Комитет по образованию г. Санкт-Петербург

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ПРЕЗИДЕНТСКИЙ ФИЗИКО-МАТЕМАТИЧЕСКИЙ
ЛИЦЕЙ №239

Отчет о практике
«Создание графических приложений на языке Java»

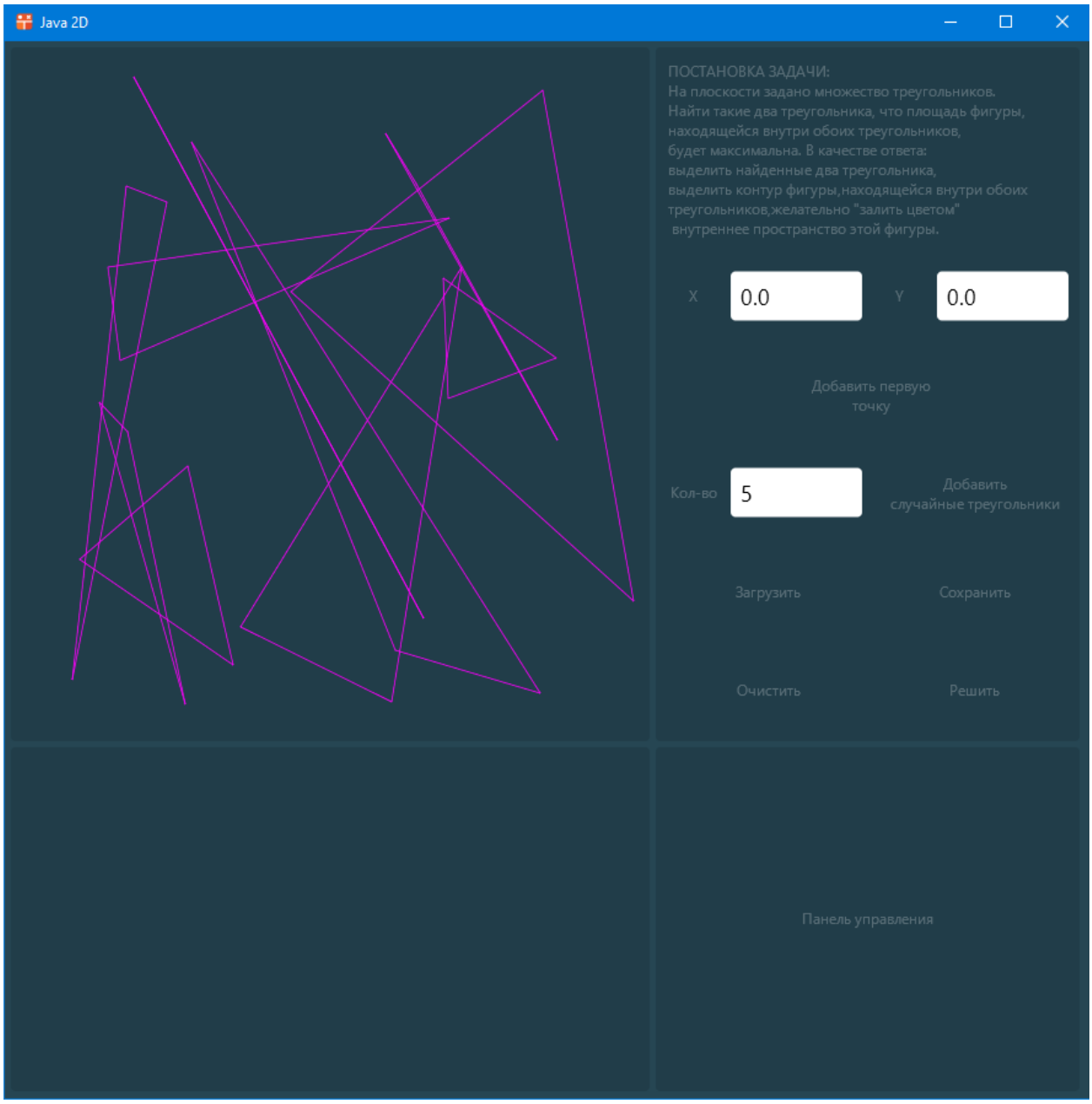
Учащийся 10-3 класса
Куликов Д.А.

Преподаватель:
Клюнин А.О.

Санкт-Петербург – 2023 год

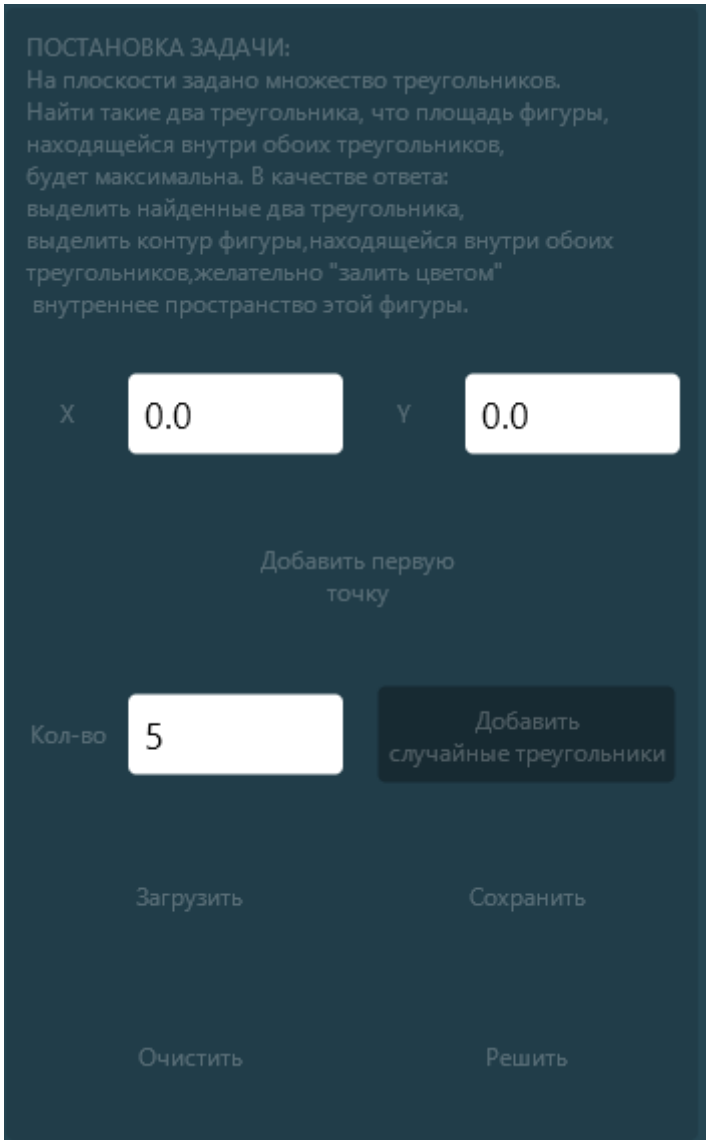
1. Постановка задачи

На плоскости задано множество треугольников. Найти такие два треугольника, что площадь фигуры, находящейся внутри обоих треугольников, будет максимальна.



2. Элементы управления

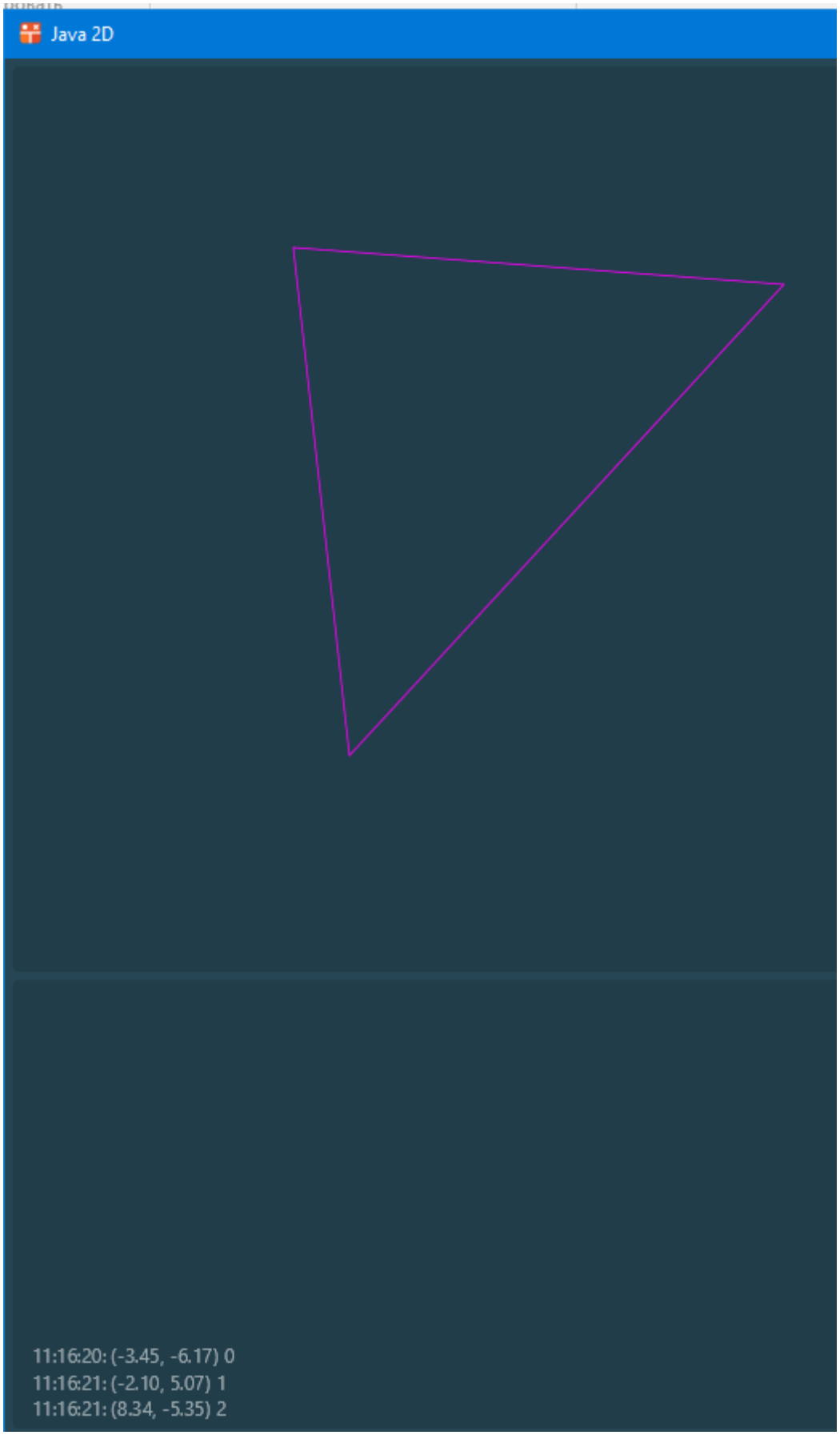
В рамках данной задачи необходимо было реализовать следующие элементы управления:



Для добавления точки по координатам было создано два поля ввода: «X» и «Y». Точки треугольника добавляются по порядку через поле-“Добавить первую/вторую/третью точку”

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

Также программа позволяет добавлять точки треугольника с помощью клика мышью по области рисования (Добавление точек мышью и кнопками не сочетаются).



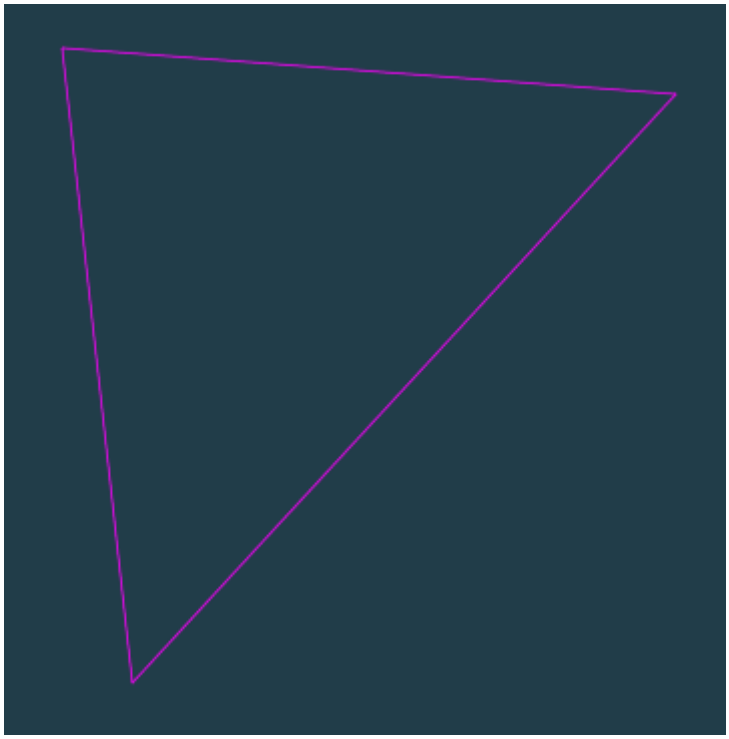
3. Структуры данных

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

В него были добавлены поля **PosA, PosB**, **PosC** соответствующие положениям точки в пространстве задачи.

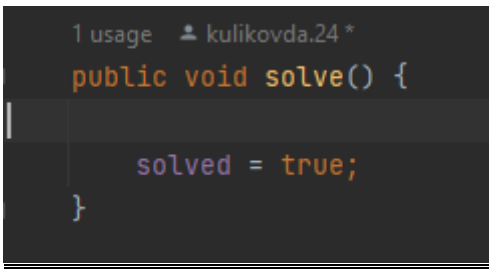
4. Рисование

Чтобы нарисовать треугольник, использовалась команда рисования линий **canvas.drawLine().**



5. Решение задачи

Для решения поставленной задачи в классе **Task** не было разработано методов**.**



Для решения задачи можно использовать метод Монте-Карло, его идея заключается в том, что мы генерируем множество случайных точек, а потом считаем, какая часть из них принадлежит области пересечения треугольников, тогда получается, что если принять площадь экрана за S, то площадь пересечения треугольников S_f будет равна

Где C-общее число случайных точек, а C_f-кол-во точек внутри фигуры.

6. Проверка

Для проверки принадлежности точки треугольнику были разработаны unit-тесты. Их листинг приведён в приложении Б.

Тест 1

Первая точка треугольника: $\{0; 2\}$

Вторая точка треугольника: : $\{2; 2\}$

Третья точка треугольника: : $\{1; -2\}$

Проверяемая точка: : $\{0; 0\}$

Ответ: не содержится

Тест 2

Первая точка треугольника: $\{0; 2\}$

Вторая точка треугольника: : $\{7; 5\}$

Третья точка треугольника: : $\{-3.5; -8\}$

Проверяемая точка: : $\{10; 0\}$

Ответ: не содержится

Тест 3

Первая точка треугольника: $\{-10; -10\}$

Вторая точка треугольника: : $\{0; 15\}$

Третья точка треугольника: : $\{10; -7.5\}$

Проверяемая точка: : $\{2; 3\}$

Ответ: содержится

Тест 4

Первая точка треугольника: $\{7; 4\}$

Вторая точка треугольника: : $\{-10; 6\}$

Третья точка треугольника: : $\{1; -2\}$

Проверяемая точка: : $\{3; 4\}$

Ответ: содержится

7. Заключение

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

Приложение А. Point.java

```
package app;

import io.github.humbleui.skija.Canvas;
import io.github.humbleui.skija.Paint;
import misc.*;

import java.util.Objects;

/**
 * Класс точки
 */
public class Triangle {

    /**
     * Координаты точки A
     */
    public final Vector2d posA;
    /**
     * Координаты точки B
     */
    public final Vector2d posB;
    /**
     * Координаты точки C
     */
    public final Vector2d posC;

    /**
     * Конструктор точки
     *
     * @param posA положение точки A
     * @param posB положение точки B
     * @param posC положение точки C
     */
    public Triangle(Vector2d posA, Vector2d posB, Vector2d posC) {
        this.posA = posA;
        this.posB = posB;
        this.posC = posC;
    }

    /**
     * Провекра, содержится ли точка v внутри треугольника
     *
     * @param v - координаты точки
     * @return - флаг, содержится ли точка v внутри треугольника
     */
    public boolean contains(Vector2d v) {
        double x0 = v.x;
        double y0 = v.y;

        double x1 = posA.x;
        double y1 = posA.y;
        double x2 = posB.x;
        double y2 = posB.y;
        double x3 = posC.x;
        double y3 = posC.y;

        double a = (x1 - x0) * (y2 - y1) - (x2 - x1) * (y1 - y0);
        double b = (x2 - x0) * (y3 - y2) - (x3 - x2) * (y2 - y0);
        double c = (x3 - x0) * (y1 - y3) - (x1 - x3) * (y3 - y0);
```

```
        if((a>0&&b>0&&c>0)||(a<0&&b<0&&c<0))
        {
            return true;
        }
        else
        {
            return false;
        }
    }


    public void paint(Canvas canvas, CoordinateSystem2i windowCS,
CoordinateSystem2d ownCS) {
        // создаём перо
        try (Paint p = new Paint()) {
            p.setColor(getColor());
            // вершины треугольника
            Vector2i pointA = windowCS.getCoords(posA, ownCS);
            Vector2i pointB = windowCS.getCoords(posB, ownCS);
            Vector2i pointC = windowCS.getCoords(posC, ownCS);
            // рисуем его стороны
            canvas.drawLine(pointA.x, pointA.y, pointB.x, pointB.y, p);
            canvas.drawLine(pointB.x, pointB.y, pointC.x, pointC.y, p);
            canvas.drawLine(pointC.x, pointC.y, pointA.x, pointA.y, p);
        }
    }

    /**
     * Получить цвет точки по её множеству
     *
     * @return цвет точки
     */
    public int getColor() {
        return Misc.getColor(0xCC, 0xFF, 0x00, 0xFF);
    }


    /**
     * Получить положение A
     * (нужен для json)
     *
     * @return положение
     */
    public Vector2d getPosA() {
        return posA;
    }
    /**
     * Получить положение B
     * (нужен для json)
     *
     * @return положение
     */
    public Vector2d getPosB() {
        return posB;
    }
    /**
     * Получить положение C
     * (нужен для json)
     *
     * @return положение
     */
    public Vector2d getPosC() {
        return posC;
```

```
    }

    /**
     * Строковое представление объекта
     *
     * @return строковое представление объекта
     */
    @Override
    public String toString() {
        return "Point{" +
                ", pos=" + posA +
                ", pos=" + posB +
                ", pos=" + posC +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Triangle triangle = (Triangle) o;
        return Objects.equals(posA, triangle.posA) && Objects.equals(posB,
triangle.posB) && Objects.equals(posC, triangle.posC);
    }

    @Override
    public int hashCode() {
        return Objects.hash(posA, posB, posC);
    }

}
```

Приложение Б. UnitTest.java

```
import app.Triangle;
import misc.Vector2d;
import org.junit.Test;

public class Tests {

    @Test
    public void test1() {
        Triangle t = new Triangle(
                new Vector2d(0, 2),
                new Vector2d(2, 2),
                new Vector2d(1, -2)
        );

        assert !t.contains(new Vector2d(0, 0));
    }

    @Test
    public void test2() {
        Triangle t = new Triangle(
                new Vector2d(0, 2),
                new Vector2d(7, 5),
                new Vector2d(-3.5, -8)
        );

        assert !t.contains(new Vector2d(10, 0));
    }



    @Test
    public void test3() {
        Triangle t = new Triangle(
                new Vector2d(-10, -10),
                new Vector2d(0, 15),
                new Vector2d(10, -7.5)
        );

        assert t.contains(new Vector2d(2, 3));
    }

    @Test
    public void test4() {
        Triangle t = new Triangle(
                new Vector2d(7, 4),
                new Vector2d(-10, 6),
                new Vector2d(1, -2)
        );

        assert t.contains(new Vector2d(3, 4));
    }
}
```