Programowanie Obiektowe

Laboratorium 5 – Lista A (25 marca 2022)

mgr inż. Paweł Majewski

Zadanie 5.0(6.0 pkt):

Opracuj metodę obliczania współrzędnych środka ciężkości dla dowolnej maski binarnej, zapisanej w pliku *.txt*. Implementacja rozwiązania powinna uwzględniać następujące etapy:

- wczytanie danych z pliku *.txt* do odpowiedniej struktury danych (nazwa pliku powinna być wpisywana przez użytkownika),
- obliczanie współrzędnych środka ciężkości dla maski binarnej,
- zapis wyników do pliku **.txt* (punkt ciężkości powinien zostać oznaczony w zapisanym pliku jako litera P).

Uwzględnij odporność programu na wprowadzenie przez użytkownika złej nazwy pliku. Sprawdź działanie programu dla trzech przykładowych masek binarnych zapisanych w plikach tesktowych: *binary_mask_[id].txt*. Zapisz wyniki w plikach tekstowych: *binary_mask_[id]_out.txt*. Załóż, że nie znasz na początku wymiarów maski binarnej zapisanej w pliku (użyj struktury z możliwością zmiany rozmiaru np. ArrayList). Kolumny w pliku są rozdzielone spacją a kolejne wiersze znajdują się w nowych liniach pliku.

Możesz przyjąć następującą strukturę klasy:

```
1  public class FeaturesCalculator {
2
3    private ArrayList<ArrayList<Integer>> binary_mask;
4    private Integer[] mass_center;
5    private String filename;
6
7    public FeaturesCalculator(String filename) {
8    }
9
10   public void read_data() {
11   }
12
13   public void calculate_mass_center() {
14   }
15
16   public void save_results() {
17   }
18
19   public static void main(String[] args) {
20   }
21 }
```

Wskazówki odnośnie obliczania środka ciężkości dla maski binarnej. Załóżmy, że wszystkie niezerowe elementy maski binarnej należą do zbioru $P \in (P_0(x_0, y_0), P_1(x_1, y_1), ..., P_n(x_n, y_n))$. Środek ciężkości dla takiej maski binarnej możemy obliczyć ze wzoru:

$$(x_c, y_c) = (\frac{\sum_{i=0}^{n} x_i}{n}, \frac{\sum_{i=0}^{n} y_i}{n}) \tag{1}$$

Zadanie 5.1(4.0 pkt):

Opracuj testy jednostkowe dla metody **calculate_roots** z klasy **RootsCalculator**. Metoda **calculate_roots** ma zwracać wszystkie różne pierwiastki rzeczywiste obliczone dla równania o ogólnej postaci:

$$y = ax^2 + bx + c \tag{2}$$

Jako parametry metody **calculate_roots** przyjmij a, b, c, które są dowolnymi liczbami rzeczywistymi. W implementacji metody oraz testów jednostkowych należy uwzględnić nastepujące przypadki:

lp.	a	b	c	Δ	liczba pierwiastków	przykład
1	$\neq 0$	$\in R$	$\in R$	> 0	2	(1, -5, -6)
2	$\neq 0$	$\in R$	$\in R$	$= 0$	1	(1, -4, 4)
3	$\neq 0$	$\in R$	$\in R$	< 0	0	(1, 1, 6)
4	$= 0$	$\neq 0$	$\in R$	$-$	1	(0, 1, 6)
5	$= 0$	$= 0$	$\neq 0$	$-$	0	(0, 0, 7)
6	$= 0$	$= 0$	$= 0$	$-$	∞	(0, 0, 0)

Tabela 1: Przypadki do uwzględnienia przy implementacji.

Przyjmij następującą strukturę klasy:

```
22 public class RootsCalculator {
23    public static List<Double> calculate_roots(double a, double b, double c) {
24    //code
25    }
26 }
```

W przypadku braku rozwiązań zwróć pustą listę. W przypadku nieskończonej liczby rozwiązań zwróć listę z nieskończonością jako jej jedyny element. Zastosuj:

```
27 double inf = Double.POSITIVE_INFINITY;
```