**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 10-1 класса

Чашин А. Е.

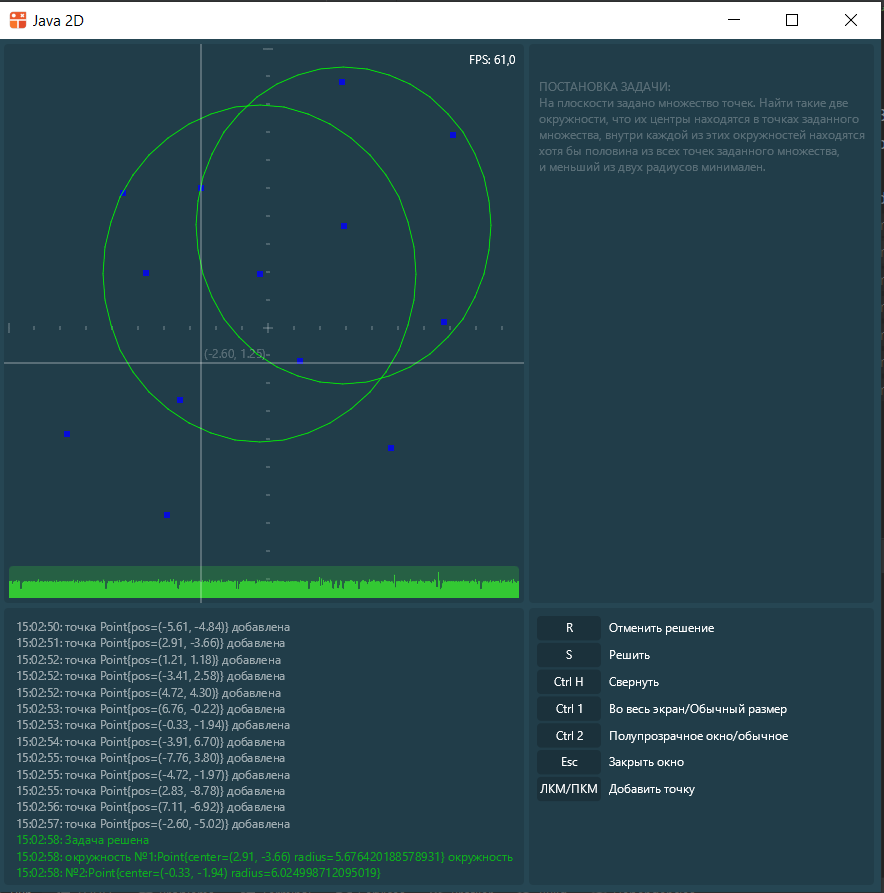
Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2023 год

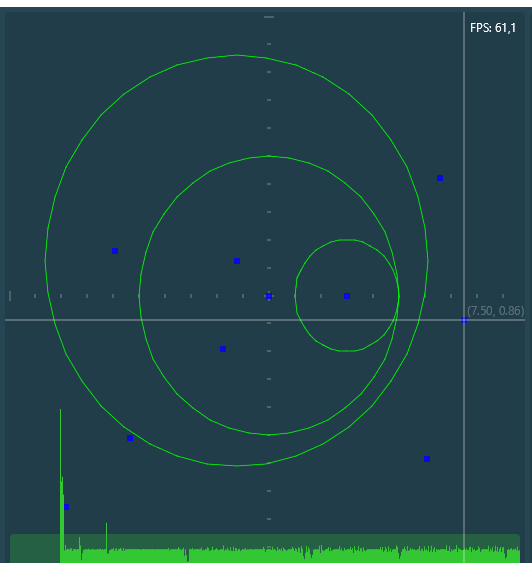
# 1. Постановка задачи

На плоскости задано множество точек. Найти такие две окружности, что их центры находятся в точках заданного множества, внутри каждой из этих окружностей находятся хотя бы половина из всех точек заданного множества, и меньший из двух радиусов минимален.



# 2. Элементы управления

Программа позволяет добавлять точки с помощью мыши. При клике на область рисование, появляется новая точка.



# 3. Структуры данных

Для того чтобы хранить точки, был разработан класс **Point.java.,** окружности - **Circle.java** Их листинг приведён в приложении А.

У окружности были добавлены поля **pos**, соответствующее положению центра окружности в пространстве задачи и её радиус - **r.**

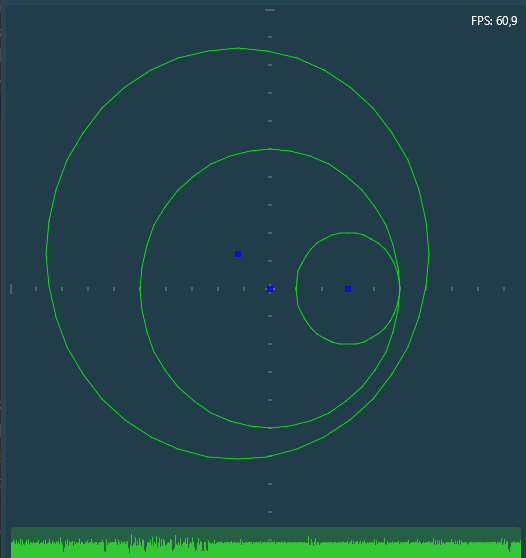
У точки было добавлено поле **pos**, соответствующее положению точки в пространстве задачи.

# 4. Рисование

Для рисования точки использовалась команда **canvas.drawRect(…)**

Для рисования окружности команда **paint()** просчитывала точки, которые, соединив их линиями, образовывали заданную окружность.

Для рисования линий использовалась команда **canvas.drawLines(…)**



# 5. Решение

Для каждой точки создать список расстояний до всех остальных. Отсортировать эти списки. Числа в центре этих списков и будут минимальные радиусы окружностей, с центрами в соответствующей точке, которые захватывают хотя бы половину всех точек. После найти 2 минимальных радиуса и по соответствующим точкам построить окружности.

# 6. Проверка

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

Тест 1

Точки: {(0, 0), (3, 0), (2, 2), (-8, -7), (-3, 2)}

Тест 2

Точки: {(-10, -10), (10, 10), (8, 8), (-3, -3), (0, 0)}

Тест 3

Точки: {(1, 0), (9, 0), (-3, 2), (-10, -7), (10, 2)}

# 7. Заключение

В рамках выполнения поставленной задачи было создано графическое приложение с требуемым функционалом.

# Приложение А. Point.java; Circle.java

package app;  
  
import misc.Misc;  
import misc.Vector2d;  
  
import java.util.Objects;  
  
*/\*\*  
 \* Класс точки  
 \*/*public class Point {  
 */\*\*  
 \* Координаты точки  
 \*/* public final Vector2d pos;  
  
 */\*\*  
 \* Конструктор точки  
 \*  
 \** ***@param*** *pos положение точки  
 \*/* public Point(Vector2d pos) {  
 this.pos = pos;  
 }  
  
  
 */\*\*  
 \* Получить цвет точки  
 \*  
 \** ***@return*** *цвет точки  
 \*/* public int getColor() {  
 return Misc.*getColor*(0xCC, 0x00, 0x00, 0xFF);  
 }  
  
 */\*\*  
 \* Строковое представление объекта  
 \*  
 \** ***@return*** *строковое представление объекта  
 \*/* @Override  
 public String toString() {  
 return "Point{" +  
 "pos=" + pos +  
 '}';  
 }  
  
 */\*\*  
 \* Получить хэш-код объекта  
 \*  
 \** ***@return*** *хэш-код объекта  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(pos);  
 }  
}

package app;  
  
import misc.Misc;  
import misc.Vector2d;  
  
import java.util.Objects;  
  
*/\*\*  
 \* Класс окружности  
 \*/*public class Circle {  
 */\*\*  
 \* Координаты центра  
 \*/* public final Vector2d pos;  
  
 */\*\*  
 \* радиус  
 \*/* public final double r;  
  
 */\*\*  
 \* Конструктор окружности  
 \*  
 \** ***@param*** *pos положение окружности  
 \** ***@param*** *r радиус окружности  
 \*/* public Circle(Vector2d pos, double r) {  
 this.pos = pos;  
 this.r = r;  
 }  
  
  
 */\*\*  
 \* Получить цвет окружности  
 \*  
 \** ***@return*** *цвет окружности  
 \*/* public int getColor() {  
 return Misc.*getColor*(0xCC, 0x00, 0xFF, 0x0);  
 }  
  
 */\*\*  
 \* получить точки для рисования окружности  
 \** ***@return*** *точки  
 \*/* public float[] paint() {  
 int loopCnt = 40;  
 // создаём массив координат опорных точек  
 float[] points = new float[loopCnt \* 4];  
 for (int i = 0; i < loopCnt; ++i) {  
 // x координата первой точки  
 points[i \* 4] = (float) (pos.x + r \* Math.*cos*(Math.*PI* / 20 \* i));  
 // y координата первой точки  
 points[i \* 4 + 1] = (float) (pos.y + r \* Math.*sin*(Math.*PI* / 20 \* i));  
  
 // x координата второй точки  
 points[i \* 4 + 2] = (float) (pos.x + r \* Math.*cos*(Math.*PI* / 20 \* (i + 1)));  
 // y координата точки  
 points[i \* 4 + 3] = (float) (pos.y + r \* Math.*sin*(Math.*PI* / 20 \* (i + 1)));  
 }  
 return points;  
 }  
  
 */\*\*  
 \* Строковое представление объекта  
 \*  
 \** ***@return*** *строковое представление объекта  
 \*/* @Override  
 public String toString() {  
 return "Point{" +  
 "center=" + pos +  
 " radius=" + r +  
 '}';  
 }  
  
 */\*\*  
 \* Получить хэш-код объекта  
 \*  
 \** ***@return*** *хэш-код объекта  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(pos, r);  
 }  
}

# Приложение Б. UnitTest.java

//import app.Point;  
import app.Circle;  
import app.Point;  
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 {  
  
 */\*\*  
 \* Проверяет, находится ли точка в окружности  
 \** ***@param*** *point точка  
 \** ***@param*** *circle окружность  
 \** ***@return*** *результат проверки  
 \*/* private static boolean isInside(Point point, Circle circle) {  
 return Vector2d.*subtract*(point.pos, circle.pos).length() <= circle.r;  
 }  
 */\*\*  
 \* Тело проверки  
 \** ***@param*** *points исходные данные  
 \*/* private static void test(ArrayList<Point> points) {  
 Task task = new Task(new CoordinateSystem2d(-10, -10, 10, 10), points);  
 task.solve();  
 ArrayList<Circle> circles = task.getCircles();  
  
 int count0 = 0;  
 int count1 = 0;  
 for (Point p: points) {  
 if (*isInside*(p, circles.get(0))) ++count0;  
 if (*isInside*(p, circles.get(1))) ++count1;  
 }  
  
 assert count0 >= points.size() / 2 && count1 >= points.size() / 2;  
 }  
  
 */\*\*  
 \* Тест 1  
 \*/* @Test  
 public void test1() {  
 ArrayList<Point> points = new ArrayList<>();  
 points.add(new Point(new Vector2d(0, 0)));  
 points.add(new Point(new Vector2d(3, 0)));  
 points.add(new Point(new Vector2d(2, 2)));  
 points.add(new Point(new Vector2d(-8, -7)));  
 points.add(new Point(new Vector2d(-3, 2)));  
  
 *test*(points);  
 }  
 */\*\*  
 \* Тест 2  
 \*/* @Test  
 public void test2() {  
 ArrayList<Point> points = new ArrayList<>();  
 points.add(new Point(new Vector2d(-10, -10)));  
 points.add(new Point(new Vector2d(10, 10)));  
 points.add(new Point(new Vector2d(8, 8)));  
 points.add(new Point(new Vector2d(-3, -3)));  
 points.add(new Point(new Vector2d(0, 0)));  
  
 *test*(points);  
 }  
 */\*\*  
 \* Тест 3  
 \*/* @Test  
 public void test3() {  
 ArrayList<Point> points = new ArrayList<>();  
 points.add(new Point(new Vector2d(1, 0)));  
 points.add(new Point(new Vector2d(9, 0)));  
 points.add(new Point(new Vector2d(-3, 2)));  
 points.add(new Point(new Vector2d(-10, -7)));  
 points.add(new Point(new Vector2d(10, 2)));  
  
 *test*(points);  
 }  
}