 (5 + 1 p)

render.shape.Triangle

- generikus osztály vagy 2 vagy 3d pontokat tárol (3db-ot), egy felület reprezentációjaként.
- A belső listában a pontok a fentebb leírt módon rendezve szerepeljenek!
- legyen egy színe - Color
- konstruktor listát kap amiről feltehetjük, hogy 3 pontot tartalmaz, azt eltárolja egy belső listában, második paraméterként egy Color objektumot. Figyeljünk a szivárogtatás mellőzésére ezen a ponton!
- ugyanakkor a implementáljuk a belső listára való referenciával visszatérő getPoints() metódust is (itt szándékosan szivárogtatunk!!)
- írjunk egy equals metódust, két Triangle akkor lesz egyenlő, ha pontjaik koordinátái megegyeznek (a színt itt nem vizsgáljuk).
- írjuk meg a Triangle hashCode-ját!
- írjunk egy toString metódust, mely a következő formában adja vissza: TR[RED[x = 1, y = 3][x = 4, y = 2][x = 4, y = 6]]

Part3 (11 + 3)

render.space.BaseSpace (4 p)

- hozzunk létre egy BaseSpace nevű absztrakt generikus osztályt, mely típusparaméterként vagy Pointot vagy Point3D-t vár.
- ebben az osztályban tároljuk a Triangle objektumainkat egy halmaz adatszerkezetben, azaz ha két háromszög koordinátái megegyeznek, csak az egyik fog bekerülni.
- írjunk egy addTriangle metódust, mely paraméterül vár egy Triangle objektumot, és hozzáadja a többihez.
- írjunk egy toString metódust, mely a következő formában adja vissza:

SPACE[TR[BLUE[x = 2, y = 3][x = 3, y = 14][x = 7, y = 1]] TR[GREEN[x = 2, y = 4][x = 5, y = 7][x = 8, y = 10]] TR[RED[x = 1, y = 3][x = 4, y = 2][x = 4, y = 6]]]

- írjunk ezen felül egy azonos típussal visszatérő nem itt implementált translate metódust, mely egy eltolást végez a tér háromszögein. A paraméterek az eltolás-vektor irányának koordinátáit reprezentáló egészek legyenek.
- írjunk egy getTriangleCount() metódust, mely visszaadja a tárolt háromszögek számát!

render.space.Plane (3 p)

- a BaseSpace absztrakt osztály Point-okkal példányosított háromszögeket tartalmazó megvalósítása.
- implementáljuk a translate metódust, mely a háromszögek pontjait tolja el a megadott koordinátáknak megfelelően. itt most egy síkról beszélünk, így a z irányú eltolást figyelmen kívül hagyhatjuk!
- írjunk egy filenevet váró writeToFile metódust, mely a toString használatával kiírja a sík szöveges reprezentációját a megadott nevű fájlba! (a fájl írása nem kerül automatikus tesztelésre, azt ellenőrizzük magunk!)

render.space.Space (4 p)

- írjunk egy paraméter nélküli konstruktort! -a BaseSpace absztrakt osztály Point3D-kel példányosított háromszögeket tartalmazó megvalósítása.
- implementáljuk a translate metódust, mely a háromszögek pontjait tolja el a megadott koordinátáknak megfelelően.
- írjunk egy readModel(String filename) osztályszintű factory metódust, mely beolvas egy fájl nevet, majd a soronként leírt háromszögekkel létrehozza a teret. Formátum:
  - egy pont: x,y,z
  - egy sor: `<pont>;<pont>;<pont>;<szín>`, ahol `<szín>` lehet `RED` vagy `GREEN` vagy `BLUE`
- feltehetjük, hogy a fájl nem tartalmaz hibás sorokat, mindazonáltal készüljünk fel az esetleges üres sorok kezelésére!

- írjunk egy project() metódust a Space osztályba, mely a tér tartalmát levetíti egy síkba. Azaz az adott Space objektumot érintetlenül hagyva létrehoz egy síkot, melyet feltölt a tér háromszögeinek 2 dimenziós (x,z) pontokat tartalmazó vetületeivel

Pontozás

A tesztelő által adott pontszám csak becslésnek tekinthető, a gyakorlatvezető levonhat pontokat, vagy adhat részpontokat.

Ponthatárok:

- 0 - 8: elégtelen (1)
- 9 - 14: elégséges (2)
- 15- 20: közepes (3)
- 21 - 27: jó (4)
- 28 - 33: jeles (5)

Jó munkát, jó játékot!