Programozási technológia 1.

beadandó feladat, dokumentáció

Szabó Norbert

YECVA4

Eötvös Loránd Tudományegyetem

Informatika Kar

2015/ősz

Gyakorlatvezető: Nagy Sára

A feladat leírása

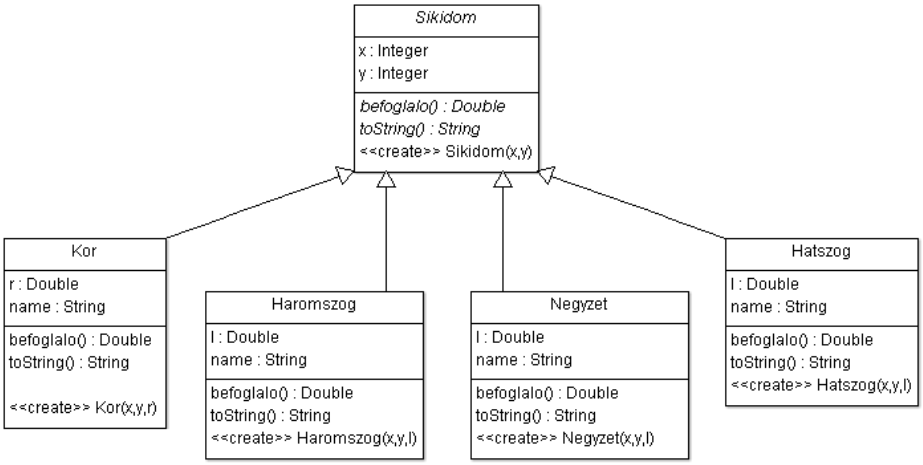
Töltsön fel egy gyűjteményt különféle szabályos (kör, szabályos háromszög, négyzet, szabályos hatszög) síkidomokkal! **Adja meg melyik síkidom befoglaló téglalapja a legnagyobb területű!** Egy síkidom befoglaló téglalapja lefedi a síkidomot, oldalai párhuzamosak a tengelyekkel. Minden síkidom reprezentálható a középpontjával és az oldalhosszal, illetve a sugárral, ha feltesszük, hogy a sokszögek esetében az egyik oldal párhuzamos a koordináta rendszer vízszintes tengelyével, és a többi csúcs ezen oldalra fektetett egyenes felett helyezkedik el. A síkidomokat szövegfájlból töltse be! A fájl első sorában szerepeljen a síkidomok száma, majd az egyes síkidomok. Az első jel azonosítja a síkidom fajtáját, amit követnek a középpont koordinátái és a szükséges hosszúság. A feladatokban a beolvasáson kívül a síkidomokat egységesen kezelje, ennek érdekében a síkidomokat leíró osztályokat egy közös ősosztályból származtassa!

A megvalósításról általánosságban

Mivel minden síkidom rendelkezik két közös tulajdonsággal (x és y koordináta), így létrehozzuk a `Sikidom` (síkidom) absztrakt osztályt, és abból származtatunk minden síkidomot. Az absztrakt osztály specializációja lehetne egyszerűen például a `SzabalyosSokszog` osztály is, ahol megadjuk az oldalak számát és egy oldal hosszát; és ha az oldalak száma 0, a síkidom körnek számít és az oldal hosszának megadott adat a sugár. A feladat mégis azt kérte, hogy csak négyféle síkidomot hozhassunk létre: kör, háromszög, négyzet és hatszög, így inkább mindegyiket saját osztállyal (`Kor, Haromszog, Negyzet, Hatszog`) valósítottam meg. Az adattagok (az absztrakt `Sikidom` osztályban is) `public` -ra lettek állítva, bár a feladat megoldásának szempontjából nincs jelentősége. Emellett megkapták a `final` jelzőt is, ennek bírálata megközelítés kérdése. A feladat megoldása közben úgy gondolkodtam, hogy ha rajzolok a (papíros) koordinátarendszerre egy síkidomot, azt nem tudom csak úgy módosítani: ki kell radíroznom az egészet és újrarajzolni. Ennek függvényében értelmezhető a `final` jelző. Az UML diagramon talán zavaró lehet, hogy minden speciális osztálynál fel van tüntetve a két függvény, holott azok az absztrakt osztályban is szerepelnek. Ez azért van, mert felül kell írja az absztrakt

osztály metódusait, hiszen minden síkidomnak egyedi `toString()` és `befoglalo()` eljárása van.

UML diagram



Sikidom (síkidom) absztrakt osztály leírása

Két `final integer` -t tárolunk: X és Y koordináta. Absztrakt függvényként definiáljuk továbbá a `toString()` és a `befoglalo()` függvényeket. Az előbbi visszaad egy `String` objektumot, amiben a síkidom adatai kerülnek megjelenítésre kényelmesen olvasható formában (típus, koordináták és befoglaló téglalap területe). Az utóbbi pedig a befoglaló téglalap területét adja vissza `double` típusú adatként. Mindkét függvény absztrakt, tehát önmagában nem értelmezhető; szükségünk van a speciális osztályra, ami kiegészíti a `Sikidom` osztályt. A speciális osztályban definiált konstruktor után nyerhet értelmet a `befoglalo()` függvény, ami nélkül pedig nem értelmezhető a `toString()` függvény sem.

Ezenkívül persze az osztálynak létezik még a konstruktora, ami egyszerűen csak értéket ad az X és Y koordinátáknak.

Kor (kör) osztály leírása

Megvalósított adattagok:

- `public final String name`
  Előre definiált: "Kör".
- `public final double r`
  A sugár hossza, amit a konstruktor szerint definiálunk.

Megvalósított függvények:

- `public Kor(int x, int y, double r)`
  A konstruktor, mellyel létrehozzuk az objektumot.

- `public double befoglalo()`
  Felülírja az absztrakt osztály `befoglalo()` függvényét; visszaadja a síkidom befoglaló téglalapjának területét.

- `public String toString()`
  Felülírja az absztrakt osztály `toString()` függvényét, visszaadja a síkidom számunka értékes adataid szöveges formában

Haromszog (háromszög) osztály leírása

Megvalósított adattagok:

- `public final String name`
  Előre definiált: "Háromszög".
- `public final double l`
  Egy oldal hossza (length), amit a konstruktor szerint definiálunk.

Megvalósított függvények:

- `public Haromszog(int x, int y, double l)`
  A konstruktor, mellyel létrehozzuk az objektumot.

- `public double befoglalo()`
  Felülírja az absztrakt osztály `befoglalo()` függvényét; visszaadja a síkidom befoglaló téglalapjának területét.

- `public String toString()`
  Felülírja az absztrakt osztály `toString()` függvényét, visszaadja a síkidom számunka értékes adataid szöveges formában.

Negyzet (négyzet) osztály leírása

Megvalósított adattagok:

- `public final String name`
  Előre definiált: "Négyzet".
- `public final double l`
  Egy oldal hossza (length), amit a konstruktor szerint definiálunk.

Megvalósított függvények:

- `public Negyzet(int x, int y, double l)`
  A konstruktor, mellyel létrehozzuk az objektumot.

- `public double befoglalo()`
  Felülírja az absztrakt osztály `befoglalo()` függvényét; visszaadja a síkidom befoglaló téglalapjának területét.

- `public String toString()`
  Felülírja az absztrakt osztály `toString()` függvényét, visszaadja a síkidom számunka értékes adataid szöveges formában.

Hatszog (hatszög) osztály leírása

Megvalósított adattagok:

- `public final String name`
  Előre definiált: "Hatszög".
- `public final double l`
  Egy oldal hossza (length), amit a konstruktor szerint definiálunk.

Megvalósított függvények:

- `public Hatszog(int x, int y, double l)`
  A konstruktor, mellyel létrehozzuk az objektumot.

- `public double befoglalo()`
  Felülírja az absztrakt osztály `befoglalo()` függvényét; visszaadja a síkidom befoglaló téglalapjának területét.

- `public String toString()`
  Felülírja az absztrakt osztály `toString()` függvényét, visszaadja a síkidom számunka értékes adataid szöveges formában.

Tesztelési terv (eredmények)

A becsomagolt NetBeans projektben elérhető néhány előre elkészített teszteset. A főprogram először a beolvasandó fájl nevét kéri, erre három példát is láthatunk a becsomagolt (beadott) mappaszerkezeten belül: 1.txt, 2.txt, 3.txt.

Mivel a feladatleírás lényegében specifikálta az input fájl kritériumait, validálásra nem is lett volna igazán szükség.

(Megjegyzendő, hogy mivel síkidomok tárolására a standard library-ben elérhető `LinkedList`-et használtam, teljesen feleslegessé vált számomra a feladatleírás szerint definiált első sor, ahol a felsorolt síkidomok számát jelenítjük meg. Ezért ezt ugyan beolvasom [hogy a fájlon végigfutó `BufferedReader` iterátora végre a második sorba érkezzen], de az adatot semmire nem használom. A feldolgozás egyszerűen a fájl végig tart.)

Ugyan nincs benne kivédve minden hibalehetőség (nem is ez volt a cél), viszont mégis sokféle problémát kezel a program.

Egyszerűen csak nézzük meg a csatolt txt-k működését (+ azt az esetet, amikor nincs meg a fájl).

1.txt (vannak benne rendes adatok, de tartalmaz két elrontott sort)

```
File neve?
1.txt
----------
File feldolgozása...
- 5. sorban ismeretlen síkidom.
- 6. sor hibás.
----------
Melyik síkidom befoglaló téglalapja a legnagyobb területű?
- Háromszög (1,1) befoglaló téglalapjának területe: 7.794228634059948
- Kör (1,1) befoglaló téglalapjának területe: 36.0
- Négyzet (1,1) befoglaló téglalapjának területe: 9.0
- Hatszög (1,1) befoglaló téglalapjának területe: 173.20508075688775
----------
A győztes a(z) 4. síkidom:
Hatszög (1,1) befoglaló téglalapjának területe: 173.20508075688775
BUILD SUCCESSFUL (total time: 8 seconds)
```

2.txt (csak elrontott sorok)

```
File neve?
2.txt
----------
File feldolgozása...
- 1. sorban ismeretlen síkidom.
- 2. sorban ismeretlen síkidom.
- 3. sor hibás.
Nincs felvitt síkidom => nincs mit kiszámolni => a program leáll.
BUILD SUCCESSFUL (total time: 2 seconds)
```

3.txt (üres fájl)

```
File neve?
3.txt
----------
File feldolgozása...
Nincs felvitt síkidom => nincs mit kiszámolni => a program leáll.
BUILD SUCCESSFUL (total time: 2 seconds)
```

Nem található fájl

```
File neve?
awddawdawd
----------
Nincs ilyen file.
Nincs felvitt síkidom => nincs mit kiszámolni => a program leáll.
BUILD SUCCESSFUL (total time: 4 seconds)
```