**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

Галалу Антон Сергеевич.

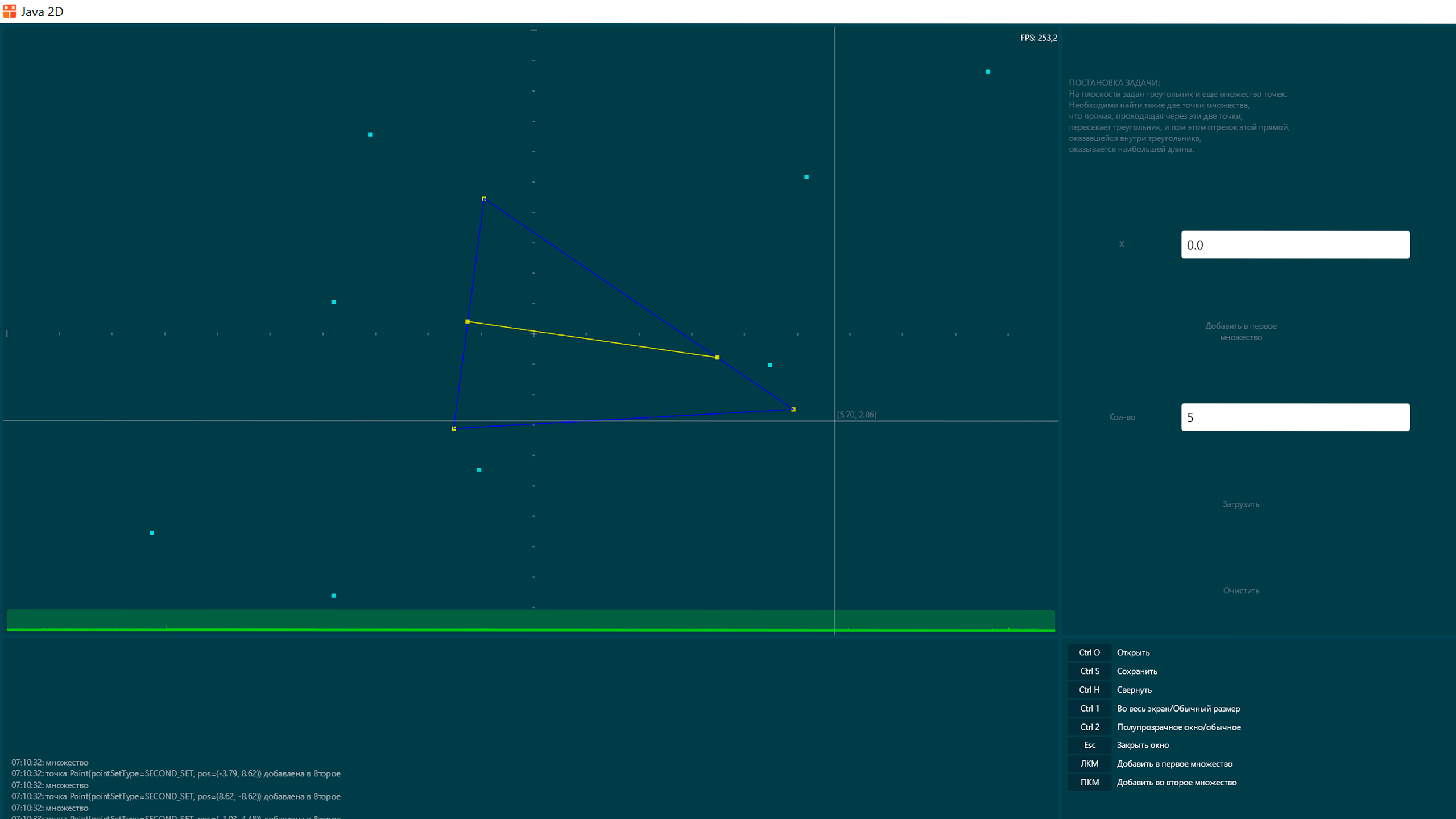
Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2022 год

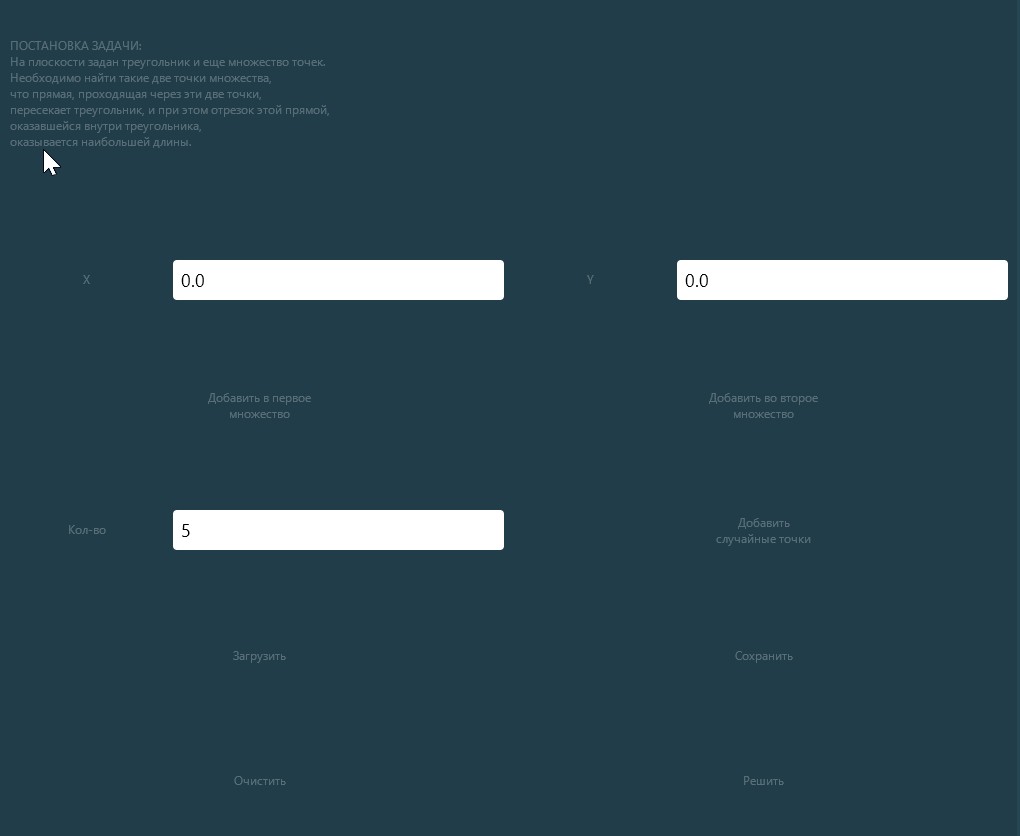
# 1. Постановка задачи

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 выделенные эти две точки, прямую, через них проходящую, и этот отрезок.



# 2. Элементы управления

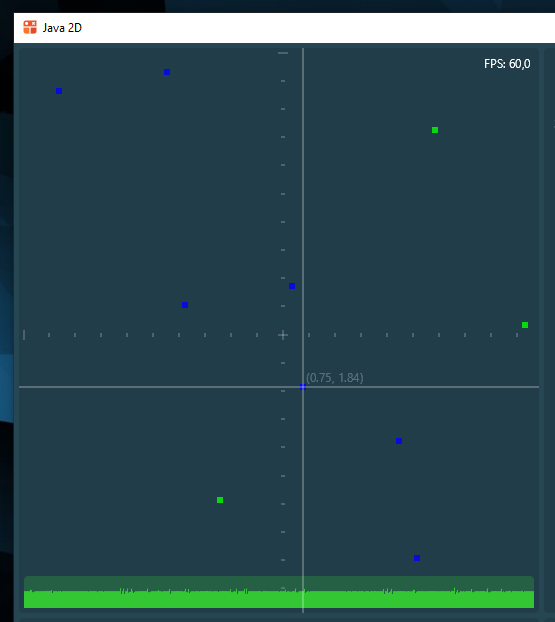
В рамках данной задачи необходимо было реализовать следующие элементы управления:



Для добавления точки по координатам было создано два поля ввода: «X» и «Y». Чтобы различить, в какое множество точка должна быть добавлена, используются две кнопки «Добавить в первое множество», «Добавить во второе множество».

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

Также программа позволяет добавлять точки с помощью клика мышью по области рисования



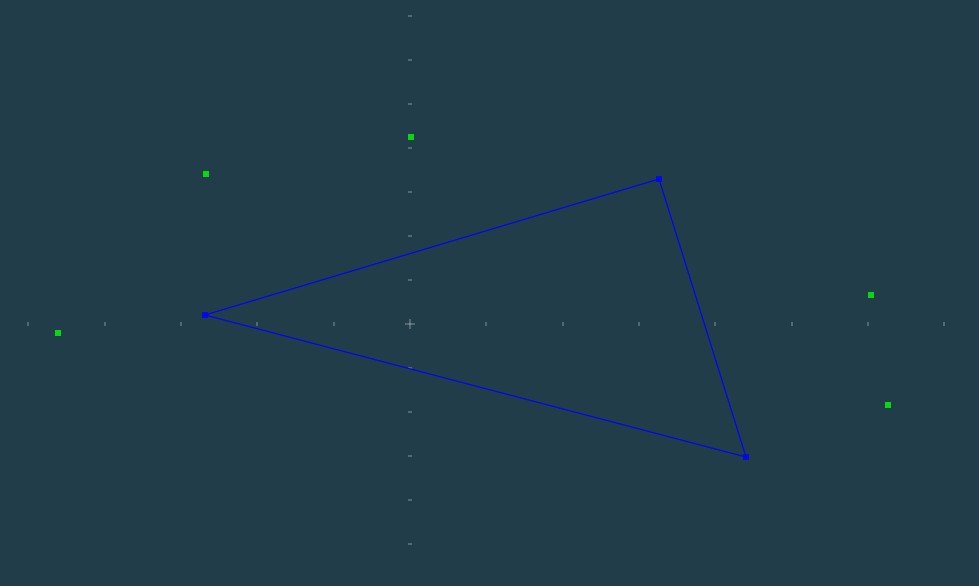
При клике левой кнопкой мыши по области рисования в месте клика создаётся точка, принадлежащая первому множеству(множество треугольника), при клике правой - второму

# 3. Структуры данных

Для того чтобы хранить точки, были разработаны классы **Point.java, Segment.java, Line.java.** Его листинг приведён в приложении А.

# 4. Рисование

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



# 5. Решение задачи

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

*/\*\**  
 *\* Решить задачу*  
 *\*/*  
*public void* solve() {  
 // очищаем списки  
  
 final\_segment.clear();  
 points3.clear();  
  
 Line l1 = new Line(triangle.get(0).pos.x, triangle.get(0).pos.y, triangle.get(1).pos.x, triangle.get(1).pos.y);  
 Line l2 = new Line(triangle.get(1).pos.x, triangle.get(1).pos.y, triangle.get(2).pos.x, triangle.get(2).pos.y);  
 Line l3 = new Line(triangle.get(2).pos.x, triangle.get(2).pos.y, triangle.get(0).pos.x, triangle.get(0).pos.y);  
 Segment s1 = new Segment(triangle.get(0).pos, triangle.get(1).pos);  
 Segment s2 = new Segment(triangle.get(1).pos, triangle.get(2).pos);  
 Segment s3 = new Segment(triangle.get(2).pos, triangle.get(0).pos);  
 boolean b1 = false;  
 for (int i = 0; i < points\_out.size(); i++) {  
 for (int j = i + 1; j < points\_out.size(); j++) {  
 b1 = false;  
 Line l = new Line(points\_out.get(i).pos.x, points\_out.get(i).pos.y, points\_out.get(j).pos.x, points\_out.get(j).pos.y);  
 if (l1.equals(l)) {  
 Vector2d[] vArr = new Vector2d[2];  
 vArr[0] = triangle.get(0).pos;  
 vArr[1] = triangle.get(1).pos;  
 points3.add(vArr);  
 } else if (l2.equals(l)) {  
 Vector2d[] vArr = new Vector2d[2];  
 vArr[0] = triangle.get(1).pos;  
 vArr[1] = triangle.get(2).pos;  
 points3.add(vArr);  
 } else if (l3.equals(l)) {  
 Vector2d[] vArr = new Vector2d[2];  
 vArr[0] = triangle.get(2).pos;  
 vArr[1] = triangle.get(0).pos;  
 points3.add(vArr);  
 } else {  
 if (s1.check(l.intersection(l1)) || s2.check(l.intersection(l2)) || s3.check(l.intersection(l3))) {  
 Vector2d[] vArr = new Vector2d[2];  
 if (l.intersection(l1) != null && s1.check(l.intersection(l1))) {  
 vArr[0] = l.intersection(l1);  
 b1 = true;  
 }  
 if (l.intersection(l2) != null && s2.check(l.intersection(l2))) {  
 if (b1) {  
 vArr[1] = l.intersection(l2);  
 b1 = false;  
 } else {  
 vArr[0] = l.intersection(l2);  
 b1 = true;  
 }  
 }  
 if (l.intersection(l3) != null && s3.check(l.intersection(l3)) && b1) {  
 vArr[1] = l.intersection(l3);  
 }  
 if (vArr[0] != null && vArr[1] != null) {  
 points3.add(vArr);  
 }  
 }  
 }  
 }  
 }  
  
 double max = 0;  
 for (Vector2d[] v : points3) {  
 if (Segment.*length*(v[0], v[1]) > max) {  
 max = Segment.*length*(v[0], v[1]);  
 }  
 }  
 int m = 0;  
 for (Vector2d[] v : points3) {  
 if (Segment.*length*(v[0], v[1]) == max) {  
 break;  
 }  
 m += 1;  
 }  
 Point pp1 = new Point(points3.get(m)[0], Point.PointSet.*FIRST\_SET*);  
 Point pp2 = new Point(points3.get(m)[1], Point.PointSet.*FIRST\_SET*);  
 final\_segment.add(pp1);  
 final\_segment.add(pp2);  
  
 solved = true;  
}

# 7. Заключение

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

# Приложение А. Point.java

package app;  
  
import com.fasterxml.jackson.annotation.JsonCreator;  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import com.fasterxml.jackson.annotation.JsonProperty;  
import misc.Misc;  
import misc.Vector2d;  
  
import java.util.Objects;

*/\*\*  
 \* Класс точки  
 \*/*public class Point {  
 */\*\*  
 \* Множества  
 \*/* public enum PointSet {  
 */\*\*  
 \* Первое  
 \*/  
 FIRST\_SET*,  
 */\*\*  
 \* Второе  
 \*/  
 SECOND\_SET* }  
  
 */\*\*  
 \* Множество, которому принадлежит точка  
 \*/* protected final PointSet pointSet;  
 */\*\*  
 \* Координаты точки  
 \*/* public final Vector2d pos;  
  
 */\*\*  
 \* Конструктор точки  
 \*  
 \** ***@param*** *pos положение точки  
 \** ***@param*** *setType множество, которому она принадлежит  
 \*/* @JsonCreator  
 public Point(@JsonProperty("pos") Vector2d pos, @JsonProperty("setType") PointSet setType) {  
 this.pos = pos;  
 this.pointSet = setType;  
 }  
  
  
 */\*\*  
 \* Получить цвет точки по её множеству  
 \*  
 \** ***@return*** *цвет точки  
 \*/* @JsonIgnore  
 public int getColor() {  
 return switch (pointSet) {  
 case *FIRST\_SET* -> Misc.*getColor*(0xCC, 0x00, 0x00, 0xFF);  
 case *SECOND\_SET* -> Misc.*getColor*(0xCC, 0x00, 0xFF, 0x0);  
 };  
 }  
  
 */\*\*  
 \* Получить положение  
 \* (нужен для json)  
 \*  
 \** ***@return*** *положение  
 \*/* public Vector2d getPos() {  
 return pos;  
 }  
  
 */\*\*  
 \* Получить множество  
 \*  
 \** ***@return*** *множество  
 \*/* public PointSet getSetType() {  
 return pointSet;  
 }  
  
  
 */\*\*  
 \* Получить название множества  
 \*  
 \** ***@return*** *название множества  
 \*/* @JsonIgnore  
 public String getSetName() {  
 return switch (pointSet) {  
 case *FIRST\_SET* -> "Первое множество";  
 case *SECOND\_SET* -> "Второе множество";  
 };  
 }  
  
 */\*\*  
 \* Строковое представление объекта  
 \*  
 \** ***@return*** *строковое представление объекта  
 \*/* @Override  
 public String toString() {  
 return "Point{" +  
 "pointSetType=" + pointSet +  
 ", 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 pointSet.equals(point.pointSet) && Objects.*equals*(pos, point.pos);  
 }  
  
 */\*\*  
 \* Получить хэш-код объекта  
 \*  
 \** ***@return*** *хэш-код объекта  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(pointSet, pos);  
 }  
}

Приложение Line.java

package app;  
  
import misc.Vector2d;  
  
import java.util.Vector;  
  
public class Line {  
 double a;  
 double b;  
 double c;  
  
 Line(double x1, double y1, double x2, double y2) {  
 a = y2 - y1;  
 b = x1 - x2;  
 c = y1 \* x2 - x1 \* y2;  
 }  
  
 @Override  
 public String toString(){  
 return a + "," + b + "," + c;  
 }  
 Vector2d intersection(Line l) {  
 if (b != 0) {  
 if (b \* l.a != a \* l.b) {  
 double x = (c \* l.b - l.c \* b) / (b \* l.a - a \* l.b);  
 double y = -1 \* (a \* x + c) / b;  
 Vector2d v = new Vector2d(x, y);  
 return v;  
 } else {  
 return null;  
 }  
 } else {  
 if (l.b == 0) {  
 return null;  
 } else {  
 double x = -c / a;  
 double y = (-l.c - l.a \* x) / l.b;  
 Vector2d v = new Vector2d(x, y);  
 return v;  
 }  
 }  
 }  
  
 public boolean equals(Line l) {  
 if (a == l.a && b == l.b && c == l.c) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
}

Приложение Segment.java

package app;  
  
import misc.Vector2d;  
  
public class Segment {  
 Vector2d v1;  
 Vector2d v2;  
  
 Segment(Vector2d v1, Vector2d v2) {  
 this.v1 = v1;  
 this.v2 = v2;  
 }  
 */\*\**  
 *\* Проверяет вхождение точки в отрезок*  
 *\**  
 *\* возвращает true или false*  
 *\*/*  
public boolean check(Vector2d v) {  
 if (v == null){  
 return true;  
 }  
 return v.x >= Math.*min*(v1.x, v2.x) && v.x <= Math.*max*(v1.x, v2.x) && v.y >= Math.*min*(v1.y, v2.y) && v.y <= Math.*max*(v1.y, v2.y);  
 }  
  
 public static double length(Vector2d v1, Vector2d v2){  
 return Math.*sqrt*(Math.*pow*(v1.x - v2.x, 2) + Math.*pow*(v1.y - v2.y, 2));  
 }  
}