

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

**Государственное бюджетное образовательное учреждение**

**Президентский физико-математический лицей №239**

**Отчет о практике**

**«Нахождение наибольшего отрезка внутри  
треугольника»**

Учащаяся 10-2 класса

Калина Е.А.

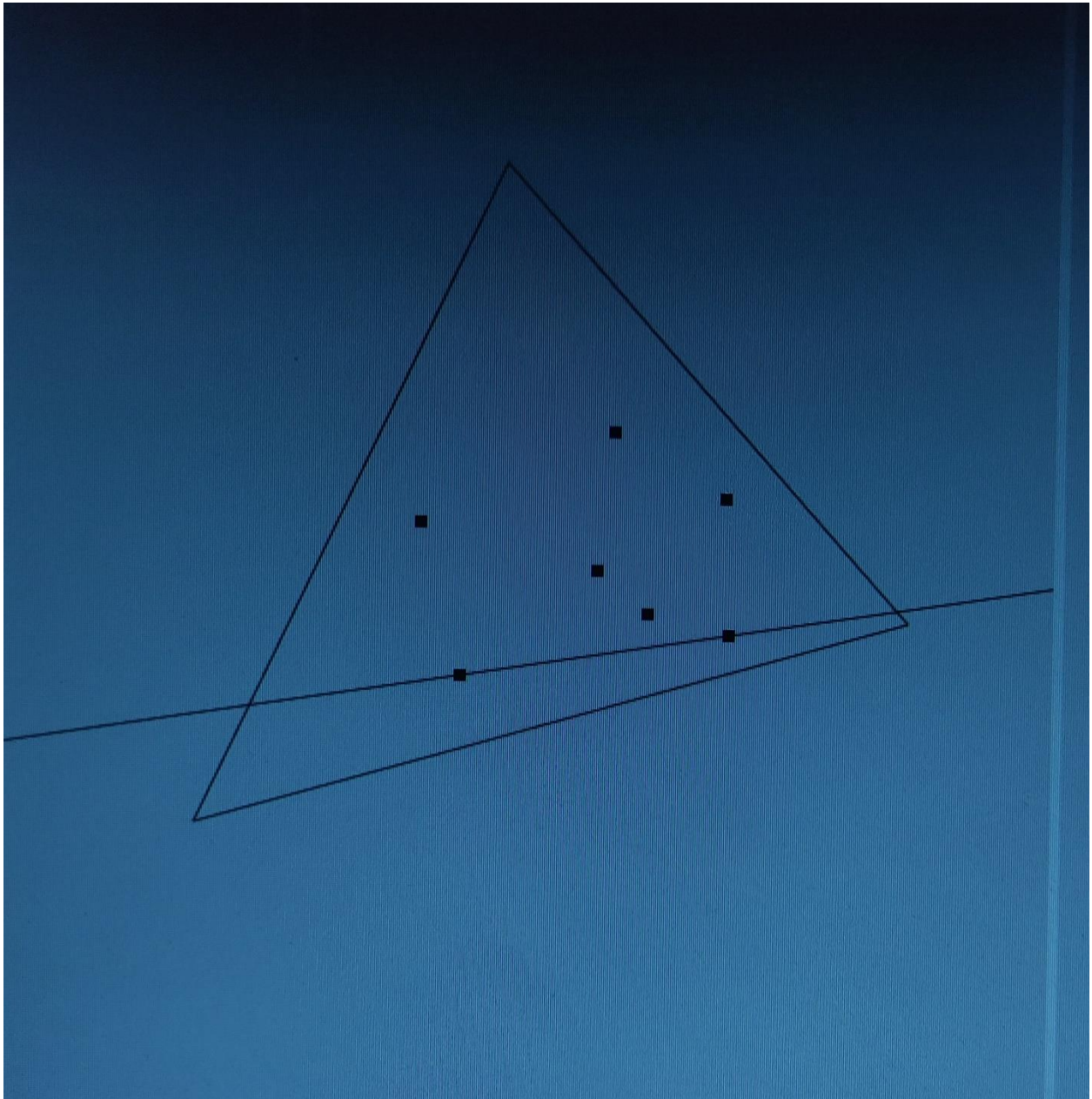
Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2023 год

## 1. Постановка задачи

На плоскости задан треугольник и еще множество точек. Необходимо найти такие две точки множества, что прямая, проходящая через эти две точки, пересекает треугольник, и при этом отрезок этой прямой, оказавшейся внутри треугольника, оказывается наибольшей длины. В качестве ответа хотелось бы видеть эти две точки, прямую, через них проходящую, и этот отрезок.



## 2. Элементы управления

В рамках данной задачи необходимо было реализовать следующие элементы управления:

The image shows a screenshot of a software interface with a dark blue background. It contains several input fields and buttons. At the top, there are two input fields labeled 'X' and 'Y', both containing the value '0.0'. Below them is a button labeled 'Добавить точку'. In the middle, there is an input field labeled 'Кол-во' containing the value '5', followed by a button labeled 'Добавить случайные точки'. Below these are two more input fields labeled 'X' and 'Y', both containing '0.0', with a button labeled 'Точка треугольника' below them. At the bottom, there are four buttons arranged in two rows: 'Загрузить' and 'Сохранить' in the top row, and 'Очистить' and 'Решить' in the bottom row.

Для добавления точки по координатам было создано два поля ввода: «X» и «Y», а также кнопка «Добавить точку».

Имеется возможность добавить точки со случайными координатами. Для этого в поле ввода «Кол-во» надо указать их число и нажать кнопку «Добавить случайные точки».

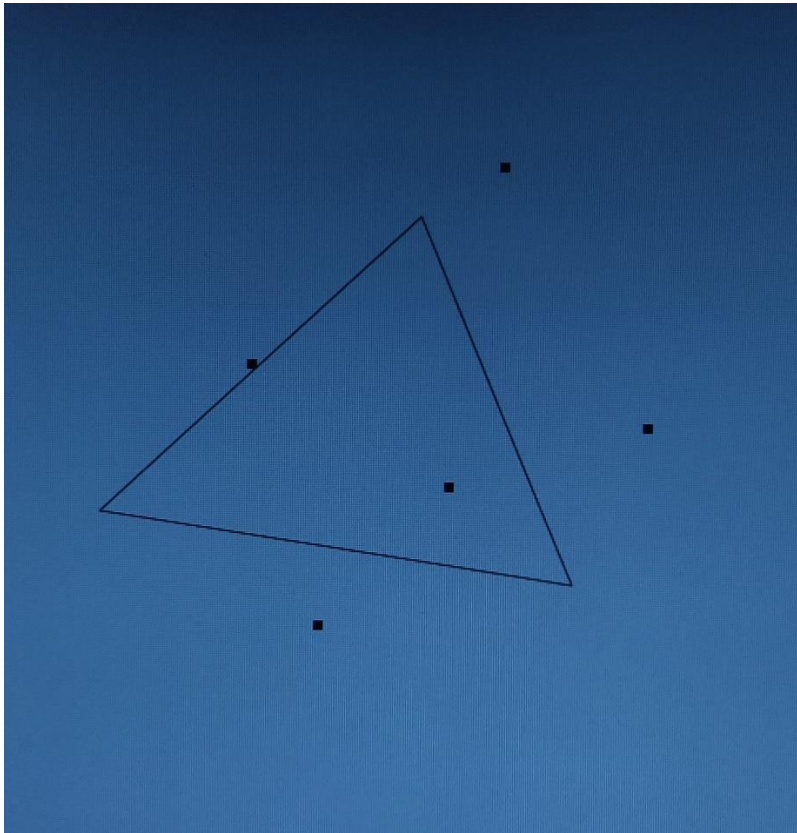
Для добавления точек треугольника по координатам было создано два поля ввода: «X» и «Y», а также кнопка «Точка треугольника».

Имеется возможность задать треугольник со случайными координатами вершин. Для этого надо нажать кнопку «Случайный треугольник».

Также можно добавлять точки и треугольник кликом мыши по области рисования.

Левой кнопкой мыши осуществляется добавление точки.

Тремя кликами правой кнопкой мыши можно добавить треугольник.



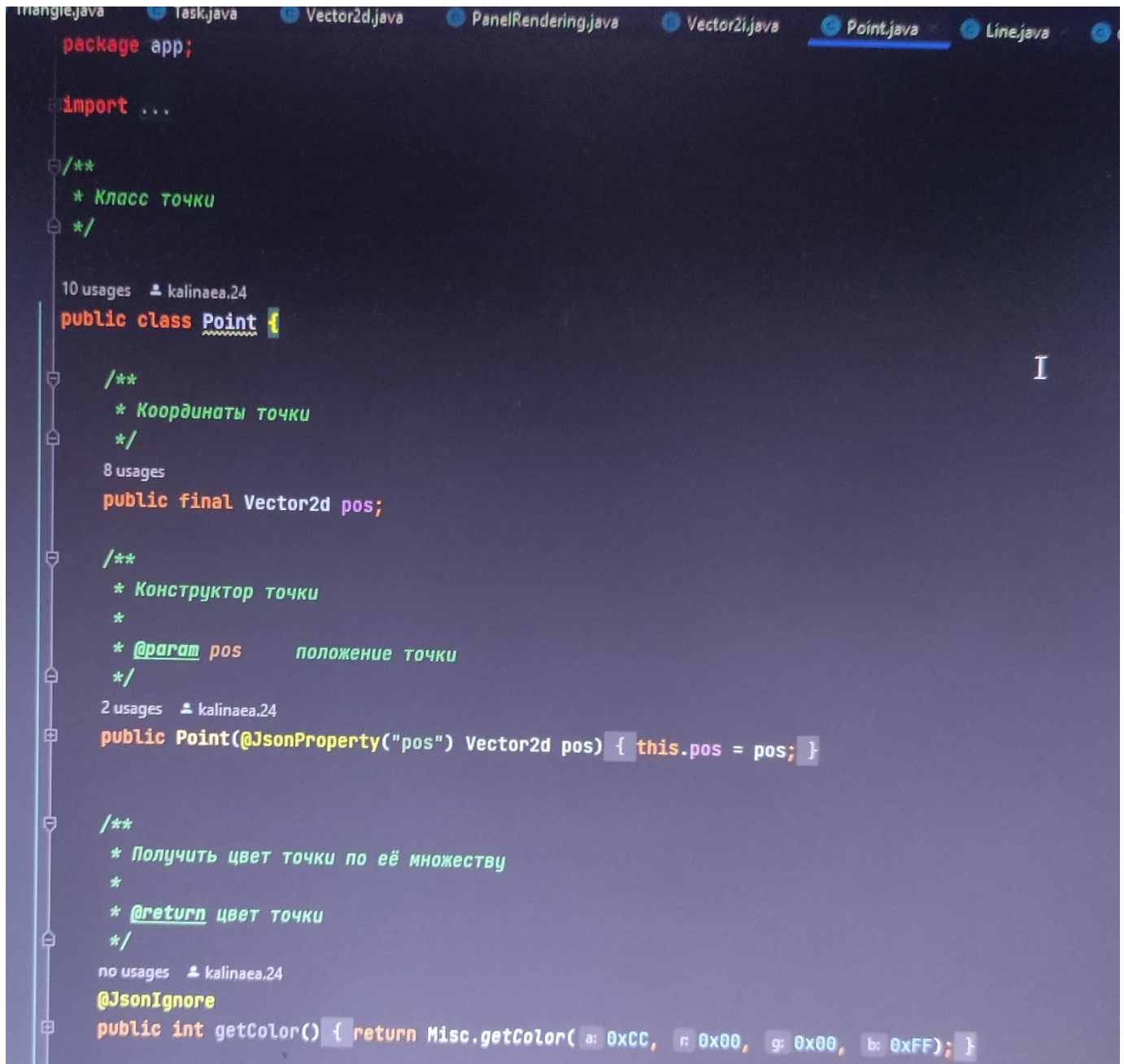
```
1 usage  ± kalinaea.24 +1
public void click(Vector2i pos, MouseButton mouseButton) {
    if (lastWindowCS == null) return;
    // получаем положение точки на экране
    Vector2d taskPos = ownCS.getCoords(pos, lastWindowCS);
    Point pointByMouse = new Point(taskPos);
    // если левая кнопка мыши, добавляем точку
    if (mouseButton.equals(MouseButton.PRIMARY)) points.add(pointByMouse);
    // если правая кнопка мыши, добавляем треугольник по точкам на экране
    else if (mouseButton.equals(MouseButton.SECONDARY)) {
        // получаем положение 2-й точки на экране
        Vector2d taskPos1 = ownCS.getCoords(pos, lastWindowCS);
        if (posA == null) {
            posA = taskPos1;
        } else if (posB == null) {
            posB = taskPos1;
        } else {
            triangle = new Triangle(posA, posB, taskPos1);
            posA = null;
            posB = null;
        }
    }
}
```



### 3. Структуры данных

Для решения задачи были разработаны классы Point.java, Line.java, Triangle.java.

- Point.java



```
package app;

import ...

/**
 * Класс точки
 */
10 usages  kalinaea,24
public class Point {

    /**
     * Координаты точки
     */
    8 usages
    public final Vector2d pos;

    /**
     * Конструктор точки
     *
     * @param pos    положение точки
     */
    2 usages  kalinaea,24
    public Point(@JsonProperty("pos") Vector2d pos) { this.pos = pos; }

    /**
     * Получить цвет точки по её множеству
     *
     * @return цвет точки
     */
    no usages  kalinaea,24
    @JsonIgnore
    public int getColor() { return Misc.getColor( a: 0xCC, r: 0x00, g: 0x00, b: 0xFF); }
```

```
/**
 * Получить положение
 * (нужен для json)
 *
 * @return положение
 */
```

2 usages  kalinaea.24

```
public Vector2d getPos() { return pos; }
```

```
/**
 * Строковое представление объекта
 *
 * @return строковое представление объекта
 */
```

 kalinaea.24

**@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
 */
```

kalinaea.24

**@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
```

```
 */
```

kalinaea.24

**@Override**

```
public int hashCode() { return Objects.hash(pos); }
```



- Line.java

```
package app;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonProperty;
import io.github.humblevi.skija.Canvas;
import io.github.humblevi.skija.Paint;
import lombok.Getter;
import misc.*;

import java.util.Objects;

25 usages  Лиза *
public class Line {
    /**
     * две точки прямой
     */
    @Getter
    public Vector2d pos1;
    @Getter
    public Vector2d pos2;

    /**
     * Конструктор прямой через векторы
     *
     * @param pos1
     * @param pos2
     * положение прямой
     */
    6 usages  Лиза
    public Line(@JsonProperty("pos") Vector2d pos1, Vector2d pos2) {
        this.pos1 = pos1;
        this.pos2 = pos2;
    }
}
```



```
/**
 * Получить положение
 * (нужен для json)
 *
 * @return положение
 */
```

no usages    👤 Лиза

```
public Line getLine() {
    Line line = new Line(pos1, pos2);
    return line;
}
```

```
/**
 * Получить цвет прямой
 *
 * @return цвет прямой
 */
```

no usages    👤 Лиза

```
@JsonIgnore
public int getColor() { return Misc.getColor( a: 0xCC, r: 0x00, g: 0x00, b: 0xFF); }
```

```
/**
 * Строковое представление объекта
 *
 * @return строковое представление объекта
 */
```

👤 Лиза

```
@Override
public String toString() {
    return "Line{" +
        "pos1=" + pos1 + ' ' + ", pos2=" + pos2 +
        '}';
}
```

```

/**
 * Проверка двух объектов на равенство
 *
 * @param o объект, с которым сравниваем текущий
 * @return флаг, равны ли два объекта
 */
// Лиза
@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
    Line line = (Line) o;
    return Objects.equals(line, o);
}

```

```

/**
 * Рисование прямой
 *
 * @param canvas
 * @param windowCS
 * @param ownCS
 */
// Лиза
1 usage
public void render(Canvas canvas, CoordinateSystem2i windowCS, CoordinateSystem2d ownCS) {
    try (Paint p = new Paint()) {
        // опорные точки прямой
        Vector2i pointA = windowCS.getCoords(pos1, ownCS);
        Vector2i pointB = windowCS.getCoords(pos2, ownCS);

        // вектор, ведущий из точки A в точку B
        Vector2i delta = Vector2i.subtract(pointA, pointB);

        // получаем максимальную длину отрезка на экране, как длину диагонали экрана
        int maxDistance = (int) windowCS.getSize().length();

        // получаем новые точки для рисования, которые гарантируют, что линия
        // будет нарисована до границ экрана
        Vector2i renderPointA = Vector2i.sum(pointA, Vector2i.mult(delta, maxDistance));
        Vector2i renderPointB = Vector2i.sum(pointA, Vector2i.mult(delta, -maxDistance));

        // рисуем линию
        canvas.drawLine(renderPointA.x, renderPointA.y, renderPointB.x, renderPointB.y, p);
    }
}

/**
 * Получить хэш-код объекта
 *
 * @return хэш-код объекта
 */
// Лиза
@Override
public int hashCode() { return Objects.hash(pos1, pos2); }
}

```



- Triangle.java

```
package app;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonProperty;
import io.github.humbleui.skija.Canvas;
import io.github.humbleui.skija.Paint;
import lombok.Getter;
import misc.*;

import java.util.ArrayList;
import java.util.Objects;

9 usages  👤 Лиза +1
public class Triangle {
    /**
     * Координаты вершин треугольника
     */
    @Getter
    public Vector2d pos1;
    @Getter
    public Vector2d pos2;
    @Getter
    public Vector2d pos3;

    4 usages  👤 kalinaea.24
    Triangle(@JsonProperty("pos1") Vector2d pos1,
             @JsonProperty("pos2") Vector2d pos2,
             @JsonProperty("pos3") Vector2d pos3) {
        this.pos1 = pos1;
        this.pos2 = pos2;
        this.pos3 = pos3;
    }

    👤 Лиза
    @Override
    public String toString() {
        return "Straight{" +
            ", pos1=" + pos1 + ", pos2=" + pos2 + ", pos3=" + pos3 +
            '}';
    }
}
```



👤 Лиза

**@Override**

```
public int hashCode() { return Objects.hash(pos1, pos2, pos3); }
```

/\*\*

\* Получить цвет треугольника

\*

\* @return цвет треугольника

\*/

no usages 👤 Лиза

**@JsonIgnore**

```
public int getColor() { return Misc.getColor( a: 0xCC, r: 0x00, g: 0x00, b: 0xFF); }
```

/\*\*

\* Рисование треугольника

\* @param canvas

\* @param windowCS

\* @param ownCS

\*/

1 usage 👤 kalinaea.24 +1

```
public void render(Canvas canvas, CoordinateSystem2i windowCS, CoordinateSystem2d ownCS) {  
    try (Paint p = new Paint()) {  
        // вершины треугольника  
        Vector2i pointA = windowCS.getCoords(pos1, ownCS);  
        Vector2i pointB = windowCS.getCoords(pos2, ownCS);  
        Vector2i pointC = windowCS.getCoords(pos3, 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);  
    }  
}
```

```
/**
 * Проверка двух объектов на равенство
 *
 * @param o объект, с которым сравниваем текущий
 * @return флаг, равны ли два объекта
 */
```

no usages Лиза

**@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 Objects.equals(triangle, o);
}
```

```
/**
 * getter треугольника
 * @return
 */
```

no usages Лиза

```
public Triangle getTriangle() {
    Triangle triangle = new Triangle(pos1, pos2, pos3);
    return triangle;
}
```

```
}
```

## 4. Рисование

Для рисования точки использовался метод **canvas.drawRect()**.

Для рисования треугольника использовался метод **canvas.drawLine()**. Он же применялся при рисовании прямой, однако в данном случае обеспечивалось нахождение границ отрезка за областью рисования.





## 5. Решение задачи

Для решения задачи были разработаны вспомогательные методы **get\_line\_a()**, **get\_line\_b()**, **get\_line\_c()**, **crossline()**, **crossLineSegment()**, **lenghtBiggerMax()**. Первые три возвращают коэффициенты в общем уравнении прямой. **crossLine()** находит точку пересечения двух прямых. **crossLineSegment()** проверяет, что точка пересечения прямых принадлежит данному отрезку прямой. **lenghtBiggerMax()** осуществляет изменение значений аргументов при переборе пар точек, если текущая длина отрезка внутри треугольника больше максимальной.

```
/**
 * Коэффициент a в общем уравнении прямой
 * @param line
 * @return a
 */
```

4 usages    👤 Лиза

```
public double get_line_a (Line line) {
    if (line.pos1.y == line.pos2.y) return 0;
    else return 1;
}
```

```
/**
 * Коэффициент b в общем уравнении прямой
 * @param line
 * @return b
 */
```

3 usages    👤 Лиза

```
public double get_line_b (Line line) {
    if (get_line_a(line) == 0) return 1;
    else return (line.pos1.x - line.pos2.x) / (line.pos2.y - line.pos1.y);
}
```

```
/**
 * Коэффициент c в общем уравнении прямой
 * @param line
 * @return c
 */
```

2 usages    👤 Лиза

```
public double get_line_c (Line line) {
    return (get_line_a(line) * line.pos1.x + get_line_b(line) * line.pos1.y) * -1;
}
```

```

    @param line1
    @param line2
    @return vector
    */
    4 usages  Лиза
    public Vector2d crossLine(Line line1, Line line2) {
        double a1 = get_line_a(line1);
        double b1 = get_line_b(line1);
        double c1 = get_line_c(line1);
        double a2 = get_line_a(line2);
        double b2 = get_line_b(line2);
        double c2 = get_line_c(line2);
        Vector2d vector = new Vector2d();
        if (a1 * b2 != a2 * b1 && a1 != 0) {
            vector.y = (a2 * c1 - a1 * c2) / (a1 * b2 - a2 * b1);
            vector.x = (c1 + b1 * vector.y) / a1 * -1;
        } else if (a1 * b2 != a2 * b1 && a1 == 0) {
            vector.y = -1 * c1;
            if (a2 != 0) vector.x = (c1 * b2 - c2) / a2;
            else return null;
        } else if (a1 * b2 == a2 * b1) return null;
        return vector;
    }

    /**
     * Пересекает ли отрезок прямой
     */
    3 usages  Лиза
    public boolean crossLineSegment(Line line1, Line line2, Vector2d pos_border_1, Vector2d pos_border_2) {
        Vector2d cross = crossLine(line1, line2);
        if (cross != null) {
            if ((cross.x > pos_border_1.x && cross.x < pos_border_2.x) || (cross.x < pos_border_1.x && cross.x > pos_border_2.x)) return true;
            else return false;
        } else return false;
    }
}

```

```

/**
 * Если отрезок прямой внутри треугольника больше максимального
 */
3 usages  Лиза
public void lenghtBiggerMax(Vector2d posM, Vector2d posN, Vector2d cross1, Vector2d cross2, double lenght) {
    maxLenght = lenght;
    pos1_answer = posM;
    pos2_answer = posN;
    pos1_cross = cross1;
    pos2_cross = cross2;
}

/**
 * Максимальная длина отрезка внутри треугольника
 */
5 usages
double maxLenght = 0;

```



Само решение задачи реализовано в методе **solve()**. В нем осуществляется перебор пар точек из множества и рассматриваются пересечения этих прямых с треугольником. Происходит поиск максимальной длины отрезка внутри треугольника с использованием вспомогательных методов.

```

/**
 * решение задачи
 */
1 usage  👤 Лиза *
public void solve() {
    //количество точек
    int numberPoints = points.size();
    // вектора вершин треугольника
    posA = triangle.pos1;
    posB = triangle.pos2;
    posC = triangle.pos3;
    // прямые, содержащие отрезки треугольника (заданы двумя точками)
    Line lineAB = new Line(posA, posB);
    Line lineBC = new Line(posB, posC);
    Line lineAC = new Line(posA, posC);
    // Длина отрезка внутри треугольника
    double lenght = 0;
    // перебор всех пар точек
    for(int i = 0; i < numberPoints; i++) {
        for(int j = 0; j < numberPoints; j++) {
            // две точки
            Vector2d posM = points.get(i).getPos();
            Vector2d posN = points.get(j).getPos();
            // прямая через них
            if (posM != posN) {
                Line line = new Line(posM, posN);
                // точки пересечения со сторонами треугольника
                Vector2d crossAB = null;
                Vector2d crossBC = null;
                Vector2d crossAC = null;

                // смотрим, пересекает ли прямая отрезки треугольника
                if (crossLineSegment(line, lineAB, posA, posB)) {
                    crossAB = crossLine(line, lineAB);
                }
                if (crossLineSegment(line, lineBC, posB, posC)) {
                    crossBC = crossLine(line, lineBC);
                }
                if (crossLineSegment(line, lineAC, posA, posC)) {
                    crossAC = crossLine(line, lineAC);
                }
            }
        }
    }
}

```

```

// вектор отрезка внутри треугольника
Vector2d lineSegment = null;
// если есть два пересечения со сторонами треугольника
if (crossAB != null && crossBC != null) {
    lineSegment = Vector2d.subtract(crossAB, crossBC);
    lenght = lineSegment.length();
    // если отрезок больше максимального
    if (lenght > maxLenght) {
        lenghtBiggerMax(posM, posN, crossAB, crossBC, lenght);
    }
} else if (crossBC != null && crossAC != null) {
    lineSegment = Vector2d.subtract(crossBC, crossAC);
    lenght = lineSegment.length();
    if (lenght > maxLenght) {
        lenghtBiggerMax(posM, posN, crossBC, crossAC, lenght);
    }
} else if (crossAB != null && crossAC != null) {
    lineSegment = Vector2d.subtract(crossAB, crossAC);
    lenght = lineSegment.length();
    if (lenght > maxLenght) {
        lenghtBiggerMax(posM, posN, crossAB, crossAC, lenght);
    }
}
}
}
}
}

solved = true;
}
}
}

```



## 6. Unit-тесты