Федеральное государственное бюджетное образовательное учреждение
высшего профессионального 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
«Полиморфизм на основе интерфейсов в языке Java (доп. задание)»
по курсу: «Языки и методы программирования»

Выполнил:
Студент группы ИУ9-21Б
Старовойтов А. И.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

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

Задачи

№60: Класс пар окружностей с порядком на основе расстояния между точками пересечения окружностей. При совпадении окружностей считать расстояние нулевым, при непересечении –– бесконечным.

Решение

geometry/CirclePair.java

```
package geometry;

import java.lang.Math;

public class CirclePair implements Comparable<CirclePair> {
    private final Circle first;
    private final Circle second;
    private final double intersectionsDist;
    private final Circle.Intersection intersection;

    public CirclePair(Circle first, Circle second) {
        this.first = first;
        this.second = second;
        this.intersection = Circle.getIntersection(first,
         ↪  second);
        if (intersection.type ==
         ↪  Circle.IntersectionType.OVERLAP
            || intersection.type ==
             ↪  Circle.IntersectionType.TOUCH) {
            this.intersectionsDist = 0;
        } else if (intersection.type ==
         ↪  Circle.IntersectionType.ABSENCE) {
            this.intersectionsDist = Constants.INF;
```

```
        } else {
            this.intersectionsDist =
             ↪  Point.getDist(intersection.first,
             ↪  intersection.second);
        }
    }

    @Override
    public int compareTo(CirclePair obj) {
        if (obj.intersection.type ==
         ↪  Circle.IntersectionType.ABSENCE) {
            if (this.intersection.type ==
             ↪  Circle.IntersectionType.ABSENCE) {
                return 0;
            }
            return -1;
        }
        if (Math.abs(this.intersectionsDist -
         ↪  obj.intersectionsDist) < Constants.EPS) {
            return 0;
        }
        if (this.intersectionsDist < obj.intersectionsDist)
         ↪  {
            return -1;
        }
        return 1;
    }

    @Override
    public String toString() {
        return "Первая окружность:\n"
            + "Центр " + first.center + "\n"
            + "Радиус " + first.radius + "\n"
            + "Вторая окружность:\n"
            + "Центр " + second.center + "\n"
            + "Радиус " + second.radius + "\n"
            + "Расстояние между пересечениями: " +
             ↪  intersectionsDist + "\n"
```

```
            + "Пересечения: " + intersection.type + "(" +
              ↪  intersection.first + ") " +
              ↪  intersection.type + "(" +
              ↪  intersection.second + ")";
    }
}
```

geometry/Circle.java

```
package geometry;

import java.lang.Math;

public class Circle {
    public final Point center;
    public final double radius;

    public static enum IntersectionType {
        OVERLAP,
        INTERSECTION,
        ABSENCE,
        TOUCH;
    }

    public static class Intersection {
        public final Point first;
        public final Point second;
        public final IntersectionType type;

        public Intersection(IntersectionType type) {
            this.type = type;
            this.first = new Point(-1, -1);
            this.second = new Point(-1, -1);
        }

        public Intersection(IntersectionType type, Point
          ↪  first, Point second) {
            this.type = type;
            this.first = first;
            this.second = second;
```

```
        }
    }

    public Circle(Point center, double radius) {
        this.center = center;
        this.radius = radius;
    }

    public static Intersection getIntersection(Circle first,
     ↪  Circle second) {
        if (Point.isEqual(first.center, second.center)) {
            if (Math.abs(first.radius - second.radius) <
             ↪  Constants.EPS) {
                return new
                 ↪  Circle.Intersection(Circle.IntersectionType.OVERLAP);
            }
            return new
             ↪  Circle.Intersection(Circle.IntersectionType.ABSENCE);
        }
        double a = 2 * first.center.x - 2 * second.center.x;
        double b = 2 * first.center.y - 2 * second.center.y;
        double c = Math.pow(second.center.x, 2) +
         ↪  Math.pow(second.center.y, 2) -
         ↪  Math.pow(second.radius, 2)
            - Math.pow(first.center.x, 2) -
             ↪  Math.pow(first.center.y, 2) +
             ↪  Math.pow(first.radius, 2);
        System.out.println(a + " " + b + " " + c);
        Line l = new Line(a, b, c);
        double h = l.getR(first.center);
        if (h > first.radius) {
            return new
             ↪  Circle.Intersection(Circle.IntersectionType.ABSENCE);
        }
        Point s = l.getProjection(first.center);
        Point q = new Point(-l.b, l.a);
        q = q.mult(Math.pow(Math.pow(first.radius, 2) -
↪   Math.pow(h, 2), 0.5) / q.getR());
        if (Math.abs(h - first.radius) < Constants.EPS) {
```

```
            return new
             ↪  Circle.Intersection(Circle.IntersectionType.TOUCH,
             ↪  s.add(q), new Point(-1, -1));
        }
        return new
         ↪  Circle.Intersection(Circle.IntersectionType.INTERSECTION,
         ↪  s.add(q), s.sub(q));
    }
}
```

geometry/Line.java

```
package geometry;

import java.lang.Math;

public class Line {
    public final double a;
    public final double b;
    public final double c;

    public Line(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public boolean isIn(Point p) {
        return Math.abs(a*p.x + b*p.y + c) < Constants.EPS;
    }

    public Point getProjection(Point p) {
        Point n = new Point(a, b);
        n = n.mult(Math.abs(a*p.x + b*p.y + c) /
 ↪  (Math.pow(a, 2) + Math.pow(b, 2)));
        Point res = p.add(n);
        if (this.isIn(res)) {
            return res;
        }
        return p.sub(n);
```

```
    }

    public double getR(Point p) {
        return Math.abs(a*p.x + b*p.y + c) /
         ↪  Math.pow(Math.pow(a, 2) + Math.pow(b, 2), 0.5);
    }
}
```

geometry/Point.java

```
package geometry;

import static java.lang.Math.*;

public class Point {
    public final double x;
    public final double y;

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

    public Point add(Point p) {
        return new Point(this.x + p.x, this.y + p.y);
    }

    public Point sub(Point p) {
        return new Point(this.x - p.x, this.y - p.y);
    }

    public Point mult(double k) {
        return new Point(this.x * k, this.y * k);
    }

    public static boolean isEqual(Point a, Point b) {
        return abs(a.x - b.x) < Constants.EPS && abs(a.y -
         ↪  b.y) < Constants.EPS;
    }
```

```
    public double getR() {
        return pow(pow(this.x, 2) + pow(this.y, 2), 0.5);
    }

    public static double getDist(Point a, Point b) {
        return pow(pow(a.x - b.x, 2) + pow(a.y - b.y, 2),
         ↪  0.5);
    }

    @Override
    public String toString() {
        return x + " " + y;
    }
}
```

geometry/Constants.java

```
package geometry;

public class Constants {
    public static final double EPS = 1e-6;
    public static final double INF = Double.MAX_VALUE;
}
```

Test.java

```
import java.util.Arrays;
import java.util.Random;
import geometry.*;

public class Test {
    public static void main(String[] args) {
        int n = 10;
        Random random = new Random();
        CirclePair[] pairs = new CirclePair[n];
        for (int i = 0; i < n; ++i) {
        pairs[i] = new CirclePair(new Circle(new
 ↪  Point(random.nextInt(5), random.nextInt(5)),
 ↪  random.nextInt(5)+1),
```

```
                                    new Circle(new
                                     ↪  Point(random.nextInt(5),
                                     ↪  random.nextInt(5)),
                                     ↪  random.nextInt(5)+1));
        }

        System.out.println("До сортировки:");
        for (CirclePair pair : pairs) {
            System.out.println(pair);
        }

        Arrays.sort(pairs);

        System.out.println("\n\nОтсортированные:");
        for (CirclePair pair : pairs) {
            System.out.println(pair);
        }
    }
}
```

Пример вывода

```
До сортировки:
Первая окружность:
Центр 2.0 0.0
Радиус 1.0
Вторая окружность:
Центр 3.0 2.0
Радиус 3.0
Расстояние между пересечениями: 1.4832396974191324
Пересечения: INTERSECTION(2.36332495807108
 ↪  -0.9316624790355399) INTERSECTION(1.0366750419289201
 ↪  -0.26833752096446)
Первая окружность:
Центр 2.0 1.0
Радиус 3.0
Вторая окружность:
Центр 3.0 2.0
Радиус 4.0
Расстояние между пересечениями: 4.847679857416329
```

```
Пересечения: INTERSECTION(2.463913650100261
 ↪  -1.963913650100261) INTERSECTION(-0.963913650100261
 ↪  1.463913650100261)
Первая окружность:
Центр 2.0 2.0
Радиус 4.0
Вторая окружность:
Центр 2.0 2.0
Радиус 2.0
Расстояние между пересечениями: 1.7976931348623157E308
Пересечения: ABSENCE(-1.0 -1.0) ABSENCE(-1.0 -1.0)
Первая окружность:
Центр 1.0 1.0
Радиус 3.0
Вторая окружность:
Центр 1.0 4.0
Радиус 2.0
Расстояние между пересечениями: 3.7712361663282534
Пересечения: INTERSECTION(2.8856180831641267
 ↪  3.3333333333333335) INTERSECTION(-0.8856180831641267
 ↪  3.3333333333333335)
Первая окружность:
Центр 0.0 2.0
Радиус 3.0
Вторая окружность:
Центр 2.0 2.0
Радиус 1.0
Расстояние между пересечениями: 0.0
Пересечения: TOUCH(3.0 2.0) TOUCH(-1.0 -1.0)
Первая окружность:
Центр 4.0 0.0
Радиус 3.0
Вторая окружность:
Центр 3.0 3.0
Радиус 2.0
Расстояние между пересечениями: 3.6742346141747673
Пересечения: INTERSECTION(4.992842505793337
 ↪  2.8309475019311128) INTERSECTION(1.5071574942066623
 ↪  1.6690524980688874)
Первая окружность:
```

```
Центр 4.0 4.0
Радиус 3.0
Вторая окружность:
Центр 0.0 2.0
Радиус 5.0
Расстояние между пересечениями: 5.93295878967653
Пересечения: INTERSECTION(2.2733500838578404
 ↪  6.453299832284319) INTERSECTION(4.92664991614216
 ↪  1.14670016771568)
Первая окружность:
Центр 4.0 2.0
Радиус 4.0
Вторая окружность:
Центр 1.0 3.0
Радиус 2.0
Расстояние между пересечениями: 3.9496835316263006
Пересечения: INTERSECTION(1.3244997998398396
 ↪  4.97349939951952) INTERSECTION(0.07550020016015979
 ↪  1.2265006004804802)
Первая окружность:
Центр 1.0 1.0
Радиус 1.0
Вторая окружность:
Центр 3.0 0.0
Радиус 3.0
Расстояние между пересечениями: 1.4832396974191324
Пересечения: INTERSECTION(0.06833752096446005
 ↪  0.6366750419289201) INTERSECTION(0.7316624790355399
 ↪  1.9633249580710799)
Первая окружность:
Центр 2.0 1.0
Радиус 1.0
Вторая окружность:
Центр 2.0 2.0
Радиус 4.0
Расстояние между пересечениями: 1.7976931348623157E308
Пересечения: ABSENCE(-1.0 -1.0) ABSENCE(-1.0 -1.0)


Отсортированные:
```

```
Первая окружность:
Центр 0.0 2.0
Радиус 3.0
Вторая окружность:
Центр 2.0 2.0
Радиус 1.0
Расстояние между пересечениями: 0.0
Пересечения: TOUCH(3.0 2.0) TOUCH(-1.0 -1.0)
Первая окружность:
Центр 2.0 0.0
Радиус 1.0
Вторая окружность:
Центр 3.0 2.0
Радиус 3.0
Расстояние между пересечениями: 1.4832396974191324
Пересечения: INTERSECTION(2.36332495807108
 ↪  -0.9316624790355399) INTERSECTION(1.0366750419289201
 ↪  -0.26833752096446)
Первая окружность:
Центр 1.0 1.0
Радиус 1.0
Вторая окружность:
Центр 3.0 0.0
Радиус 3.0
Расстояние между пересечениями: 1.4832396974191324
Пересечения: INTERSECTION(0.06833752096446005
 ↪  0.6366750419289201) INTERSECTION(0.7316624790355399
 ↪  1.9633249580710799)
Первая окружность:
Центр 4.0 0.0
Радиус 3.0
Вторая окружность:
Центр 3.0 3.0
Радиус 2.0
Расстояние между пересечениями: 3.6742346141747673
Пересечения: INTERSECTION(4.992842505793337
 ↪  2.8309475019311128) INTERSECTION(1.5071574942066623
 ↪  1.6690524980688874)
Первая окружность:
Центр 1.0 1.0
```

```
Радиус 3.0
Вторая окружность:
Центр 1.0 4.0
Радиус 2.0
Расстояние между пересечениями: 3.7712361663282534
Пересечения: INTERSECTION(2.8856180831641267
 ↪  3.3333333333333335) INTERSECTION(-0.8856180831641267
 ↪  3.3333333333333335)
Первая окружность:
Центр 4.0 2.0
Радиус 4.0
Вторая окружность:
Центр 1.0 3.0
Радиус 2.0
Расстояние между пересечениями: 3.9496835316263006
Пересечения: INTERSECTION(1.3244997998398396
 ↪  4.97349939951952) INTERSECTION(0.07550020016015979
 ↪  1.2265006004804802)
Первая окружность:
Центр 2.0 1.0
Радиус 3.0
Вторая окружность:
Центр 3.0 2.0
Радиус 4.0
Расстояние между пересечениями: 4.847679857416329
Пересечения: INTERSECTION(2.463913650100261
 ↪  -1.963913650100261) INTERSECTION(-0.963913650100261
 ↪  1.463913650100261)
Первая окружность:
Центр 4.0 4.0
Радиус 3.0
Вторая окружность:
Центр 0.0 2.0
Радиус 5.0
Расстояние между пересечениями: 5.93295878967653
Пересечения: INTERSECTION(2.2733500838578404
 ↪  6.453299832284319) INTERSECTION(4.92664991614216
 ↪  1.14670016771568)
Первая окружность:
Центр 2.0 2.0
```

```
Радиус 4.0
Вторая окружность:
Центр 2.0 2.0
Радиус 2.0
Расстояние между пересечениями: 1.7976931348623157E308
Пересечения: ABSENCE(-1.0 -1.0) ABSENCE(-1.0 -1.0)
Первая окружность:
Центр 2.0 1.0
Радиус 1.0
Вторая окружность:
Центр 2.0 2.0
Радиус 4.0
Расстояние между пересечениями: 1.7976931348623157E308
Пересечения: ABSENCE(-1.0 -1.0) ABSENCE(-1.0 -1.0)
```