**Feladatleírás**

Töltsön fel egy gyűjteményt különféle szabályos (kör, szabályos háromszög, négyzet, szabályos hatszög) síkidomokkal!

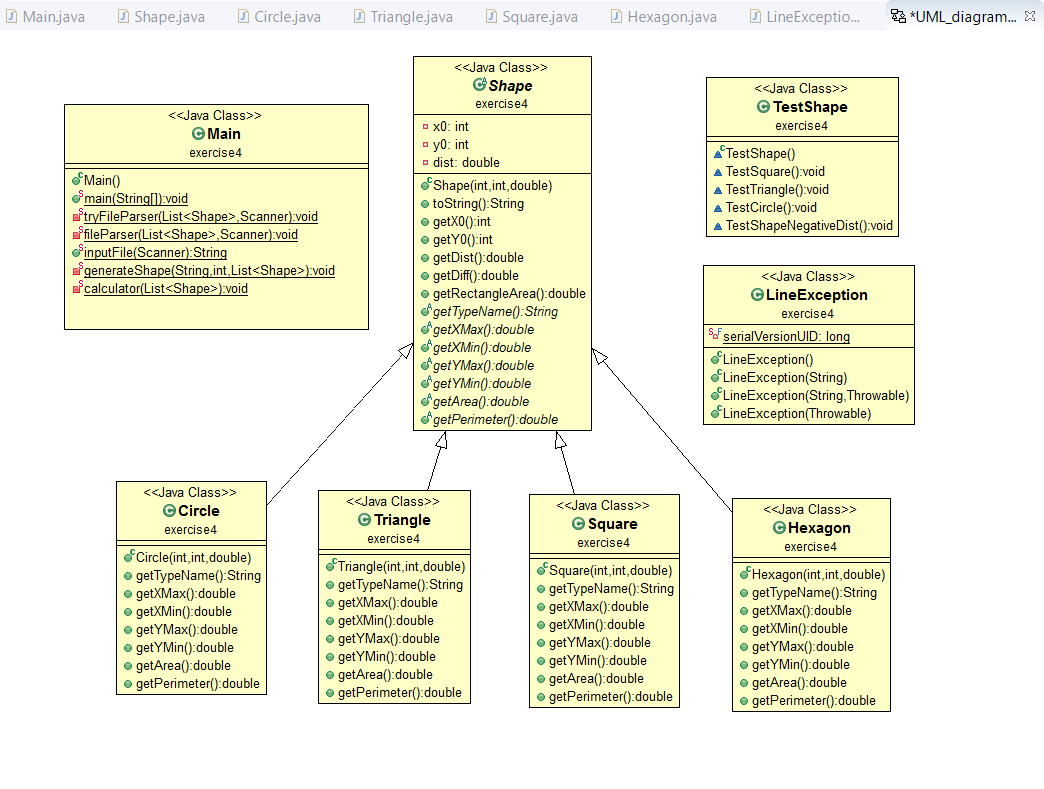
Minden síkidom reprezentálható a középpontjával és az oldalhosszal, illetve kör esetében a sugárral, ha feltesszük, hogy a sokszögek esetében az egyik oldal párhuzamos a koordináta rendszer vízszintes tengelyével, és a többi csúcs ezen oldalra fektetett egyenes felett helyezkedik el.

***5. feladat:*** *Határozza meg a legkisebb téglalapot, amely lefedi az összes síkidomot és oldalai párhuzamosak a tengelyekkel!*

***6. feladat:*** *Adja meg azt a síkidomot, amelynek a területe és a kerülete a legkisebb mértékben tér el egymástól!*

***7. feladat:*** *Egy síkidom befoglaló téglalapja lefedi a síkidomot, oldalai párhuzamosak a tengelyekkel. Adja meg melyik síkidom befoglaló téglalapja a legnagyobb területű!*

**Osztályszerkezet**

****

* A program belépési pontját tartalmazó Main osztály a bemeneti fájlnak megfelelően példányosítja a Shape osztály leszármazottjait, illetve a származtatott osztályok függvényeit használva kiszámítja és kiírja az eredményt, illetve hiba esetén a hiba okát
* A LineException osztályt egy tipikusan az én programomban előforduló kivétel kezelésére hoztam létre, amikor a bemeneti fájl valamelyik sora formai hibát tartalmaz
* A TestShape osztály egy JUnit teszteket tartalmazó osztály, mely a származtatott osztályok függvényeit teszteli

**Megvalósítás**

* A feladatot egy Java8 nyelven írt, objektumorientált szemléletet követő programmal oldottuk meg
* A bemeneti fájl első sorában szerepel a síkidomok száma, majd az egyes síkidomok
* Az első jel azonosítja a síkidom fajtáját, amit követnek a középpont koordinátái és a szükséges hosszúság

0 - kör, 3 - háromszög, 4 - négyzet, 6 - hatszög

* A Main osztály hívja meg a fileParser metódusát, ami az alakzatok fájlból való beolvasását végzi. Az alábbi kivételeket dobhatja:
  + FileNotFoundException – helytelen fájlnév
  + NumberFormatException – ha az oldalak száma nem értelmezhető számként
  + NoSuchElementException – ha az inputfájl a megadottnál rövidebb
  + LineException – ha az egyik sor hibás formátumú
* A sor formátumát egy regex kifejezéssel vizsgáljuk, ha az egyik sorra nincs match, az a sor indexével jelzett hibát dob

?<type>[0,3,4,6])\\s+(?<x0>-?\\d+)\\s+(?<y0>-?\\d+)\\s+(?<dist>\\d+

* Az eldobott kivételeket a tryFileParser metódusig dobjuk vissza, ahol mindegyiket lekezeljük és kiírjuk a megfelelő hibaüzenetet
* A tryFileParser rekurzívan meghívja önmagát a hibaüzenet kiírása után így a felhasználó újra beírhatja a fájlnevet
* A programban szereplő entitásokat a Shape osztály fogja össze: ez tartalmazza a közös mezőket, azaz az x0, y0 és dist Integer-eket és a közös metódusokat.
* Rendelkezik számos absztrakt függvénnyel amelyek törzse az egyes leszármazott osztályokban különbözőképpen van megírva az adott alakzat geometriájának megfelelően
* A számolást a Main osztály calculator metódusa végzi a Shape osztály leszármazottjainak függvényeit hívja meg, ezután consolra kiírja a kapott eredményeket
* Az 5., 6. és 7. feladat számolását párhuzamosan végzi, majd utolsó lépésként írja ki az összes eredményt

**Tesztelés**

A programhoz az alábbi érvénytelen teszteseteket készítettük:

* Nem létező fájl [rossz0.txt] – kivételt dob (FileNotFoundException)
* Nem szám formátumú darabszámot tartalmazó [rossz1.txt] – kivételt dob

(NumberFormatException)

* Negatív oldalhosszat tartalmazó fájl [rossz2.txt] – kivételt dob

(LineException)

* Nem elég sok elemből álló sort tartalmazó fájl [rossz3.txt] – kivételt dob

(NoSuchElementException)

* Több darabszámot tartalmazó [rossz4.txt] – kivételt dob

(NumberFormatException)

Érvényes tesztesetek:

* A 3 soros megadott standard bemenet [jo1.txt]
* 4 soros bemenet az összes dist érték 0 [jo2.txt]
* 0 területeket és kerületeket számol
* 6 soros standard bemenet [jo3.txt]
* 4 soros megadott standard bemenet, az előzőeknél sokkal nagyobb bemeneti értékekkel [jo4.txt]