

**Комитет по образованию г. Санкт-Петербург**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ПРЕЗИДЕНТСКИЙ ФИЗИКО-МАТЕМАТИЧЕСКИЙ  
ЛИЦЕЙ №239**

**Отчет о практике  
«Создание графических приложений на языке Java»**

Учащийся 10-2 класса  
Зубарев А.Т.

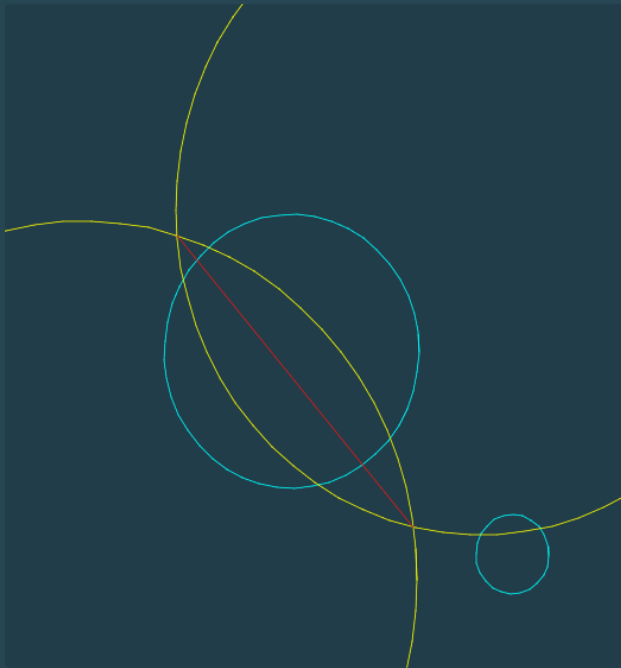
Преподаватель:  
Клюнин А.О.

Санкт-Петербург – 2023 год

# 1. Постановка задачи

На плоскости задано множество окружностей. Найти такую пару пересекающихся окружностей, что длина отрезка, проведенного от одной точки пересечения этих двух окружностей до другой, максимальна. В качестве ответа: выделить эту пару окружностей, нарисовать отрезок между найденными точками пересечения.

Java 2D



ПОСТАНОВКА ЗАДАЧИ:  
На плоскости задано множество окружностей. Найти такую пару пересекающихся окружностей, что длина отрезка, проведенного от одной точки пересечения этих двух окружностей до другой, максимальна.

X

0.0

Y

0.0

R

0.0

Добавить окружность

Кол-во

5

Добавить случайные окружности

Загрузить

Сохранить

Очистить

Сбросить

01:17:18: окружность Circle(center=(-0.87, -0.32), radius=4.132329107653052) добавлена  
01:17:22: окружность Circle(center=(5.26, 4.01), radius=9.88691511889168) добавлена  
01:17:26: окружность Circle(center=(-7.61, -7.20), radius=10.813049175379449) добавлена  
01:17:30: окружность Circle(center=(6.30, -6.45), radius=1.192797384699522) добавлена  
01:17:32: Задача решена  
01:17:32: Окружность №1: Circle(center=(5.26, 4.01), radius=9.88691511889168)  
01:17:32: Окружность №2: Circle(center=(-7.61, -7.20), radius=10.813049175379449)  
01:17:32: Длина пересечения: 11.68620848926567

## 2. Элементы управления

В рамках данной задачи необходимо было реализовать следующие элементы управления:

ПОСТАНОВКА ЗАДАЧИ:  
На плоскости задано множество окружностей. Найти такую пару пересекающихся окружностей, что длина отрезка, проведенного от одной точки пересечения этих двух окружностей до другой, максимальна.

X  Y

R

Кол-во

Для добавления точки по координатам было создано три поля ввода: «X», «Y» и «R» - координата центра окружности по оси X и Y и радиус соответственно. Т.к. задача предполагает только один вид геометрических объектов, то достаточно одной кнопки.

Т.к. задача предполагает только один вид геометрических объектов, то для добавления случайных элементов достаточно одного поля ввода. В него вводится количество случайных точек, которые будут добавлены.

Также программа позволяет добавлять точки с помощью мыши. При нажатии выбирается место центра окружности, а при отпускании мыши задаётся радиус.

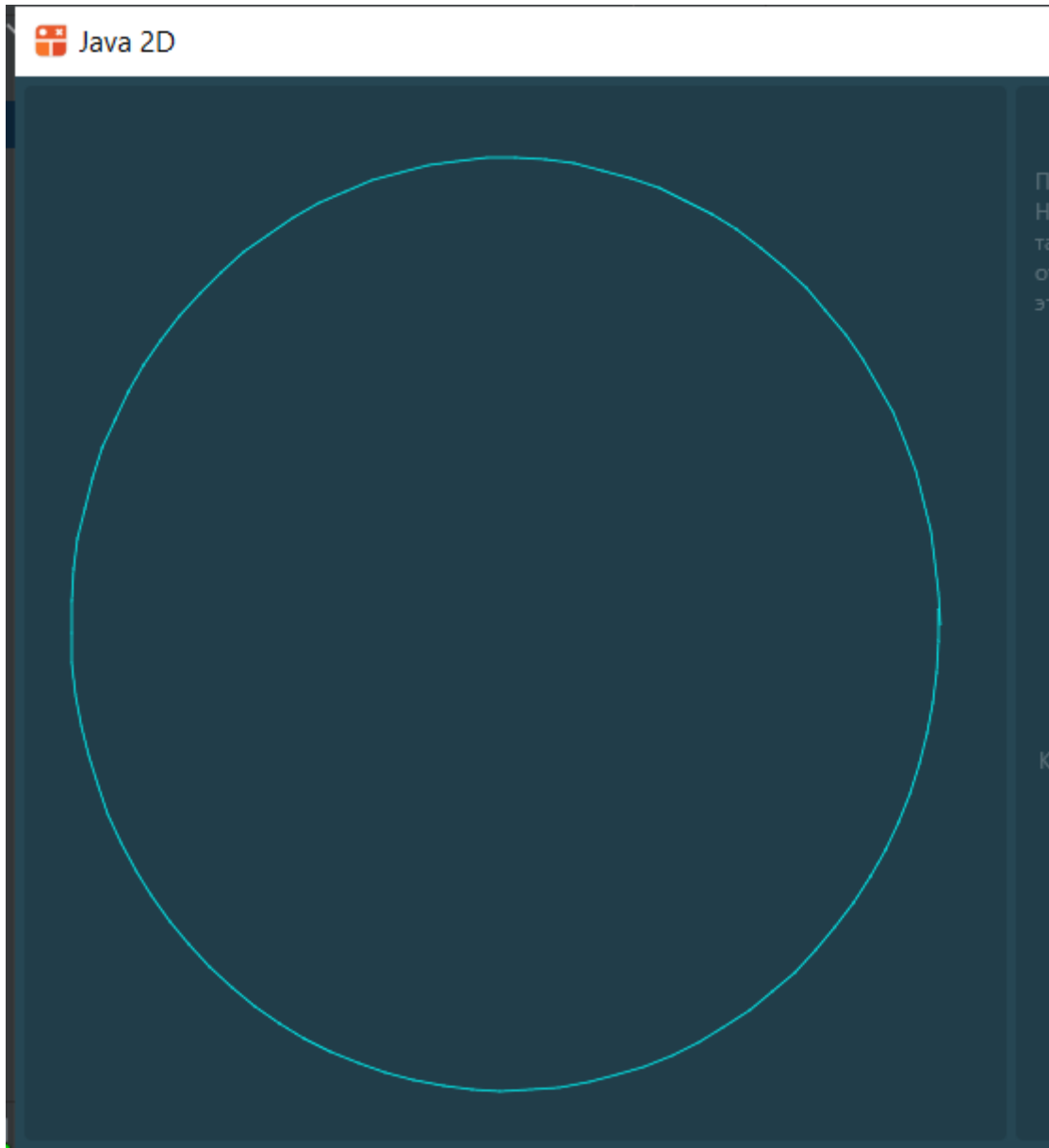
### 3. Структуры данных

Для того чтобы хранить окружности, был разработан класс **Circle.java**. Его листинг приведён в приложении А.

В него были добавлены поля **center**, соответствующее положению центра окружности в пространстве задачи и её радиус - **rad**.

## 4. Рисование

Чтобы нарисовать окружность, использовалась команда рисования окружности **printCircle**, которая в свою очередь базируется на рисовании линий, длина которых составляет  $1/25$  от радиуса окружности. Функция для рисования линий **canvas.drawLine(...)**



## 5. Решение задачи

Для решения поставленной задачи в классе **Task** был разработан метод **solve()**.

```
/**
 * Решить задачу
 */
public void solve() {
    cancel();

    int index1 = -1, index2 = -1;
    double len = 0;
    for (int i = 0; i < circles.size(); ++i) {
        for (int j = i + 1; j < circles.size(); ++j) {
            if (isCrossed(circles.get(i), circles.get(j))) {
                Vector2d[] now = crossing(circles.get(i), circles.get(j));
                double nowLen = Vector2d.subtract(now[0], now[1]).length();
                if (nowLen > len) {
                    len = nowLen;
                    index1 = i;
                    index2 = j;
                    pointing = now;
                }
            }
        }
    }

    if (index1 != -1 && index2 != -1) {
        crossed.add(circles.get(index1));
        crossed.add(circles.get(index2));
        circles.remove(Math.max(index2, index1));
        circles.remove(Math.min(index2, index1));

        // задача решена
        solved = true;
    } else {
        cancel();
    }
}
```

В нём перебираются пары окружностей и, если длина их пересечения больше той, что была ранее, то мы меняем указатели ответа (**index1** и **index2**) на текущие.

После цикла проверяется, что решение возможно, то есть существует хотя бы 1 пара пересекающихся в двух точках окружностей.

Так же созданы функции **crossing** (возвращает точки пересечения окружностей) и **isCrossed** (проверяет, пересекаются ли окружности)

```
private boolean isCrossed (Circle c1, Circle c2) {
    double d = Vector2d.subtract(c1.center, c2.center).length();
    return d < c1.rad + c2.rad && d > Math.abs(c1.rad - c2.rad);
}
```

```

private Vector2d[] crossing (Circle c1, Circle c2) {
    Vector2d d = Vector2d.subtract(c2.center, c1.center);
    double dLen = d.length();
    double l = (c1.rad * c1.rad - c2.rad * c2.rad + dLen * dLen) / (2 *
dLen);
    Vector2d[] points = {new Vector2d(0, 0), new Vector2d(0, 0)};
    if (d.y == 0) {
        points[0].x = dLen * l / d.x;
        points[1].x = points[0].x;
    } else {
        double a = dLen * dLen;
        double b = -2 * dLen * l * d.x;
        double c = dLen * dLen * l * l - c1.rad * c1.rad * d.y * d.y;
        double Discriminant4 = (b / 2) * (b / 2) - a * c;
        points[0].x = (-b/2 + Math.sqrt(Discriminant4)) / a;
        points[1].x = (-b/2 - Math.sqrt(Discriminant4)) / a;
    }
    points[0].y = (dLen * l - points[0].x * d.x) / d.y;
    points[1].y = (dLen * l - points[1].x * d.x) / d.y;
    points[0].add(c1.center);
    points[1].add(c1.center);
    return points;
}

```

## 6. Проверка

Для проверки правильности решённой задачи были разработаны unit-тесты. Их листинг приведён в приложении Б.

### Тест 1

Окружности:  $\{ (0, 0, 5); (-2, -2, 5) \}$

Ответ:  $\{ (0, 0, 5); (-2, -2, 5) \}$

### Тест 2

Окружности:  $\{ (0, 0, 5); (0, 0, 3) \}$

Ответ:  $\{ \}$

### Тест 3

Окружности:  $\{ (0, 0, 5); (0, 4, 2); (0, -4, 3); (0, 0, 1) \}$

Ответ:  $\{ (0, 0, 5); (0, -4, 3) \}$



## 7. Заключение

В рамках выполнения поставленной задачи было создано графическое приложение с требуемым функционалом. Правильность решения задачи проверена с помощью юнит-тестов.

## Приложение А. Circle.java

```
package app;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import misc.Vector2d;

import java.util.Objects;

/**
 * Класс окружности
 */
public class Circle{
    /**
     * Координаты центра окружности
     */
    public final Vector2d center;

    /**
     * Радиус Окружности
     */
    public final double rad;

    /**
     * Конструктор точки
     *
     * @param center    положение центра окружности
     * @param radius    радиус окружности
     */
    @JsonCreator
    public Circle(@JsonProperty("center") Vector2d center,
        @JsonProperty("radius") double radius) {
        this.center = center;
        this.rad = radius;
    }

    /**
     * Получить положение
     * (нужен для json)
     *
     * @return положение
     */
    public Vector2d getCenter() {
        return center;
    }

    /**
     * Строковое представление объекта
     *
     * @return строковое представление объекта
     */
    @Override
    public String toString() {
        return "Circle{" +
            "center=" + center +
            ", radius=" + rad +
            '}';
    }
}
```

```
/**
 * Получить хэш-код объекта
 *
 * @return хэш-код объекта
 */
@Override
public int hashCode() {
    return Objects.hash(center, rad);
}
}
```

## Приложение Б. UnitTest.java

```
import app.Circle;
import app.Task;
import misc.CoordinateSystem2d;
import misc.Vector2d;
import org.junit.Test;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

/**
 * Класс тестирования
 */
public class UnitTest {

    /**
     * Тест
     *
     * @param circles список окружностей
     * @param crossed окружности в ответе
     */
    private static void test(ArrayList<Circle> circles, Circle[] crossed) {
        Task task = new Task(new CoordinateSystem2d(10, 10, 20, 20),
circles);
        task.solve();

        // проверяем размерности массивов
        assert task.getCrossed().size() == 2 || task.getCrossed().isEmpty();
        assert task.getPointing().length == 2;

        if (!task.getCrossed().isEmpty()) {
            // проверяем, что ответ правильный
            assert
crossed[0].toString().equals(task.getCrossed().get(0).toString());
            assert
crossed[1].toString().equals(task.getCrossed().get(1).toString());
        }
    }

    /**
     * Первый тест
     */
    @Test
    public void test1() {
        ArrayList<Circle> circles = new ArrayList<>();
        Circle[] crossed = new Circle[2];

        circles.add(new Circle(new Vector2d(0, 0), 5));
        circles.add(new Circle(new Vector2d(-2, -2), 5));

        crossed[0] = (new Circle(new Vector2d(0, 0), 5));
        crossed[1] = (new Circle(new Vector2d(-2, -2), 5));

        test(circles, crossed);
    }

    /**
     * Второй тест
     */
}
```

```

@Test
public void test2() {
    ArrayList<Circle> circles = new ArrayList<>();
    Circle[] crossed = new Circle[2];

    circles.add(new Circle(new Vector2d(0, 0), 5));
    circles.add(new Circle(new Vector2d(0, 0), 3));

    test(circles, crossed);
}

/**
 * Третий тест
 */
@Test
public void test3() {
    ArrayList<Circle> circles = new ArrayList<>();
    Circle[] crossed = new Circle[2];

    circles.add(new Circle(new Vector2d(0, 0), 5));
    circles.add(new Circle(new Vector2d(0, 4), 2));
    circles.add(new Circle(new Vector2d(0, -4), 3));
    circles.add(new Circle(new Vector2d(0, 0), 1));

    crossed[0] = (new Circle(new Vector2d(0, 0), 5));
    crossed[1] = (new Circle(new Vector2d(0, -4), 3));

    test(circles, crossed);
}
}

```