**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 10-8 класса

Иванов А.И.

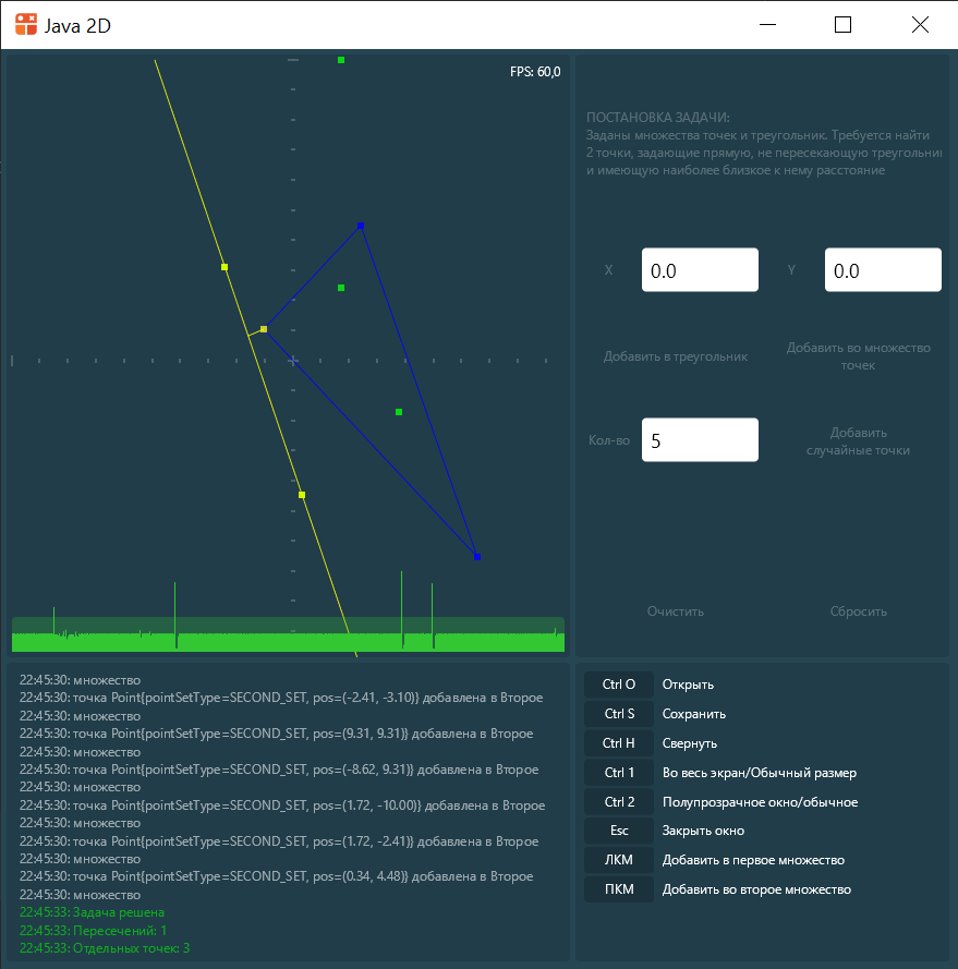
Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2022 год

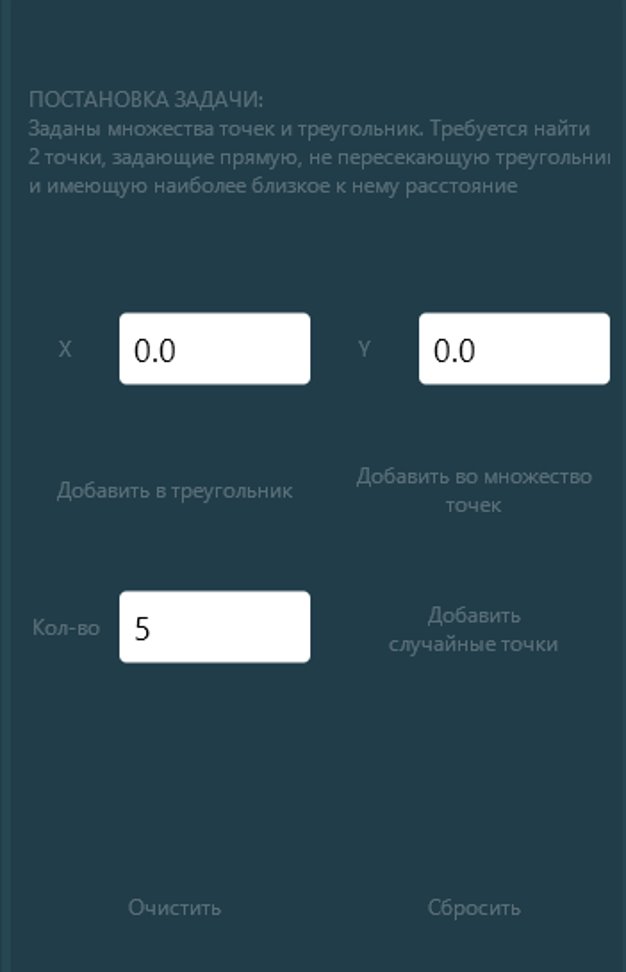
# 1. Постановка задачи

Заданы множество точек в вещественном пространстве и треугольник. Требуется найти 2 такие точки множества, что прямая проходящая через них не пересекает данный треугольник и имеет кратчайшее расстояние до него.



# 2. Элементы управления

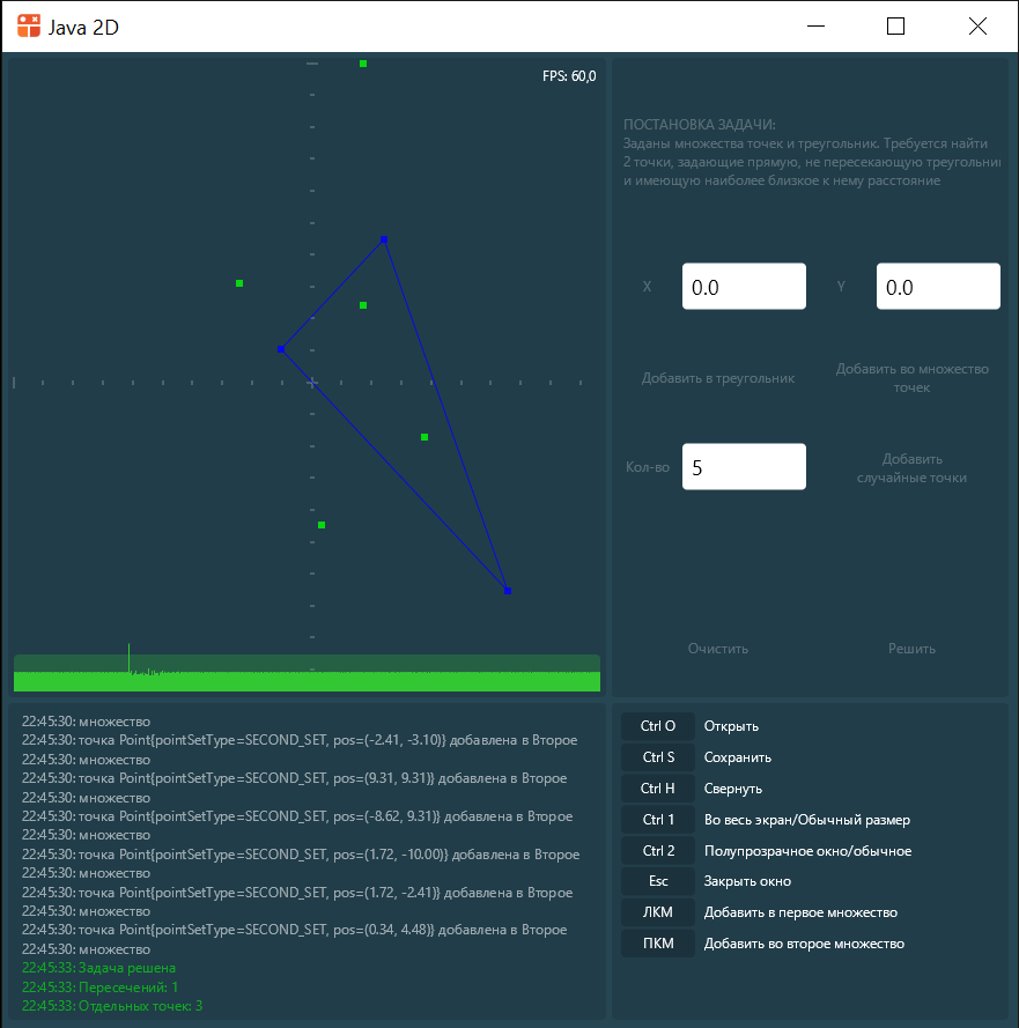
В рамках данной задачи необходимо было реализовать следующие элементы управления:



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 треугольник», «Добавить во множество точек».

Добавление случайных фигуры и точек происходит нажатием на одну кнопку: первыми записываются случайные точки вершин треугольника, затем все остальные во множество точек.

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 треугольнику, при клике правой - множеству точек

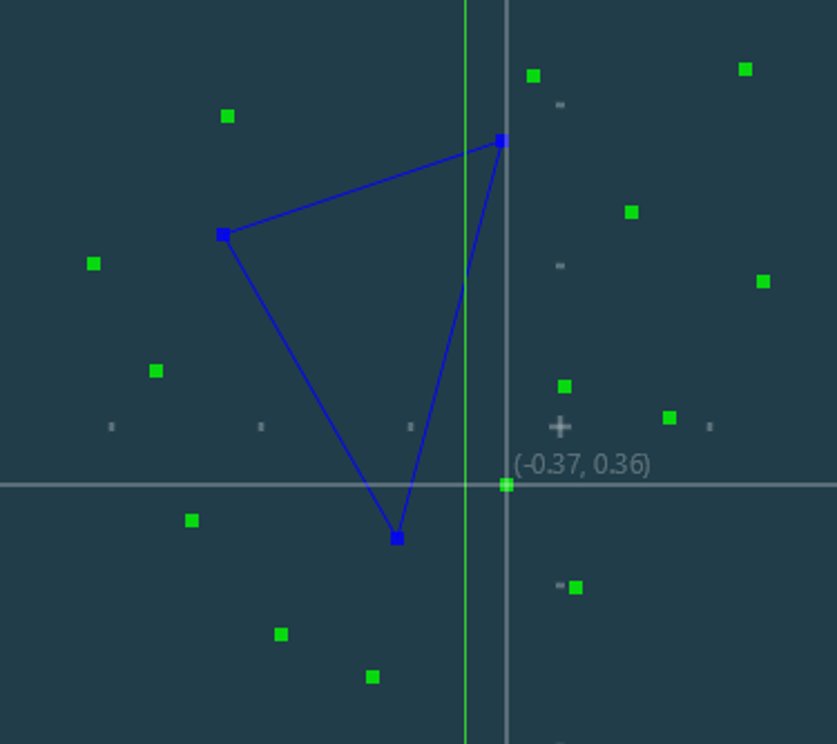
# 3. Структуры данных

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

В него были добавлены поля **pos**, соответствующее положению точки в пространстве задачи и тип множества **pointset.** Хранение типа множества обеспечено за счёт введения нового перечисления **PointSet**.

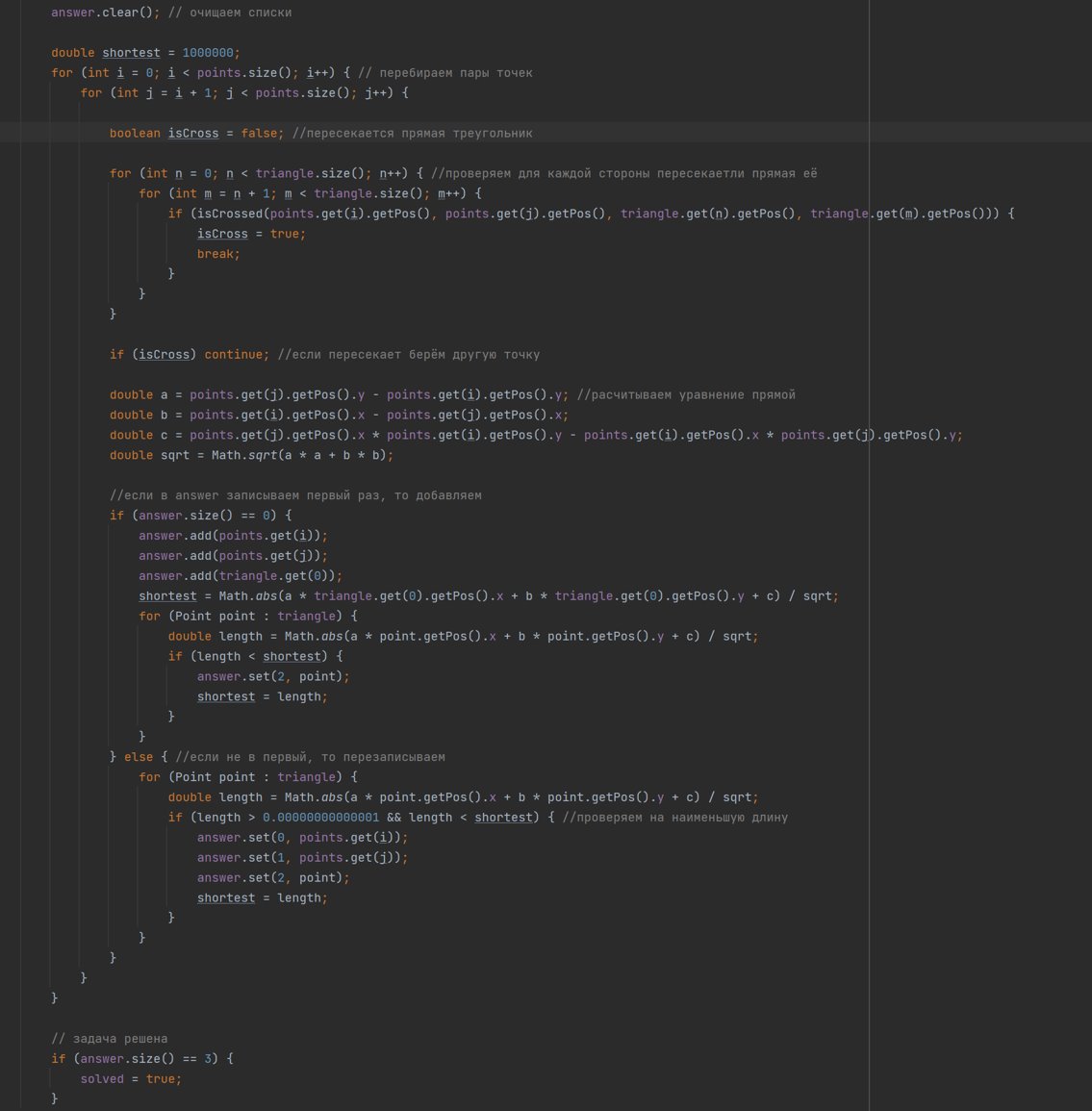
# 4. Рисование

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



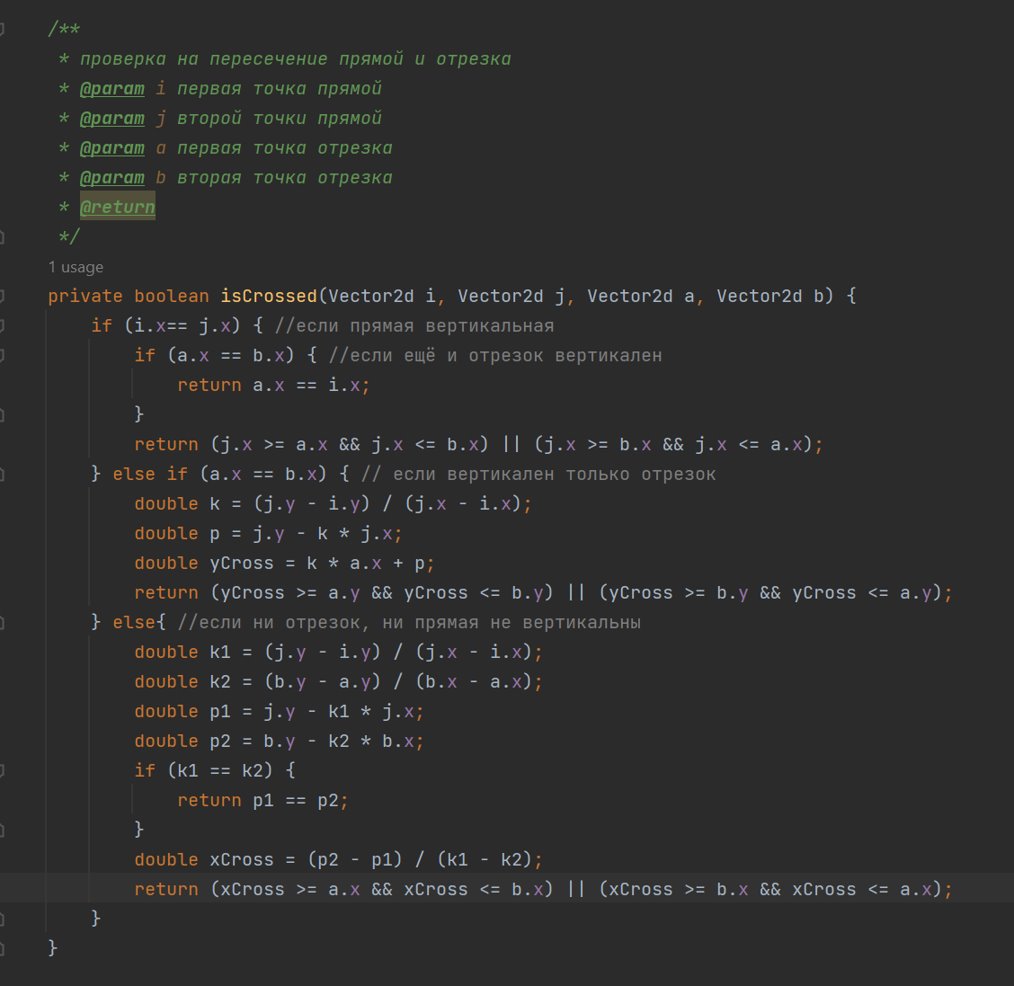
# 5. Решение задачи

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



В нём перебираются пары точек и, если они не пересекают треугорльник и имеют кратчайшее расстояние до него, то их координаты и координаты ближайшей вершины треугольника к прямой записывается в массив ответа.

Так же отдельно выделена функция isCrossed(...) для выявления пересекается ли отрезок и прямая, задающиеся по двум точкам. Решение производится через формулы прямых.



# 6. 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);  
 }  
}