

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №3-1
по курсу: «Языки и методы программирования»
«Полиморфизм на основе интерфейсов в языке Java»

Выполнил:
Студент группы ИУ9-21Б

Проверил:
Посевин Д. П.

Москва, 2024

1. Цель

Приобретение навыков реализации интерфейсов для обеспечения возможности полиморфной обработки объектов класса.

2. Персональный вариант

Класс ломаных линий на плоскости с порядком на основе количества пересечений ломаной линии с осями координат.

3. Решение

3.1. Код

```
import java.util.*;  
  
public class Test {  
    static BrokenLine genNewBrokenLine(Random rand) {  
        BrokenLine bl = new BrokenLine();  
        int n = rand.nextInt(1, 10);  
        for (int i = 0; i < n; i++) {  
            int x = rand.nextInt(-100, 100), y =  
                rand.nextInt(-100, 100);  
            bl.addPoint(x, y);  
        }  
        return bl;  
    }  
  
    public static void main(String[] args) {  
        Random rand = new Random();  
        ArrayList<BrokenLine> a = new ArrayList<BrokenLine>();  
        int m = rand.nextInt(1, 10);  
        for (int i = 0; i < m; i++)  
            a.add(genNewBrokenLine(rand));  
  
        Collections.sort(a);  
  
        for (BrokenLine bl : a)  
            bl.print();  
    }  
}
```

Код 1: Test.java

```

import java.util.ArrayList;

public class BrokenLine implements Comparable<BrokenLine> {
    private ArrayList<Point> points;
    private int intersections;

    public BrokenLine() {
        this.points = new ArrayList<Point>();
        this.intersections = -1;
    }

    public void addPoint(int x, int y) {
        points.add(new Point(x, y));
    }

    public ArrayList<Point> getPoints() {
        return this.points;
    }

    public void print() {
        System.out.println("Points:");
        for (Point p : this.points)
            System.out.printf(" %d %d\n", p.x, p.y);
        System.out.printf("Total intersections: %d\n",
                         this.getIntersections());
    }

    void calculateIntersections() {
        this.intersections = 0;

        if (this.points.size() == 0) {
            Point p = this.points.get(0);
            if (p.x == 0 || p.y == 0)
                this.intersections = 1;
        }

        for (int i = 0; i < this.points.size()-1; i++) {
            Point p1 = this.points.get(i), p2 =
                this.points.get(i+1);
            if (p1.x * p2.x <= 0) this.intersections++;
            if (p1.y * p2.y <= 0) this.intersections++;
            if (p1.x * p2.y == p1.y * p2.x)
                this.intersections--;
        }
    }
}

```

```
        if (p1.x == 0 && p2.x == 0) this.intersections =
            → Integer.MAX_VALUE;
        if (p1.y == 0 && p2.y == 0) this.intersections =
            → Integer.MAX_VALUE;
    }
}

int getIntersections() {
    if (this.intersections == -1)
        this.calculateIntersections();
    return this.intersections;
}

public int compareTo(BrokenLine other) {
    return this.getIntersections() -
        → other.getIntersections();
}
}
```

Код 2: BrokenLine.java

```
public class Point {
    public int x, y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Код 3: Point.java

3.2. Скриншоты

```
[thecakeisfalse@desktop-sc src]$ java Test
Points:
 2 67
 43 72
Total intersections: 0
Points:
 -92 -78
 61 -30
 -22 -78
 42 -49
 -62 -74
 -21 76
 77 -40
 26 37
Total intersections: 8
[thecakeisfalse@desktop-sc src]$
```

Рис. 1: Пример работы

The screenshot shows the IntelliJ IDEA interface with two tabs open: `Test.java` and `Point.java`. The `Test.java` tab is active.

```

    import java.util.*;
    public class Test {
        static BrokenLine genNewBrokenLine(Random rand) {
            BrokenLine bl = new BrokenLine();
            int n = rand.nextInt(origin: 1, bound: 10);
            for (int i = 0; i < n; i++) {
                int x = rand.nextInt(origin: -100, bound: 100), y = rand.nextInt(origin: -100, bound: 100);
                bl.addPoint(x, y);
            }
            return bl;
        }
        public static void main(String[] args) {
            Random rand = new Random();
            ArrayList<BrokenLine> a = new ArrayList<BrokenLine>();
            int m = rand.nextInt(origin: 1, bound: 10);
            for (int i = 0; i < m; i++) {
                a.add(genNewBrokenLine(rand));
            }
            Collections.sort(a);
            for (BrokenLine bl : a)
                bl.print();
        }
    }
  
```

The `Point.java` tab is also visible, showing the definition of the `Point` class:

```

    public class Point {
        public int x, y;
        public Point(int x, int y) {
            this.x = x;
            this.y = y;
        }
    }
  
```

Рис. 2: Исходный код ч. 1

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "lab2-1". The "src" folder contains three packages: "BrokenLine", "Point", and "Test".
- Code Editor:** The active file is `BrokenLine.java`. The code implements a class `BrokenLine` that stores points and calculates intersections between them.

```
11
12     1 usage
13     public void addPoint(int x, int y) {
14         points.add(new Point(x, y));
15     }
16
17     no usages
18     public ArrayList<Point> getPoints() { return this.points; }
19
20
21     1 usage
22     public void print() {
23         System.out.println("Points:");
24         for (Point p : this.points)
25             System.out.printf(" %d %d\n", p.x, p.y);
26         System.out.printf("Total intersections: %d\n", this.getIntersections());
27     }
28
29
30     1 usage
31     void calculateIntersections() {
32         this.intersections = 0;
33
34         if (this.points.size() == 0) {
35             Point p = this.points.get(0);
36             if (p.x == 0 || p.y == 0)
37                 this.intersections = 1;
38         }
39
40         for (int i = 0; i < this.points.size()-1; i++) {
41             Point p1 = this.points.get(i), p2 = this.points.get(i+1);
42             if (p1.x * p2.x <= 0) this.intersections++;
43             if (p1.y * p2.y <= 0) this.intersections++;
44             if (p1.x * p2.y == p1.y * p2.x) this.intersections--;
45             if (p1.x == 0 && p2.x == 0) this.intersections = Integer.MAX_VALUE;
46             if (p1.y == 0 && p2.y == 0) this.intersections = Integer.MAX_VALUE;
47         }
48
49         3 usages
50         int getIntersections() {
51             if (this.intersections == -1)
52                 this.calculateIntersections();
53             return this.intersections;
54         }
55     }
56
57     @Override
58     public int compareTo(BrokenLine other) {
59         return this.getIntersections() - other.getIntersections();
60     }
61 }
```

Рис. 3: Исходный код ч. 2