**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 10-3 класса

Тестов М.Е.

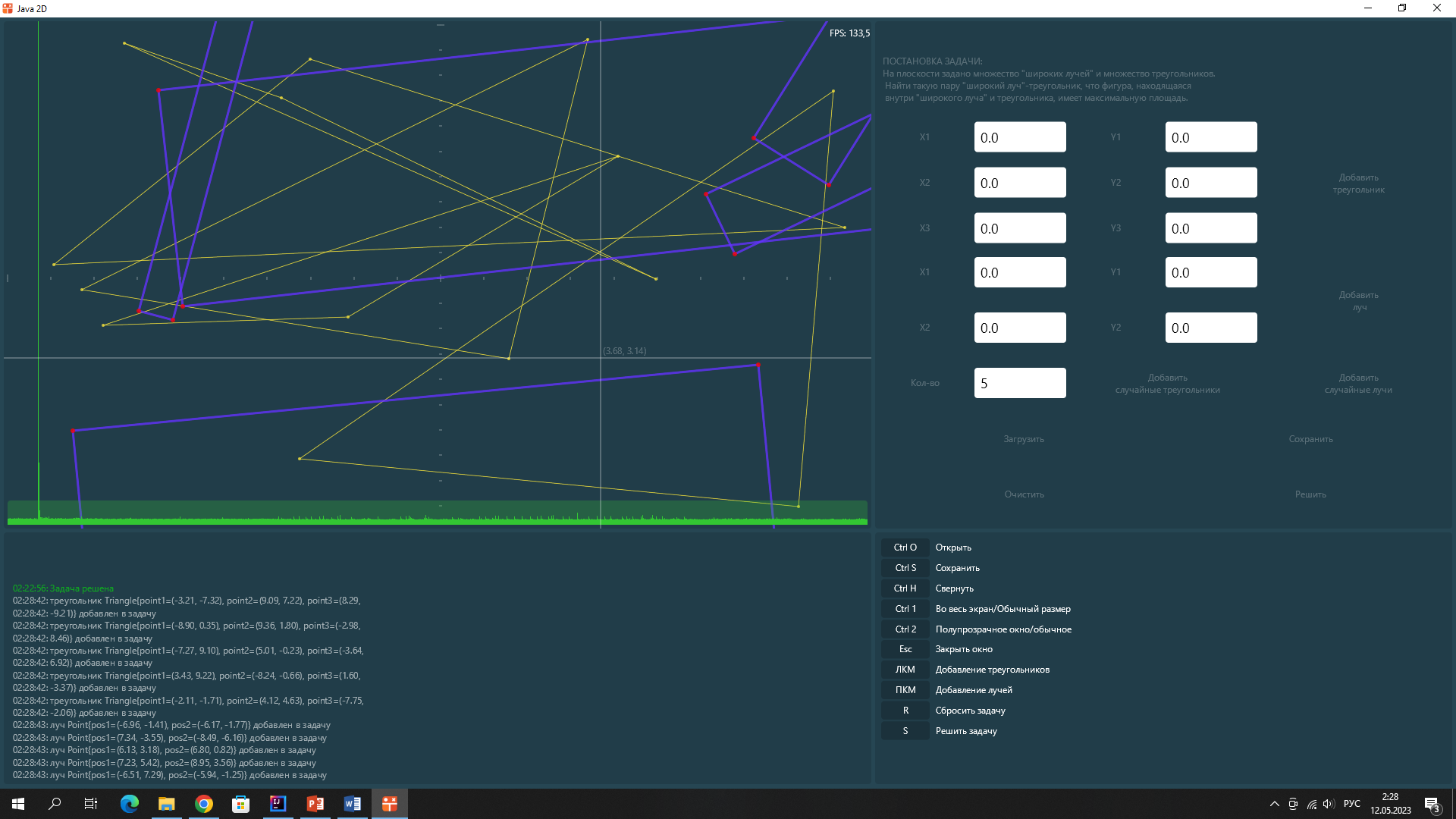
Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2022 год

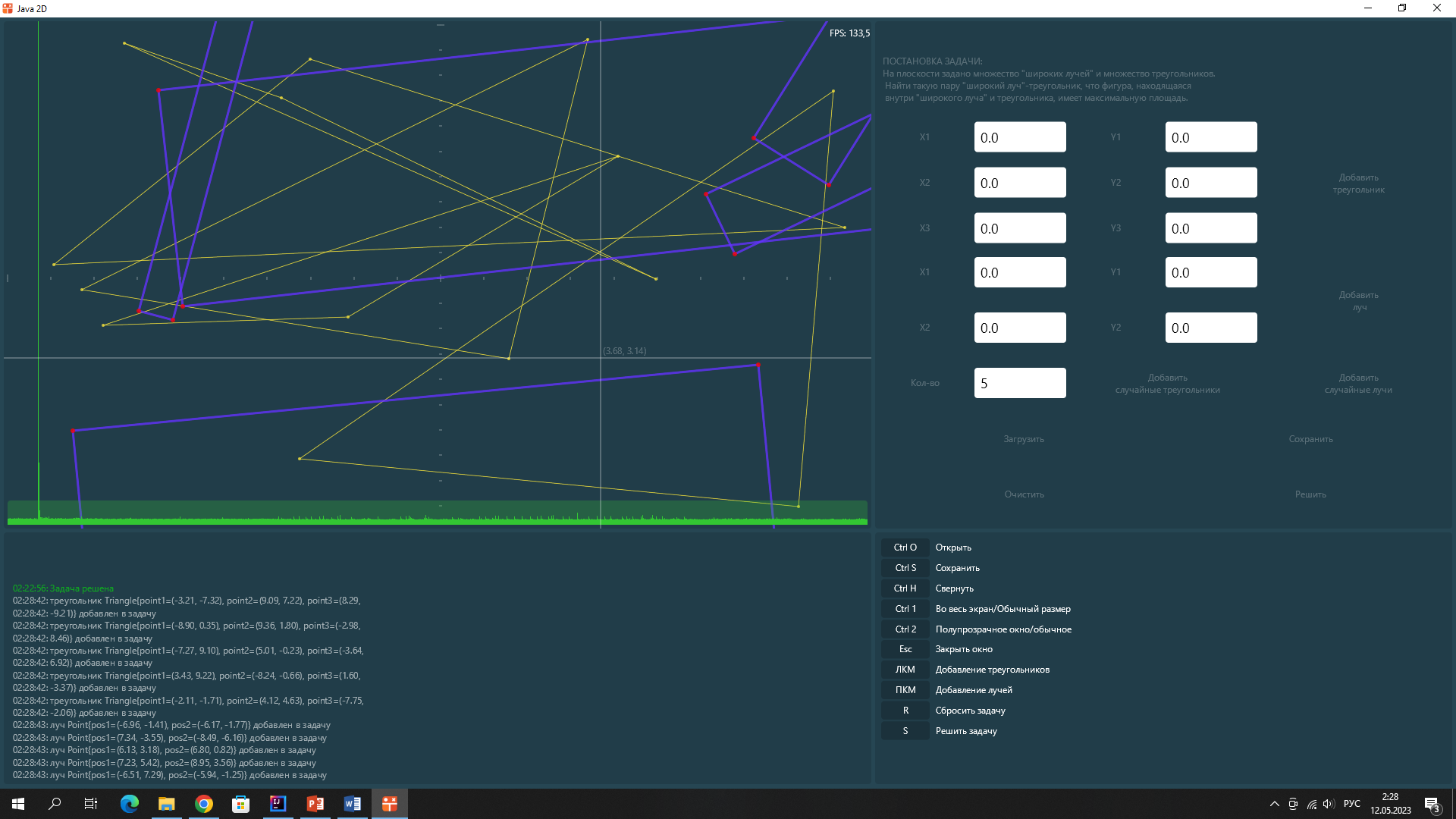
# 1. Постановка задачи

На плоскости задано множество "широких лучей" и множество треугольников. Найти такую пару "широкий луч"-треугольник, что фигура, находящаяся внутри "широкого луча" и треугольника, имеет максимальную площадь. В качестве ответа: выделить найденные "широкий луч" и треугольник, выделить контур фигуры, которая ограничивает точки внутри найденного "широкого луча" и треугольника, желательно выделить внутреннее пространство фигуры ("залить цветом").



# 2. Элементы управления

В рамках данной задачи необходимо было реализовать следующие элементы управления:



Для добавления треугольника по координатам было создано шесть полей:

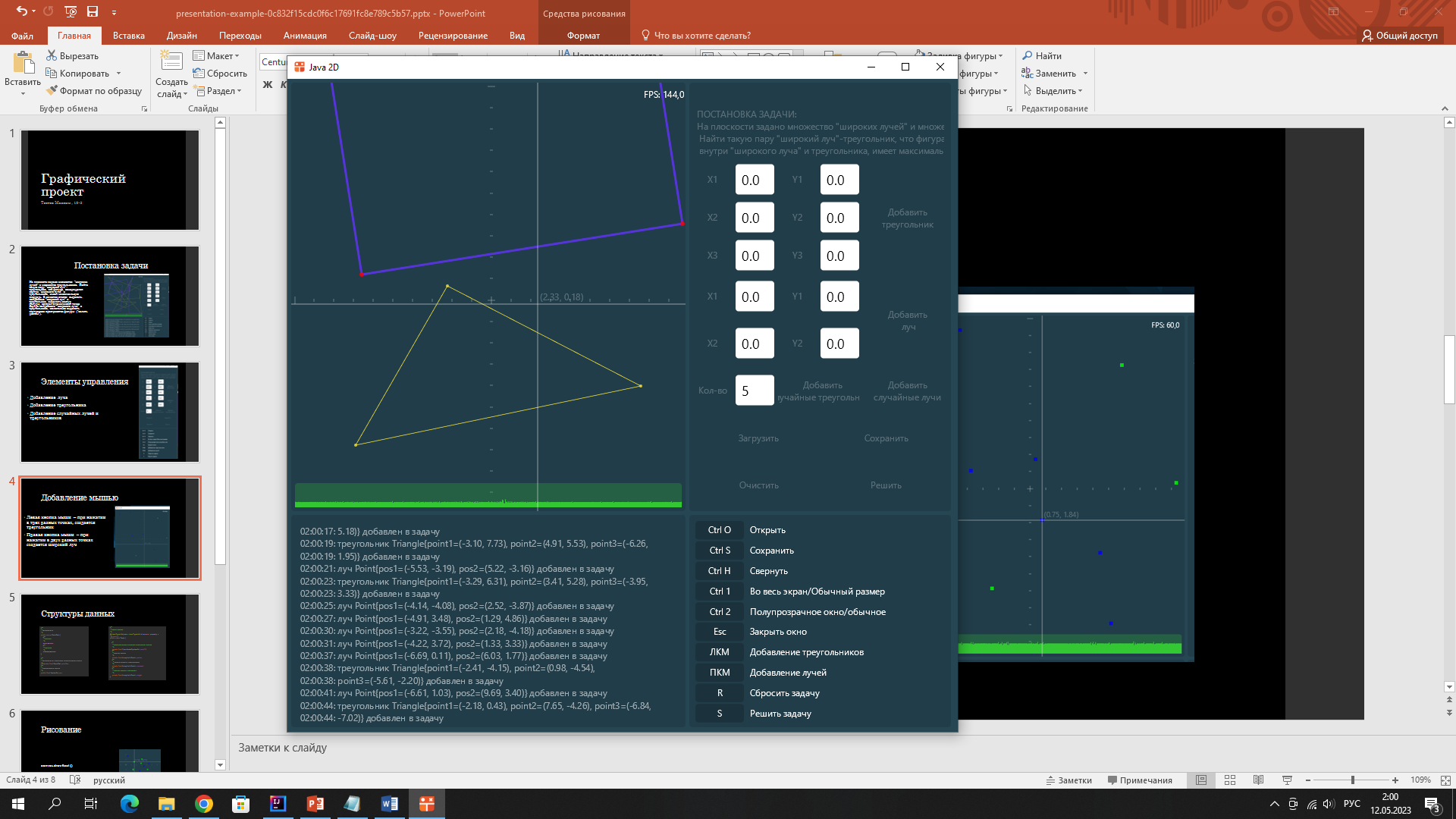
«X1» и «Y1», «X2» и «Y2», «X3» и «Y3».

Аналогично, для добавления широких лучей было создано еще 4 поля: «X1» и «Y1», «X2» и «Y2».

Для добавления случайных элементов достаточно одного поля ввода. В него вводится количество случайных треугольников или лучей, которые будут добавлены при нажатии на соответствующую кнопку.

Также программа позволяет добавлять треугольники с помощью клика левой кнопкой мышки по области рисования и широкие лучи при нажатии правой.

При клике левой кнопкой мыши по области рисования в месте клика создаётся точка, принадлежащая первому множеству, при клике правой - второму



# 3. Структуры данных

Для того чтобы хранить точки (для последующего решения задачи), треугольники и широкие лучи, были разработаны классы **Point.java, Triangle.java, Ray.java.** Их листинг приведён в приложении А.

**Point.java:**

В него были добавлены поля **pos**, соответствующее положению точки в пространстве задачи.

**Triangle.java:**

В него были добавлены поля: **point1**, **point2**, **point3**, которые определяют положение вершин треугольника в пространстве задачи.

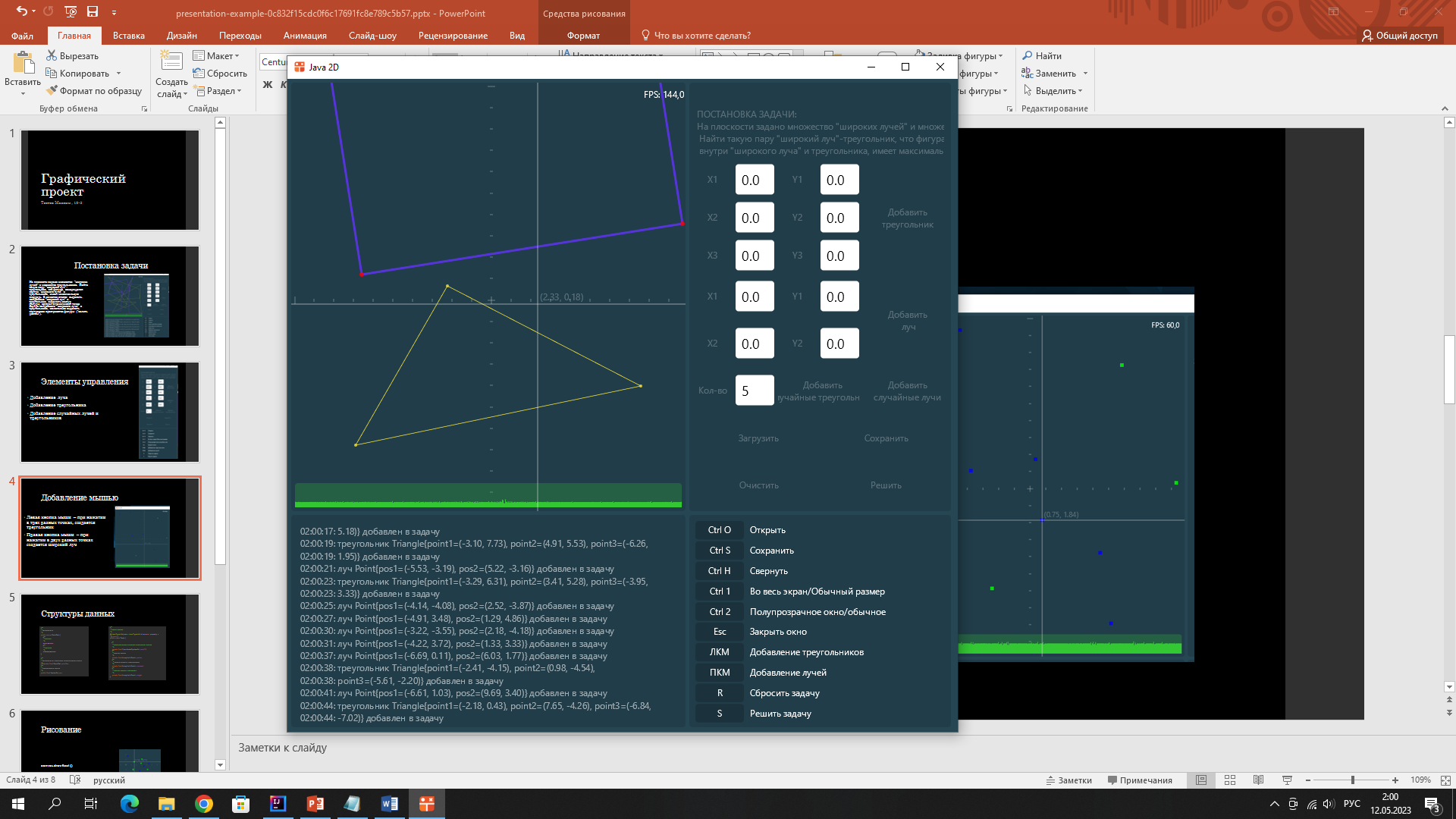
**Ray.java:**

В него были добавлены поля: **pos1**, **pos2**, которые определяют положение вершин луча в пространстве задачи.

# 4. Рисование

Чтобы нарисовать точку, использовалась команда рисования прямоугольников **canvas.drawRect().**

Чтобы нарисовать треугольники и лучи, использовалась команда рисования прямых **canvas.drawline().**



# 5. Решение задачи

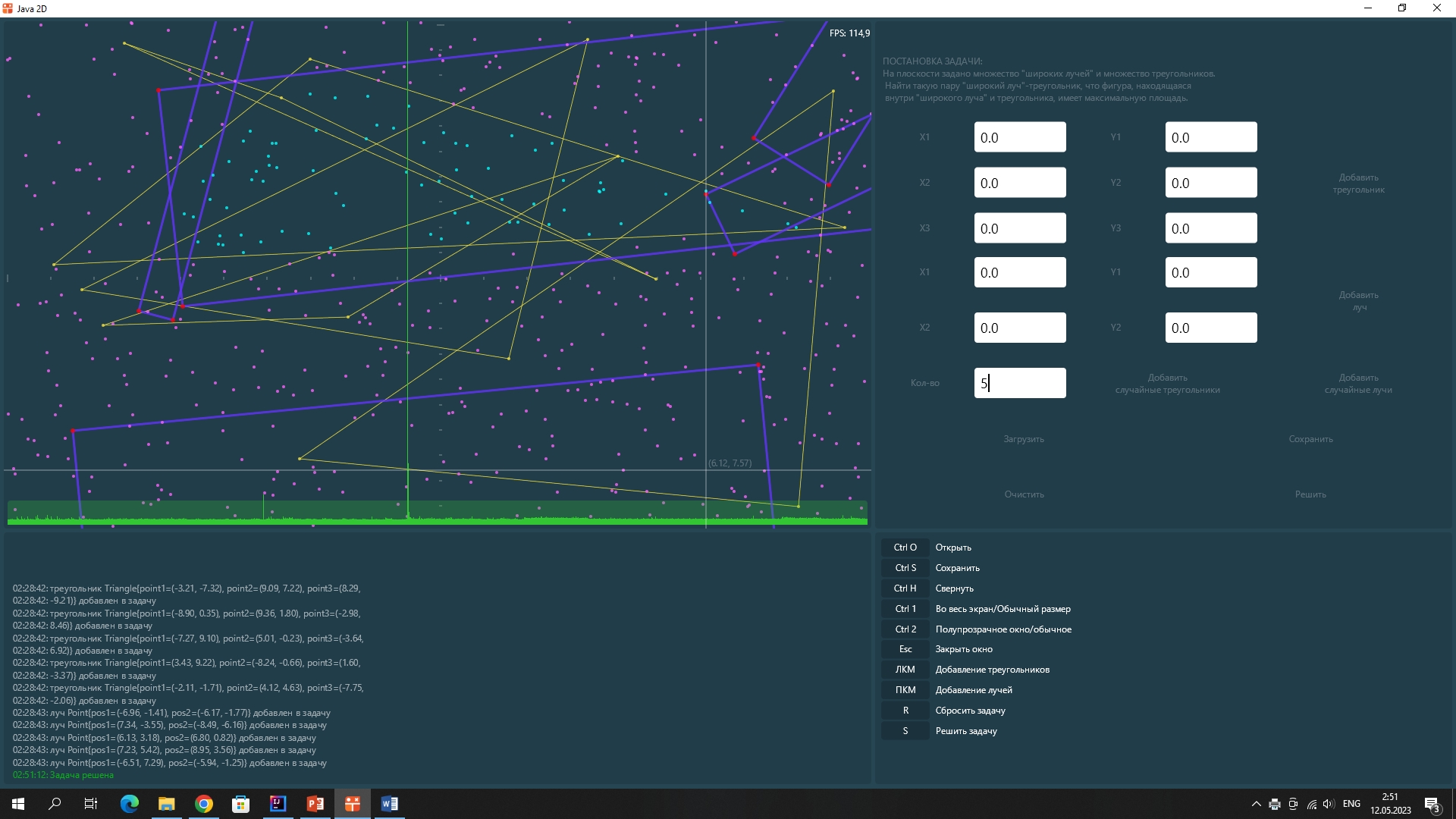
Для решения поставленной задачи в классе **Task** был разработан метод **solve().**

*/\*\*  
 \* Решить задачу  
 \*/*public void solve() {  
 // выделяем область в которой будем раскидывать точки  
 if (triangles.size() != 0 && rays.size() != 0) {  
 int[] na = new int[triangles.size()];  
 int[] da = new int[rays.size()];  
  
 for (int i = 0; i < 500; i++) {  
 Vector2d pos = ownCS.getRandomCoords();  
 addPoint(pos);  
 }  
 int max = -1;  
 for (Triangle t : triangles) {  
 for (Ray r : rays) {  
 int c = 0;  
 for (Point p : points) {  
 if (check\_tri(t, p) && check\_ray(r, p, lastWindowCS))  
 c++;  
  
 }  
 if (c > max) {  
 max = c;  
 *t1* = t;  
 *r1* = r;  
 }  
 }  
 }  
  
 // задача решена  
 solved = true;  
 } else{  
 PanelLog.*info*("добавьте лучи и треугольники");  
 }  
}

А так же методы определения лежат ли точки внутри треугольника или луча соответственно

*/\*\*  
 \*   
 \* лежит ли точка в треугольнике  
 \*/*public boolean check\_tri(Triangle a, Point b) {  
 double st = Math.*abs*((a.point2.x - a.point1.x) \* (a.point3.y - a.point1.y) - (a.point3.x - a.point1.x) \* (a.point2.y - a.point1.y));  
 double st1 = Math.*abs*((a.point2.x - b.pos.x) \* (a.point3.y - b.pos.y) - (a.point3.x - b.pos.x) \* (a.point2.y - b.pos.y));  
 double st2 = Math.*abs*((b.pos.x - a.point1.x) \* (a.point3.y - a.point1.y) - (a.point3.x - a.point1.x) \* (b.pos.y - a.point1.y));  
 double st3 = Math.*abs*((a.point2.x - a.point1.x) \* (b.pos.y - a.point1.y) - (b.pos.x - a.point1.x) \* (a.point2.y - a.point1.y));  
 if (Math.*abs*(st - (st1 + st2 + st3)) < 0.000005)  
 return true;  
 else  
 return false;  
}  
  
*/\*\*  
 \*  
 \* лежит ли точка в луче  
 \*/*public boolean check\_ray(Ray a, Point b , CoordinateSystem2i windowCS) {  
 Vector2d AB = Vector2d.*subtract*(a.pos1, a.pos2);  
  
 // создаём вектор направления для рисования условно бесконечной полосы  
 Vector2d dir = AB;  
 //System.out.print("fx:"+dir.x+" fy:"+dir.y);  
 dir = dir.rotated(Math.*PI* / 2 + Math.*PI*).norm();  
 dir.mult(50);  
 Vector2d direction = new Vector2d((int) dir.x, (int) dir.y);  
 // получаем точки рисования  
 //System.out.println(" x:"+dir.x+" y:"+dir.y);  
 Vector2d renderPointC = Vector2d.*sum*(a.pos1, direction);  
 Vector2d renderPointD = Vector2d.*sum*(a.pos2, direction);  
  
 double sta = Math.*abs*((a.pos2.x - a.pos1.x) \* (renderPointC.y - a.pos1.y) - (renderPointC.x - a.pos1.x) \* (a.pos2.y - a.pos1.y)) / 2;  
 double stb = Math.*abs*((a.pos2.x - renderPointD.x) \* (renderPointC.y - renderPointD.y) - (renderPointC.x - renderPointD.x) \* (a.pos2.y - renderPointD.y)) / 2;  
 double st1 = Math.*abs*((a.pos2.x - a.pos1.x) \* (b.pos.y - a.pos1.y) - (b.pos.x - a.pos1.x) \* (a.pos2.y - a.pos1.y)) / 2;  
 double st2 = Math.*abs*((a.pos2.x - renderPointD.x) \* (b.pos.y - renderPointD.y) - (b.pos.x - renderPointD.x) \* (a.pos2.y - renderPointD.y)) / 2;  
 double st3 = Math.*abs*((renderPointC.x - renderPointD.x) \* (b.pos.y - renderPointD.y) - (b.pos.x - renderPointD.x) \* (renderPointC.y - renderPointD.y)) / 2;  
 double st4 = Math.*abs*((renderPointC.x - a.pos1.x) \* (b.pos.y - a.pos1.y) - (b.pos.x - a.pos1.x) \* (renderPointC.y - a.pos1.y)) / 2;  
 if (Math.*abs*(sta + stb - (st1+st2+st3+st4)) < 0.00005)  
 return true;  
 else  
 return false;  
}

В **solve()** перебираются все пары треугольник-луч и выбирается пара, в которой оказалось максимальное количество точек



# 6. Проверка

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

Было создано три теста, каждый из которых проверят правильность принадлежности точек к лучам и треугольникам в разных обстоятельствах.

# 7. Заключение

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

# Приложение А. Point.java

package app;  
  
import Misc.Misc;  
import Misc.Vector2d;  
import com.fasterxml.jackson.annotation.JsonCreator;  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import com.fasterxml.jackson.annotation.JsonProperty;  
  
import java.util.Objects;  
  
*/\*\*  
 \* Класс точки  
 \*/*public class Point {  
 */\*\*  
 \* Множества  
 \*/* public enum PointSet {  
 */\*\*  
 \* Первое  
 \*/  
 FIRST\_SET*,  
 */\*\*  
 \* Второе  
 \*/  
 SECOND\_SET* }  
  
 */\*\*  
 \* Координаты точки  
 \*/* public final Vector2d pos;  
  
  
 */\*\*  
 \* Конструктор точки  
 \*  
 \** ***@param*** *pos положение точки  
 \*/* @JsonCreator  
 public Point(@JsonProperty("pos") Vector2d pos) {  
 this.pos = pos;  
 }  
  
  
  
 */\*\*  
 \* Получить положение  
 \* (нужен для json)  
 \*  
 \** ***@return*** *положение  
 \*/* public Vector2d getPos() {  
 return pos;  
 }  
  
  
  
 */\*\*  
 \* Строковое представление объекта  
 \*  
 \** ***@return*** *строковое представление объекта  
 \*/* @Override  
 public String toString() {  
 return "Point{" +  
 "pos=" + pos +  
 '}';  
 }  
  
 */\*\*  
 \* Проверка двух объектов на равенство  
 \*  
 \** ***@param*** *o объект, с которым сравниваем текущий  
 \** ***@return*** *флаг, равны ли два объекта  
 \*/* @Override  
 public boolean equals(Object o) {  
 // если объект сравнивается сам с собой, тогда объекты равны  
 if (this == o) return true;  
 // если в аргументе передан null или классы не совпадают, тогда объекты не равны  
 if (o == null || getClass() != o.getClass()) return false;  
 // приводим переданный в параметрах объект к текущему классу  
 Point point = (Point) o;  
 return Objects.*equals*(pos, point.pos);  
 }  
  
 */\*\*  
 \* Получить хэш-код объекта  
 \*  
 \** ***@return*** *хэш-код объекта  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(pos);  
 }  
}

Triangle.java

package app;  
  
import com.fasterxml.jackson.annotation.JsonCreator;  
import com.fasterxml.jackson.annotation.JsonProperty;  
import Misc.Vector2d;  
  
import java.util.Objects;  
  
public class Triangle {  
 */\*\*  
 \* Координаты 1-ой точки  
 \*/* public Vector2d point1;  
  
 */\*\*  
 \* Координаты 2-ой точки  
 \*/* public Vector2d point2;  
  
 */\*\*  
 \* Координаты 3-ой точки  
 \*/* public Vector2d point3;  
  
 */\*\*  
 \* Конструктор треугольника  
 \*  
 \** ***@param*** *point1 положение 1-ой точки  
 \** ***@param*** *point2 положение 2-ой точки  
 \** ***@param*** *point3 положение 3-ой точки  
 \*/* @JsonCreator  
 public Triangle(@JsonProperty("point1") Vector2d point1, @JsonProperty("point2") Vector2d point2, @JsonProperty("point3") Vector2d point3) {  
 this.point1 = point1;  
 this.point2 = point2;  
 this.point3 = point3;  
 }  
  
 */\*\*  
 \* Получить положение 1-ой точки  
 \*  
 \** ***@return*** *point1  
 \*/* public Vector2d getPoint1() {  
 return point1;  
 }  
  
 */\*\*  
 \* Получить положение 2-ой точки  
 \*  
 \** ***@return*** *point2  
 \*/* public Vector2d getPoint2() {  
 return point2;  
 }  
  
 */\*\*  
 \* Получить положение 3-ой точки  
 \*  
 \** ***@return*** *point3  
 \*/* public Vector2d getPoint3() {  
 return point3;  
 }  
  
 */\*\*  
 \* Строковое представление объекта  
 \*  
 \** ***@return*** *строковое представление объекта  
 \*/* @Override  
 public String toString() {  
 return "Triangle{" +  
 "point1=" + point1 + ", point2=" + point2 + ", point3=" + point3 +  
 '}';  
 }  
  
 */\*\*  
 \* Проверка двух объектов на равенство  
 \*  
 \** ***@param*** *o объект, с которым сравниваем текущий  
 \** ***@return*** *флаг, равны ли два объекта  
 \*/* @Override  
 public boolean equals(Object o) {  
 // если объект сравнивается сам с собой, тогда объекты равны  
 if (this == o) return true;  
 // если в аргументе передан null или классы не совпадают, тогда объекты не равны  
 if (o == null || getClass() != o.getClass()) return false;  
 // приводим переданный в параметрах объект к текущему классу  
 Triangle triangle = (Triangle) o;  
 return (point1 == triangle.point1) && (point2 == triangle.point2) && (point3 == triangle.point3);  
 }  
  
 */\*\*  
 \* Получить хэш-код объекта  
 \*  
 \** ***@return*** *хэш-код объекта  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(point1, point2, point3);  
 }  
}

# Ray.java

package app;  
  
import Misc.Misc;  
import Misc.Vector2d;  
import Misc.Vector2i;  
import com.fasterxml.jackson.annotation.JsonCreator;  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import com.fasterxml.jackson.annotation.JsonProperty;  
  
import java.util.Objects;  
  
public class Ray {  
 */\*\*  
 \* Координаты 1 точки  
 \*/* public final Vector2d pos1;  
 */\*\*  
 \* Координаты 2 точки  
 \*/* public final Vector2d pos2;  
  
 */\*\*  
 \* Конструктор точки  
 \*  
 \** ***@param*** *pos1 положение точки 1  
 \** ***@param*** *pos2 положение точки 2  
 \*/* @JsonCreator  
 public Ray(@JsonProperty("pos1") Vector2d pos1, @JsonProperty("pos2") Vector2d pos2) {  
 this.pos1 = pos1;  
 this.pos2 = pos2;  
 }  
  
 */\*\*  
 \* Получить положение  
 \* (нужен для json)  
 \*  
 \** ***@return*** *положение  
 \*/*// public Vector2d getPos() {  
// return pos;  
// }  
  
 */\*\*  
 \* Получить радуис  
 \*  
 \** ***@return*** *радиус  
 \*/*// public int getRadius() {  
// return radius;  
// }  
  
  
 */\*\*  
 \* Строковое представление объекта  
 \*  
 \** ***@return*** *строковое представление объекта  
 \*/* @Override  
 public String toString() {  
 return "Point{" +  
 "pos1=" + pos1 +  
 ", pos2=" + pos2 +  
 '}';  
 }  
  
 */\*\*  
 \* Проверка двух объектов на равенство  
 \*  
 \** ***@param*** *o объект, с которым сравниваем текущий  
 \** ***@return*** *флаг, равны ли два объекта  
 \*/* @Override  
 public boolean equals(Object o) {  
 // если объект сравнивается сам с собой, тогда объекты равны  
 if (this == o) return true;  
 // если в аргументе передан null или классы не совпадают, тогда объекты не равны  
 if (o == null || getClass() != o.getClass()) return false;  
 // приводим переданный в параметрах объект к текущему классу  
 app.Ray ray = (app.Ray) o;  
 return Objects.*equals*(pos1, ray.pos1) && Objects.*equals*(pos2, ray.pos2);  
 }  
  
 */\*\*  
 \* Получить хэш-код объекта  
 \*  
 \** ***@return*** *хэш-код объекта  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*( pos1, pos2);  
 }  
}

# Приложение Б. UnitTest.java

import app.Point;  
import app.Ray;  
import app.Task;  
import app.Triangle;  
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*** *points список точек  
 \** ***@param*** *rays список лучей  
 \** ***@param*** *triangles список треугольников  
 \*/* private static void test(ArrayList<Point> points, ArrayList<Ray> rays, ArrayList<Triangle> triangles, int num) {  
 Task task = new Task(new CoordinateSystem2d(10, 10, 20, 20), triangles, rays, points);  
 int max = -1;  
 for (Triangle t : triangles) {  
 for (Ray r : rays) {  
 int c = 0;  
 for (Point p : points) {  
 if (task.check\_tri(t, p) && task.check\_ray(r, p, task.lastWindowCS))  
 c++;  
  
 }  
 if (c > max) {  
 max = c;  
 }  
 }  
 }  
 assert max == num;  
 }  
  
  
 */\*\*  
 \* Первый тест  
 \*/* @Test  
 public void test1() {  
 ArrayList<Point> points = new ArrayList<>();  
 ArrayList<Ray> rays = new ArrayList<>();  
 ArrayList<Triangle> triangles = new ArrayList<>();  
  
 points.add(new Point(new Vector2d(1, 1)));  
 points.add(new Point(new Vector2d(1, 2)));  
 points.add(new Point(new Vector2d(0, 0)));  
 rays.add(new Ray(new Vector2d(-1, -4), new Vector2d(3, -4)));  
 triangles.add(new Triangle(new Vector2d(0,0), new Vector2d(3, 0), new Vector2d(0, 4)));  
 triangles.add(new Triangle(new Vector2d(-8,0), new Vector2d(-5, 0), new Vector2d(-3, 6)));  
  
  
  
 *test*(points, rays, triangles, 3);  
 }  
  
 */\*\*  
 \* Второй тест  
 \*/* @Test  
 public void test2() {  
 ArrayList<Point> points = new ArrayList<>();  
 ArrayList<Ray> rays = new ArrayList<>();  
 ArrayList<Triangle> triangles = new ArrayList<>();  
  
 points.add(new Point(new Vector2d(1, 1)));  
 points.add(new Point(new Vector2d(5, 2)));  
 points.add(new Point(new Vector2d(-3, -1)));  
 points.add(new Point(new Vector2d(0, 0)));  
 rays.add(new Ray(new Vector2d(5, 4), new Vector2d(-5, -2)));  
 triangles.add(new Triangle(new Vector2d(-4,-2), new Vector2d(-4, 7), new Vector2d(8, 1)));  
 triangles.add(new Triangle(new Vector2d(-8,0), new Vector2d(-5, 0), new Vector2d(-3, 6)));  
  
  
  
 *test*(points, rays, triangles, 4);  
 }  
  
 */\*\*  
 \* Второй тест  
 \*/* @Test  
 public void test3() {  
 ArrayList<Point> points = new ArrayList<>();  
 ArrayList<Ray> rays = new ArrayList<>();  
 ArrayList<Triangle> triangles = new ArrayList<>();  
  
 points.add(new Point(new Vector2d(1, 1)));  
 points.add(new Point(new Vector2d(5, 2)));  
 points.add(new Point(new Vector2d(-3, -1)));  
 points.add(new Point(new Vector2d(0, 0)));  
 points.add(new Point(new Vector2d(1, -2)));  
 points.add(new Point(new Vector2d(3, -6)));  
 rays.add(new Ray(new Vector2d(5, -4), new Vector2d(-5, -2)));  
 rays.add(new Ray(new Vector2d(2, -4), new Vector2d(3, -4)));  
 rays.add(new Ray(new Vector2d(4, -5), new Vector2d(2, -6)));  
 triangles.add(new Triangle(new Vector2d(-4,6), new Vector2d(-2, 5), new Vector2d(3, 3)));  
 triangles.add(new Triangle(new Vector2d(-8,5), new Vector2d(-5, -3), new Vector2d(-3, 6)));  
  
  
  
 *test*(points, rays, triangles, 0);  
 }  
  
}