Második Java házi feladat

Beadási határidő: 2018. 03. 22. 23:59

A feladat egy egyszerű képrajzoló program elkészítése. A program egy szöveges fájlból olvas be egyszerű grafikus utasításokat, amelyek alapján elkészíti a képet, majd kiírja egy kimeneti fájlba. A kimeneti formátum SVG[1].

A feladat egyik nem funkcionális követelménye, hogy a megadott projektet kell kiegészíteni, az alábbiak szerint. A projektet nevezzétek át, hogy megfeleljen az általános házifeladat névkonvenciónak!

A továbbiakban az x tengely jobbra, az y tengely lefele mutat.

Kép megalkotása

A `hu.ppke.itk.java.homework.picture.components` csomag tartalmaz néhány egyszerű osztályt, amelyekben különböző egyszerű alkotóelemek (törtvonal, kör és téglalap) rajzolása már meg van írva, a `public void write (Writer) throws IOException` szignatúrájú metódusban. Egészítsd ki ezeket az osztályokat a megfelelő reprezentációval úgy, hogy azokat osztályon kívül ne lehessen módosítani; a kezdeti értéket konstruktor állítsa! Praktikusan legyenek `float` felbontásban tarolva; egy kétdimenziós sík elemeiként.

Ahhoz, hogy ezeket jól együtt lehessen kezelni, írj egy `Component` interface-t (a fenti metódussal), amelyiket mindegyik implementál.

Készíts el egy külön osztályt az SVG `<g>` tagéhez, `Group` néven! Ez ugyanúgy implementálja a `Component` interface-t. Komponensek (akár más csoportok) egy csoportja, a `write` metódusa sorban meghívja a részkomponensek `write` metódusát, illetve azok eredményét beágyazza egy nyitó és egy záró `<g>` tagbe. Egy másik metódusban lehessen részeket hozzáadni.

Egészítsd ki a `Component` interface-t (és leszármazottait) a következő metódusokkal:

- `public void translate(float dx, float dy)`: a megadott távolsággal eltolja a komponenst
- `public void flipHorizontal(float axis)`: a megadott (x koordinátájú függőleges) tengely mentén tükrözi a komponenst
- `public void flipVertical(float axis)`: a megadott (y koordinátájú vízszintes) tengely mentén tükrözi a komponenst

`Picture` osztály

Készíts el egy `Picture` osztályt, amely a komponenseket tartalmazza.

Egész pontosan csoportok (`Groupok`) egy listáját (vermét) tartalmazza, ahol minden operáció az utolsó csoporton történik, kivéve a `merge` utasítás (lásd lejjebb), amely az utolsó csoportot beszúrja az utolsó előtti részeként. Ha már csak egy szabad csoport van, akkor az bemeneti hibát jelent.

[1] https://en.wikipedia.org/wiki/SVG

`public void build (Reader) throws IOException, SyntaxErrorException` metódusa felépíti a képet a megadott bemeneti streamből. Ha hibás a szintaxis, akkor `SyntaxError` kivételt dob.

`public void write (Writer) throws IOException` metódusa kiírja a legutolsó csoport által leírt képet SVG formátumban (a már megírt kód segítségével).

A bemeneti fájl formátuma

A bemeneti fájl egy egyszerű szovegfájl, amelyben Unixos sorvége karakterekkel ("\n") elválasztva szerepelnek az egyes parancsok. Minden parancs egy parancsszóból és nulla vagy több paraméterből áll, szóközzel elválasztva.

A parancsok a következőek lehetnek:

- `add_line_segments <x0> <y0> <x1> <y1> [<xn> <yn>...]`: törtvonal hozzáadása; legalább 4 paramétere van, a szakaszok végpontjainak x és y koordinátái
- `add_circle <centre_x> <centre_y> <radius>`: kör hozzáadása
- `add_rectangle <corner_x> <corner_y> <width> <height>`: téglalap hozzáadása, `corner_x` es `corner_y` a bal felső sarok koordinátái
- `new_group`: új csoport hozzáadása (nem az eddigi csoport részeként, hanem a lista végére)
- `translate <dx> <dy>`: eltolás
- `flip_vertical <axis>`: vízszintes tengely menti tükrözés
- `flip_horizontal <axis>`: függőleges tengely menti tükrözés
- `merge`: az aktuálisan kijelölt csoportot (a szabad csoportok listájának végén) beszúrja az egyel előző részei közé (azok végére); ha már csak egy ilyen van, akkor az hiba

Elképzelhető, hogy a bemeneti fájl hibás, ekkor a program értelmes hibaüzenet kíséretében leáll[2]. A triviálisan javítható hibáknál (üres sor, felesleges whitespace sor végén, egyebek) nem kötelező leállni.

A kimeneti fájl formátuma

Kétféle kimeneti fájl van: az egyik eseten közvetlenül SVG formátumban, a másikban egy HTML kódba beágyazva szerepel a kép. Mindkét formátum szöveg alapú, az új sor karakter a platform alapértelmezettje legyen!

Mindkét formátumú fájlt meg tudod nyitni, például egy böngészőben.

Ügyelj rá, hogy mindkét formátumú fájl konzisztensen indentálva legyen!

SVG fájl

Az SVG kódhoz a `write` függvényekben előre megírtuk a kiiratást, neked csak az elejére és a végére kell ezt beszúrni:

```
<?xml version="1.0" encoding="UTF-8" ?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  kép kódja...
</svg>
```

[2] Nem számít értelmes hibaüzenetnek például a Java stacktrace. Viszont érdemes pl. kiírni, hogy hol történt a hiba, gondolj arra, hogy te hogyan debugolnad a saját bemeneti fájlodat.

Illetve még szükséged lesz a `<g>` tagek formátumára: a részkomponensek egy szerűen a nyitó és záró tagek között helyezkednek el.

HTML fájl

A HTML formátum esetén egy `<img>` tagben, base64[3] kódolással helyezendő el a kép.

Ehhez írj egy `java.io.Writer` utódosztályt (filtert), amely egy szűrőként működve megvalósítja a base64 kódolást. (Javasolt a `java.util.Base64` osztály tanulmányozása.) Ez egy `OutputStreamet` (karakterstreamet) csomagoljon be, amelybe kiírja a base64 karaktereket.

A komplett SVG kódot base64 kódolva a következő sablonba kell beszúrnod (a `[...]` helyére):

```
<html>
  <body>
    <img height='100%' width='100%' src="[...]" />
  </body>
</html>
```

Megjegyzés: a base64 egy olyan kódolás, amellyel bináris adatot olyan szöveggé lehet konvertálni, amelyben kevés speciális karakter van, ezért például HTML fájlokba remekül be lehet ágyazni. Mivel az SVG önmagában szöveges formátum, ezért ez a lépés itt teljesen felesleges. Ha tömörített SVG formátumot használnánk, ahhoz már kellene a base64.

A program használata

A program használatát parancssori argumentumok segítségével lehet irányítani: két kötelező argumentuma a rendre bemeneti és a kimeneti fájl elérési útvonala. Ezek előtt opcionális paraméter a `--html` kapcsoló, amellyel a kimeneti fájl formátumát lehet átállítani a HTML formátumra, a fentiek szerint.

Egyéb rendelkezések

A feladat megoldásához kövessétek az OOP elveket, készítsetek átlátható, könnyen olvasható kódot! Felhasználó okozta hibákat (érvénytelen a fájl(név)) kezeljétek le, és informáljátok a felhasználót; érvénytelen paraméterekkel ne fusson tovább a program!

Nem kötelező használni (nyilván anélkül is meg lehet oldani a feladatot), de érdemes átnézni, hogy a `java.util.ArrayList` hogy működik. Pl. az alábbi kód keszit egy listát számokból:

```
ArrayList<Integer> list = new ArrayList<>(); // int-tel nem fog működni
// ArrayList<PrintWriter> otherList;         // de osztályokkal simán
list.add(2);
list.add(3);
list.add(0, 1);
list.remove(1);
System.out.println(list.get(1));  // 3
```

Vájt fülűek számára a feladat megoldásához a fentiek alapján a Composite, a Memento és az Adapter tervezési mintákat kell használni.

[3] https://en.wikipedia.org/wiki/Base64

Extra feladat (ráadás pontokért)

Implementálj visszavonás mechanizmust a Picture osztályban!

Az osztály tartalmazzon egy undo metódust, amelyik az utolsó műveletet képes visszavonni. (Majd utána a többit, visszafele haladva.) Ehhez az eddig elvégzett műveleteket tárolni kell egy listában (a `Picture` osztályon belül), pontosabban ezek inverzeit (még pontosabban a műveletek előtti állapotot). Ezeket a `PictureState` interface-t implementáló osztályok reprezentálják, amelyek `public void undo()` metódusa valósítja meg a tényleges visszavonást.

Vigyázzatok rá, hogy a visszavonás mechanizmus ne törje meg az enkapszulációt!

A bemeneti parancsok közé végy fel egy új parancsot:

- undo: visszavonja az utolsó (meg vissza nem vont) műveletet; ha nincs ilyen, az hiba

Jó munkát!