**Programozási Technológia**

**I. beadandó feladat**

**Perczel-Szabó Dániel**

**GQF4SF**

# **1. Feladat**

Rögzítsen a síkon egy pontot, és töltsön fel egy gyűjteményt különféle szabályos (kör, szabályos háromszög, négyzet, szabályos hatszög) síkidomokkal! Határozza meg a legkisebb téglalapot, amely lefedi az összes síkidomot és 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!

# **2. Terv**

## **2.1. Típusok**

### **2.1.1. PlaneFigure**

A különböző síkiidomok származtatására a PlaneFigure absztrakt ősosztályt használjuk. Ez három adattagot tartalmaz, a síkidom típusát (char type), a középpontot reprezentáló (kételemű) tömböt (double centerPoint[]) és a sugár/oldal hosszát (double lengthOfR). Továbbá itt kerültek létrehozásra az absztrakt leftmostPoint(), rightmostPoint(), topPoint() és bottomPoint() függvények.

### **2.1.2. Síkidomok**

A síkidomokat (Circle, Triangle, Square, Hexagon) a PlaneFigures osztályból származtatjuk. Az ősosztály absztrakt metódusait felülírjuk úgy, hogy minden síkidom esetén a megfelelő (legbaloldalibb, legjobboldalibb, legfelső, legalsó) pontokkal térjenek vissza.

### **2.1.3. Database**

A síkidomok tárolására a Database osztályban létrehozott listát használjuk. Ebben az osztályban definiáljuk a read() metódust, amely a fájlbeolvasásért felel. Itt található továbbá a maxPoint(ArrayList<double[]>,int) és a minPoint(ArrayList<double[]>,int) függvény, melyek a paraméterben megadott listában megkeresik a legnagyobb illetve legkisebb elemet a második paraméter alapján. Végül itt definiáljuk a smallestRectangle() függvényt, amely megválaszolja a feladat kérdését, először listákba rendezi az egyes síkidomok legszélső pontjait, majd ezek közül kiválasztja a legkisebbeket, illetve legnagyobbakat. Ezen pontok koordinátáinak felhasználásával pedig megkapjuk a legkisebb, minden síkidomot magába foglaló téglalap sarkainak koordinátáit két tizedesjegyre kerekítve.

## **2.2. Osztálydiagram**

# **3. Tesztesetek**

|  |  |  |
| --- | --- | --- |
| **Leírás** | **Bemenet (fájl)** | **Elvárt kimenet** |
| Nem létező fájl | noSuchFile.txt | File not found! |
| Üres fájl | emptyFile.txt | NoSuchElementException |
| Ismeretlen síkidom a fájlban | unknownFigure.txt | Invalid input! |
| Egy elemű fájl | oneFigure.txt | Top right: (3.0;3.0), bottom right: (3.0;-3.0), bottom left: (-3.0;-3.0), top left: (-3.0;3.0) |
| Több elemű fájl | data.txt | Top right: (7.5;6.89), bottom right: (7.5;-5.46), bottom left: (-5.0;-5.46), top left: (-5.0;6.89) |
| Több elemű fájl #2 | data2.txt | Top right: (7.8;8.07), bottom right: (7.8;-5.87), bottom left: (-4.7;-5.87), top left: (-4.7;8.07) |
| Nagy elemszámú fájl | data3.txt | Top right: (8.5;8.43), bottom right: (8.5;-5.87), bottom left: (-4.7;-5.87), top left: (-4.7;8.43) |