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

```java
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

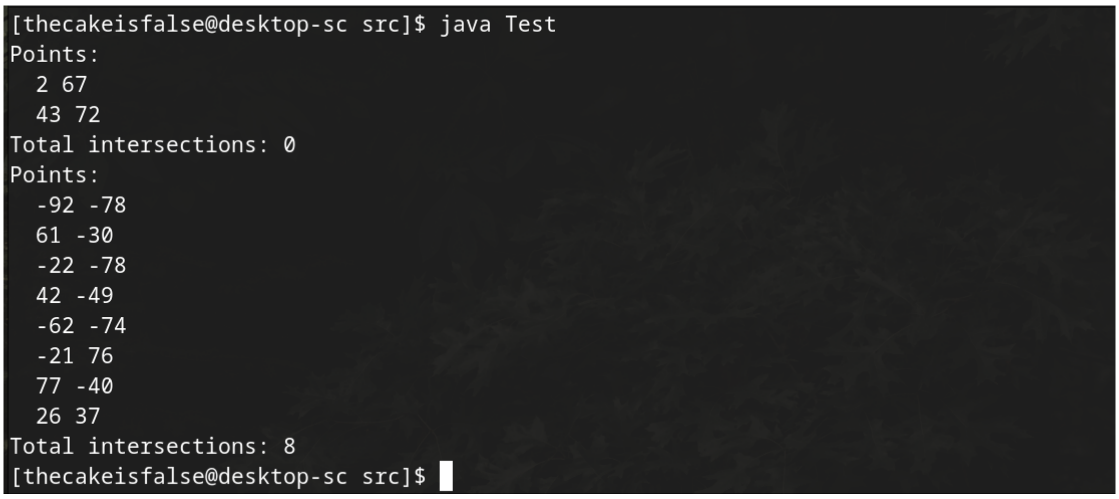
```java
import java.util.ArrayList;

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

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

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

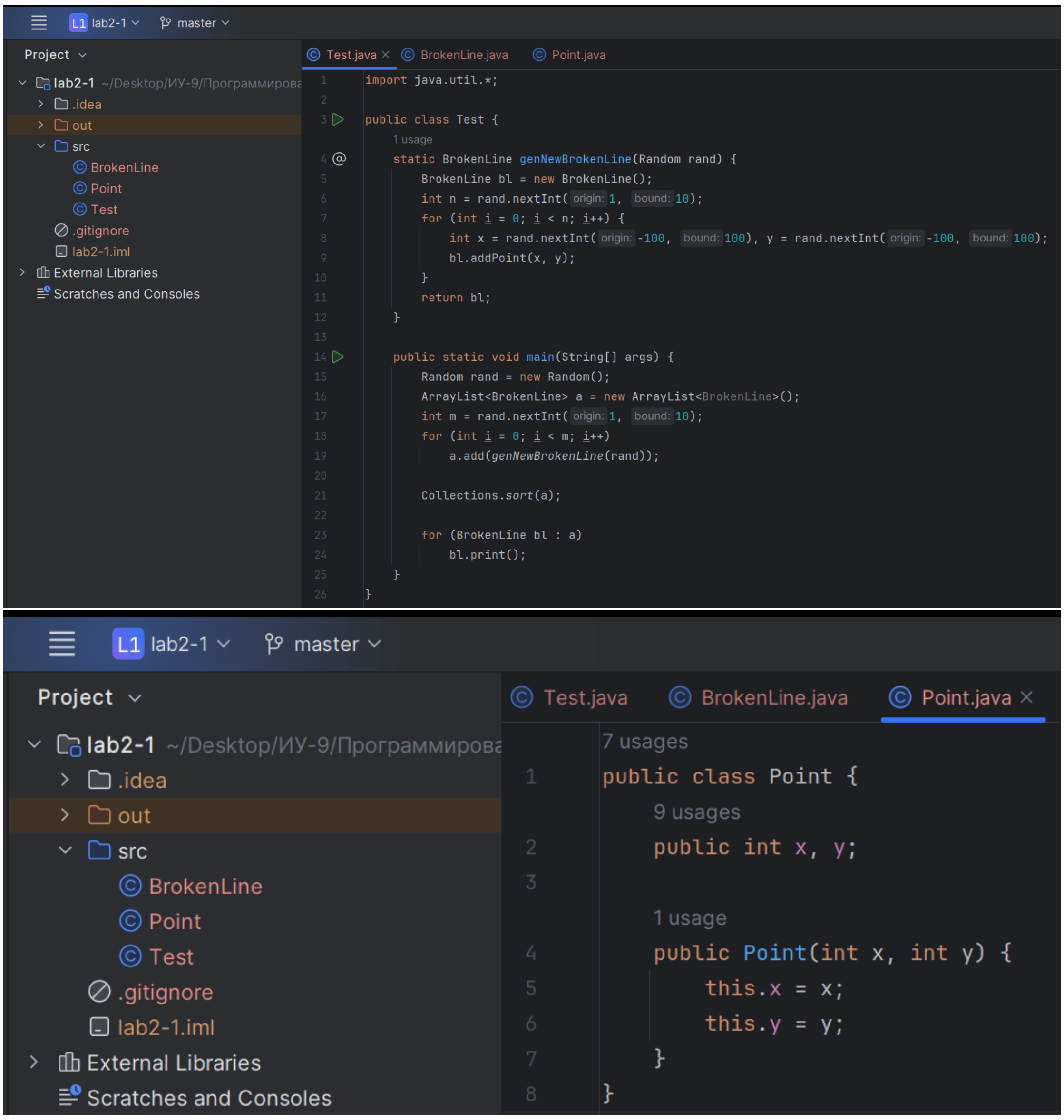
Код 3: Point.java

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]$ 
```

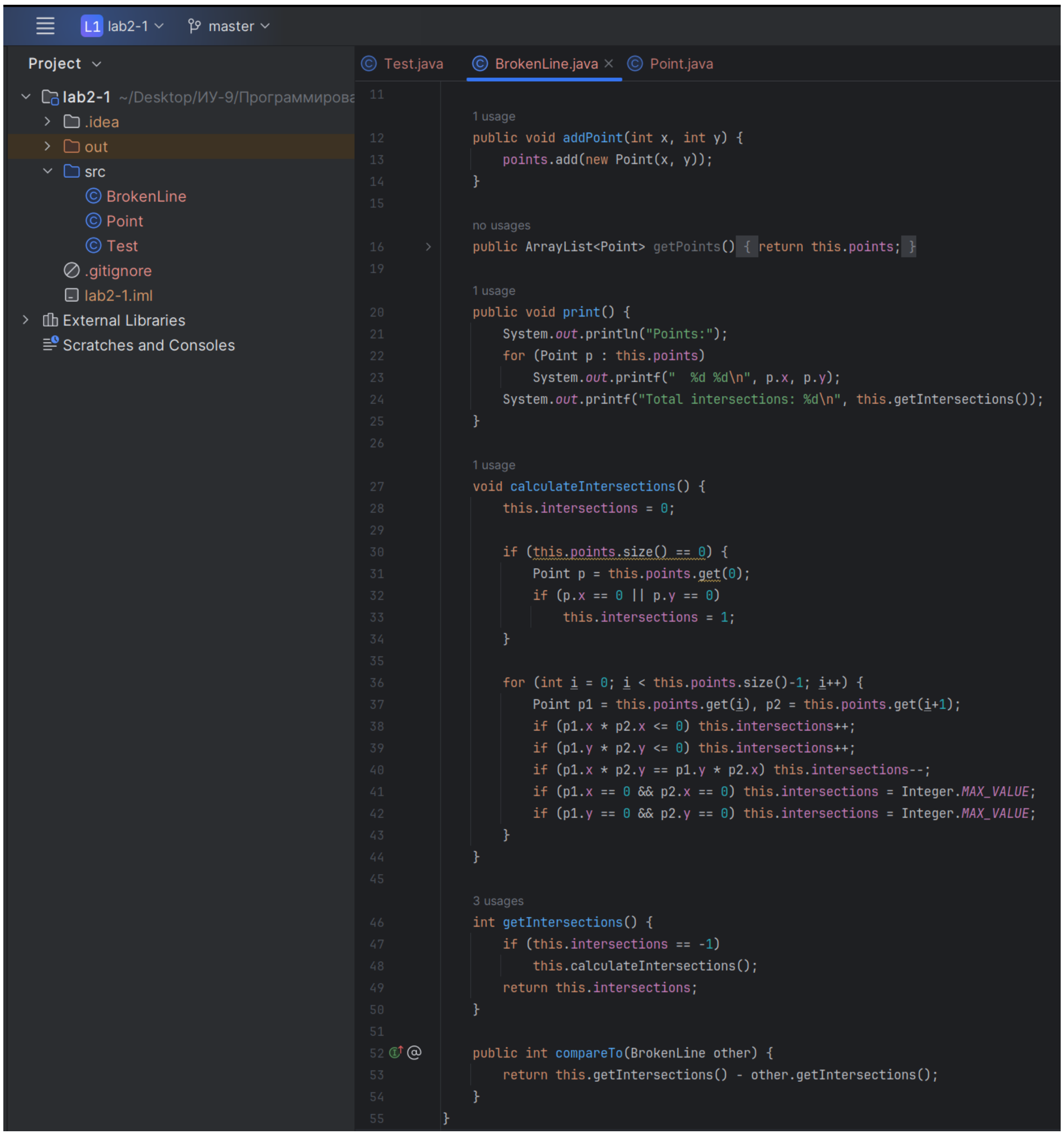
Рис. 1: Пример работы



```java
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();
    }
}
```

```java
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



```java

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

    no usages
    public ArrayList<Point> getPoints() { return this.points; }

    1 usage
    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());
    }

    1 usage
    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;
        }
    }

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

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

Рис. 3: Исходный код ч. 2