Комитет по образованию г. Санкт-Петербурга

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ПРЕЗИДЕНТСКИЙ ФИЗИКО-МАТЕМАТИЧЕСКИЙ
ЛИЦЕЙ № 239

Отчет о практике
“Создание графических приложений на языке Java”

Учащийся 10-3 класса
Кощеев Г.О.

Преподаватель :
Клюнин А.О.

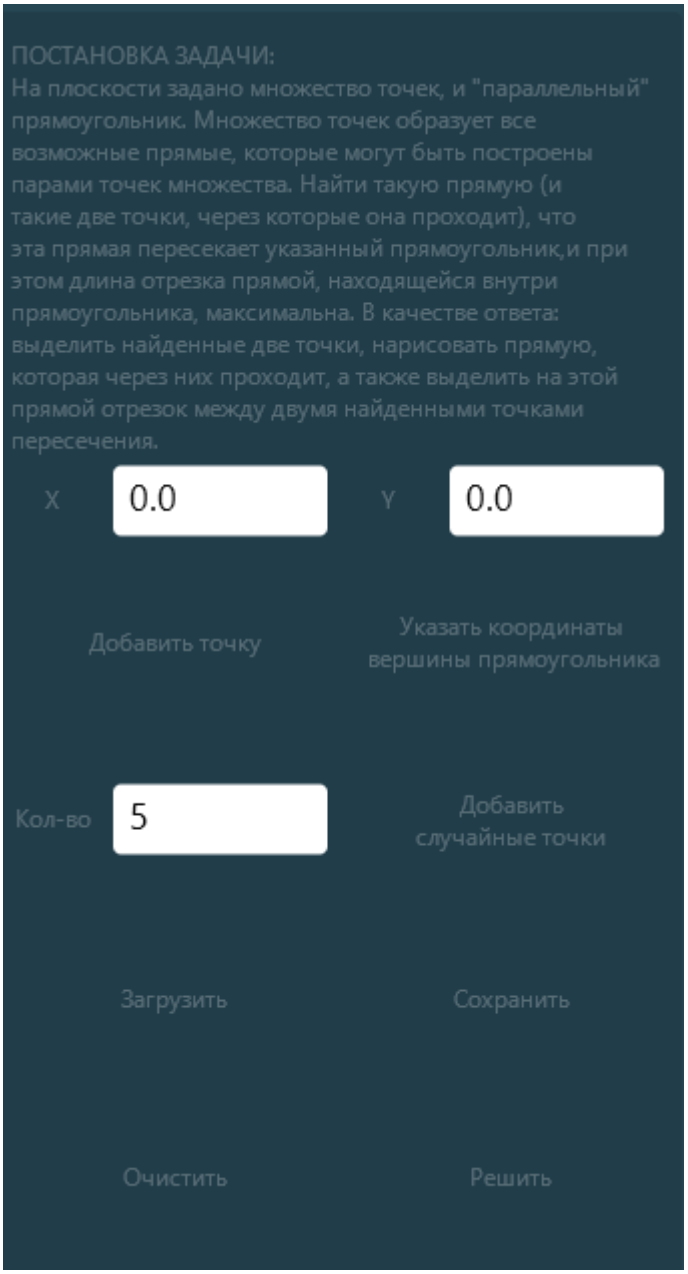
Санкт-Петербург - 2022 год

Постановка задачи.

На плоскости задано множество точек и “параллельный прямоугольник”. Множество точек образует все возможные прямые, которые могут быть построены парами точек множества. Найти такую прямую(и такие две точки множества, через которые она проходит), что эта прямая пересекает указанный прямоугольник, и при этом длина отрезка прямой, находящейся внутри прямоугольника, максимальна. В качестве ответа : выделить найденные две точки, нарисовать прямую, которая через них проходит, а также выделить на этой прямой отрезок между двумя найденными точками пересечения.

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». Первая кнопка добавляет точку во множество точек, вторая же делает ее одной из вершин прямоугольника. Прямоугольник в этой задаче задается двумя противоположными вершинами. Также программа позволяет добавлять точки во множество точек с помощью клика мыши.

Структуры данных.

Для хранения точек, через которые мы проводим прямые, был создан массив **ArrayList<Point> points**. Вершины прямоугольника хранятся в массиве **ArrayList<Point> tops.** Нашу задачу мы решаем в вещественной системе координат **ownCS.** Для решения задачи были также созданы 2 массива : **ArrayList<Point> sPoints**(в нем хранятся точки, через которую мы проведем прямую, являющуюся решением задачи) и **ArrayList<Point> crossPoints**(здесь хранятся точки пересечения нужной нам прямой и прямоугольника).

```
/**
 * Вещественная система координат задачи
 */
private final CoordinateSystem2d ownCS;
/**
 * Список точек, через которые мы проводим прямые
 */
private final ArrayList<Point> points;
/**
 * Список вершин прямоугольника
 */
private final ArrayList<Point> tops;
/**
 * Список точек, являющихся решением нашей задачи
 */
private final ArrayList<Point> sPoints;
/**
 * Список точек пересечения прямой и прямоугольника
 */
private final ArrayList<Vector2d> crossPoints;
```

Рисование.

Для рисования точек использовалась команда **canvas.drawRect().** Прямоугольник же мы рисовали, используя команду рисования линии **canvas.drawLine()** 4 раза.

Решение задачи.

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

```
/**
 * Решить задачу
 */
public void solve() {
    if(tops.isEmpty() || tops.size() == 1){
        PanelLog.error( text: "Задача не может быть решена, потому что прямоугольник не был задан");
        rectangle = false;
    }else if(points.size() >= 2){
        rectangle = true;
        // k - коэффициент наклона прямой
        double k;
        // c - свободный член
        double c;
        // создаем переменную, в которую мы будем записывать длины отрезков, расположенных внутри прямоугольника
        double lengthMax = -1;
        double length = -1;

        double _top = max(tops.get(0).getPos().y, tops.get(1).getPos().y);
        double _bottom = min(tops.get(0).getPos().y, tops.get(1).getPos().y);
        double _right = max(tops.get(0).getPos().x, tops.get(1).getPos().x);
        double _left = min(tops.get(0).getPos().x, tops.get(1).getPos().x);

        Point aMax = null;
        Point bMax = null;
        Vector2d crossA = null;
        Vector2d crossB = null;
        Vector2d crossAmax = null;
        Vector2d crossBmax = null;
```

```
// перебираем пары точек
for (int i = 0; i < points.size(); i++) {
    for (int j = i + 1; j < points.size(); j++) {
        // сохраняем точки
        Point a = points.get(i);
        Point b = points.get(j);

        // случай, когда прямая параллельна ординате
        if (a.getPos().x == b.getPos().x) {
            // проверяем, пересекает ли прямая прямоугольник
            if (a.getPos().x >= _left && a.getPos().x <= _right) {
                length = _top - _bottom;
                crossA = new Vector2d(a.getPos().x, _bottom);
                crossB = new Vector2d(a.getPos().x, _top);
            }
        } else {
            k = (a.getPos().y - b.getPos().y) / (a.getPos().x - b.getPos().x);
            // ищем коэффициент наклона прямой и свободный член
            c = a.getPos().y - a.getPos().x * k;

            // ищем точку пересечения прямой и нашего прямоугольника

            // случай, когда прямая параллельна абсциссе
            if (k == 0) {
                if (a.getPos().y >= _bottom && a.getPos().y <= _top) {
                    length = _right - _left;
                    crossA = new Vector2d(_left, a.getPos().y);
                    crossB = new Vector2d(_right, a.getPos().y);
                }
```

```
} else {
    double yLeft = k * _left + c;
    double yRight = k * _right + c;
    double xTop = (_top - c) / k;
    double xBottom = (_bottom - c) / k;

    if (yLeft <= _top && yLeft >= _bottom) {
        Vector2d v = new Vector2d(_left, yLeft);
        crossA = v;
        if (yRight <= _top && yRight >= _bottom) {
            Vector2d v2 = new Vector2d(_right, yRight);
            crossB = v2;
            length = Math.sqrt((v2.x - v.x) * (v2.x - v.x) + (v2.y - v.y) * (v2.y - v.y));
        } else if (xBottom >= _left && xBottom <= _right) {
            Vector2d v2 = new Vector2d(xBottom, _bottom);
            crossB = v2;
            length = Math.sqrt((v2.x - v.x) * (v2.x - v.x) + (v2.y - v.y) * (v2.y - v.y));
        } else if (xTop >= _left && xTop <= _right) {
            Vector2d v2 = new Vector2d(xTop, _top);
            crossB = v2;
            length = Math.sqrt((v2.x - v.x) * (v2.x - v.x) + (v2.y - v.y) * (v2.y - v.y));
        }
    } else if (yRight <= _top && yRight >= _bottom) {
        Vector2d v = new Vector2d(_right, yRight);
        crossA = v;
        if (xBottom >= _left && xBottom <= _right) {
            Vector2d v2 = new Vector2d(xBottom, _bottom);
            crossB = v2;
            length = Math.sqrt((v2.x - v.x) * (v2.x - v.x) + (v2.y - v.y) * (v2.y - v.y));
```

```
                    } else if (xTop >= _left && xTop <= _right) {
                        Vector2d v2 = new Vector2d(xTop, _top);
                        crossB = v2;
                        length = Math.sqrt((v2.x - v.x) * (v2.x - v.x) + (v2.y - v.y) * (v2.y - v.y));
                    }
                } else if (xBottom >= _left && xBottom <= _right) {
                    Vector2d v = new Vector2d(xBottom, _bottom);
                    Vector2d v2 = new Vector2d(xTop, _top);
                    crossA = v;
                    crossB = v2;
                    length = Math.sqrt((v2.x - v.x) * (v2.x - v.x) + (v2.y - v.y) * (v2.y - v.y));
                }
            }
            if (length > lengthMax) {
                lengthMax = length;
                aMax = a;
                bMax = b;
                crossAmax = crossA;
                crossBmax = crossB;
            }
        }
    }
    // задача решена
    solved = true;

    if (aMax != null && bMax != null) {
        sPoints.add(aMax);
        sPoints.add(bMax);
        crossPoints.add(crossAmax);
        crossPoints.add(crossBmax);
```

Проверка.

Для проверки правильности решённой задачи были разработаны unit-тесты.

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

    /**
     * Тест
     *
     * @param points        список точек
     * @param tops          вершины прямоугольника
     */
    private static void test(CoordinateSystem2d ownCS, ArrayList<Point> points, ArrayList<Point> tops,
                             ArrayList<Point> sPoints, ArrayList<Vector2d> crossPoints) {
        Task task = new Task(ownCS, points, tops);
        task.solve();

        // проверяем, правильно ли мы нашли точки, через которые будем проводить прямую
        for(Point p : sPoints){
            assert task.getsPoints().contains(p);
        }

        // проверяем, правильно ли мы нашли точки пересечения прямой и прямоугольника
        for(Vector2d p : crossPoints){
            assert task.getCrossPoints().contains(p);
        }

    }
```

```java
/**
 * Первый тест
 */
@Test
public void test1() {
    ArrayList<Point> points = new ArrayList<>();
    ArrayList<Point> tops = new ArrayList<>();
    CoordinateSystem2d ownCS = new CoordinateSystem2d( minX: -10,  minY: -10,  sizeX: 20,  sizeY: 20);
    ArrayList<Point> sPoints = new ArrayList<>();
    ArrayList<Vector2d> crossPoints = new ArrayList<>();

    points.add(new Point(new Vector2d( x: 1,  y: 1)));
    points.add(new Point(new Vector2d( x: 9,  y: 9)));
    points.add(new Point(new Vector2d( x: 6,  y: 1)));
    points.add(new Point(new Vector2d( x: 6,  y: 9)));

    tops.add(new Point(new Vector2d( x: 9,  y: 8)));
    tops.add(new Point(new Vector2d( x: 2,  y: 3)));

    sPoints.add(new Point(new Vector2d( x: 1, y: 1)));
    sPoints.add(new Point(new Vector2d( x: 9, y: 9)));

    crossPoints.add(new Vector2d( x: 3, y: 3));
    crossPoints.add(new Vector2d( x: 8, y: 8));

    test(ownCS, points, tops, sPoints, crossPoints);
}
```

```java
/**
 * Второй тест
 */
@Test
public void test2() {
    ArrayList<Point> points = new ArrayList<>();
    ArrayList<Point> tops = new ArrayList<>();
    CoordinateSystem2d ownCS = new CoordinateSystem2d( minX: -10,  minY: -10,  sizeX: 20,  sizeY: 20);
    ArrayList<Point> sPoints = new ArrayList<>();
    ArrayList<Vector2d> crossPoints = new ArrayList<>();

    points.add(new Point(new Vector2d( x: -5,  y: 1)));
    points.add(new Point(new Vector2d( x: -5,  y: -8)));

    tops.add(new Point(new Vector2d( x: 1,  y: 1)));
    tops.add(new Point(new Vector2d( x: 9,  y: 9)));

    test(ownCS, points, tops, sPoints, crossPoints);
}
```

```java
/**
 * Третий тест
 */
@Test
public void test3() {
    ArrayList<Point> points = new ArrayList<>();
    ArrayList<Point> tops = new ArrayList<>();
    CoordinateSystem2d ownCS = new CoordinateSystem2d( minX: -10,  minY: -10,  sizeX: 20,  sizeY: 20);
    ArrayList<Point> sPoints = new ArrayList<>();
    ArrayList<Vector2d> crossPoints = new ArrayList<>();

    points.add(new Point(new Vector2d( x: 6,  y: 9)));
    points.add(new Point(new Vector2d( x: 0,  y: -9)));

    tops.add(new Point(new Vector2d( x: 1,  y: 1)));
    tops.add(new Point(new Vector2d( x: 6,  y: 7)));

    test(ownCS, points, tops, sPoints, crossPoints);
}
```

Заключение.

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