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

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ПРЕЗИДЕНТСКИЙ ФИЗИКО-МАТЕМАТИЧЕСКИЙ**

**ЛИЦЕЙ №239**

**Отчет о практике**

**«Создание графических приложений на языке Java»**

Учащаяся 10-1 класса

Плотникова А.А.

Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2023 год

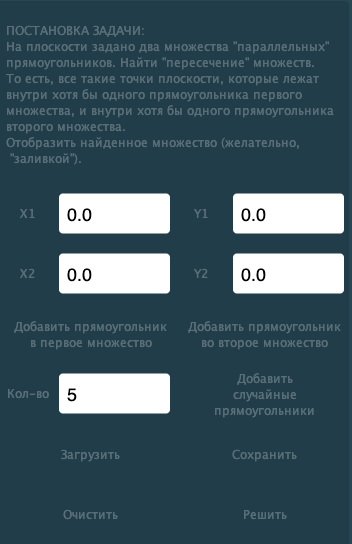
# 1. Постановка задачи

На плоскости задано два множества "параллельных" прямоугольников. Найти "пересечение" множеств. То есть, все такие точки плоскости, которые лежат внутри хотя бы одного прямоугольника первого множества, и внутри хотя бы одного прямоугольника второго множества. Отобразить найденное множество (желательно, "заливкой").



# 2. Элементы управления

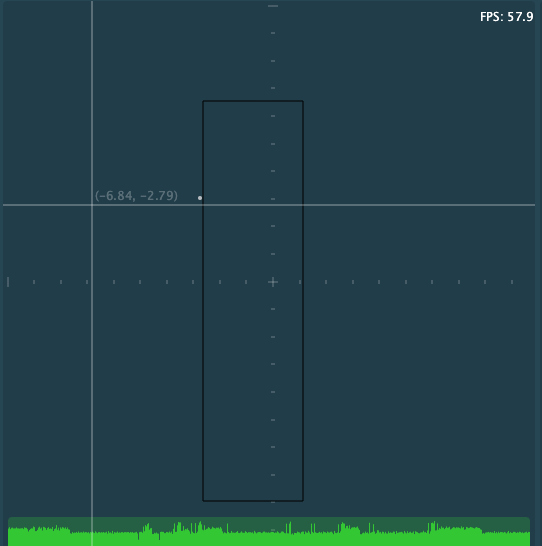
В рамках данной задачи необходимо было реализовать следующие элементы управления:



Для добавления прямоугольника по координатам было создано четыре поля ввода: «X1», «Y1» и «X2», «Y2». Чтобы различить, в какое множество точка должна быть добавлена, используются две кнопки «Добавить прямоугольник в первое множество», «Добавить прямоугольник 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 параллельных прямоугольников, которые будут добавлены.

Также программа позволяет добавлять прямоугольники с помощью клика мышью по области рисования.



При клике левой кнопкой мыши по области рисования в местах кликов создаётся прямоугольник, принадлежащий первому множеству, при клике правой – второму.

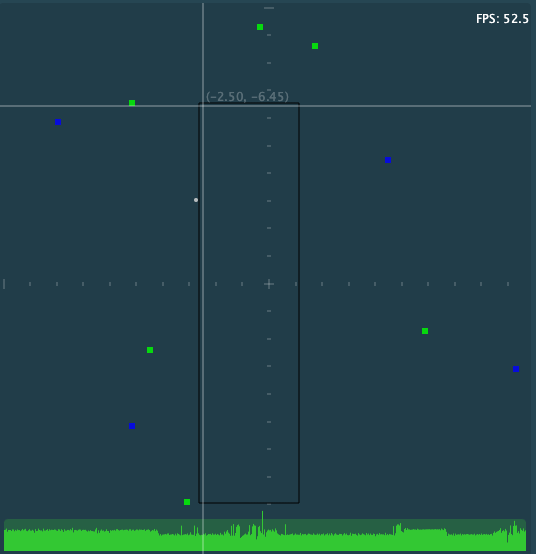
# 3. Структуры данных

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

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

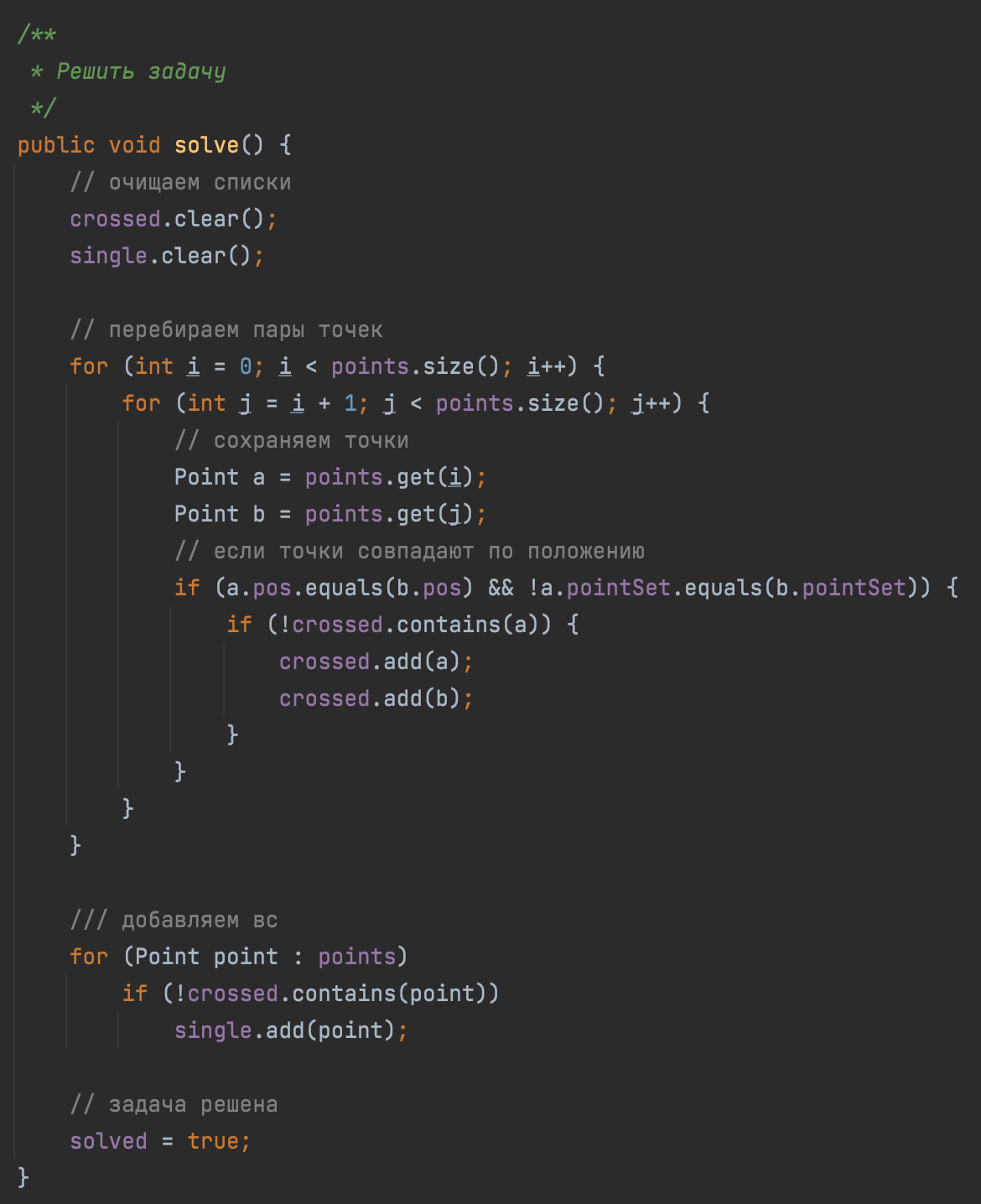
# 4. Рисование

Чтобы нарисовать прямоугольник, 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 их координаты совпадают, то обе эти точки добавляются в список пересечения.

После цикла перебираются пары точек и в множество разности добавляются все те, которые не содержатся множестве пересечения.

# 6. Проверка

Для проверки правильности решённой задачи были разработаны unit-тесты. Их листинг приведён в приложении Б.

Тест 1

Координаты вершин первого прямоугольника: }

Координаты вершин второго прямоугольника: }

Тест 2

Координаты вершин первого прямоугольника: }

Координаты вершин второго прямоугольника: }

Тест 3

Координаты вершин первого прямоугольника: }

Координаты вершин второго прямоугольника: }

Тест 4

Координаты вершин первого прямоугольника: }

Координаты вершин второго прямоугольника: }

Тест 5

Координаты вершин первого прямоугольника: }

Координаты вершин второго прямоугольника: }

Тест 6

Координаты вершин первого прямоугольника: }

Координаты вершин второго прямоугольника: }

# 7. Заключение

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

# Приложение А. Rectangle.java

package app;  
  
import lombok.Getter;import misc.Vector2d;  
import org.w3c.dom.css.Rect;  
  
import java.util.Objects;  
  
import static java.lang.Math.*max*;  
import static java.lang.Math.*min*;  
  
public class Rectangle {  
 @Getter  
 private final Vector2d posA;  
 @Getter  
 private final Vector2d posB;  
  
 public boolean checkcross (Rectangle b) {  
 // a.posA.x - точка  
 if ((this.posA.x >= *min*(b.posA.x, b.posB.x)) && (this.posA.x <= *max*(b.posA.x, b.posB.x)) && (this.posA.y >= *min*(b.posA.y, b.posB.y)) && (this.posA.x <= *max*(b.posA.y, b.posB.y))) {  
 return true;  
 }  
 if ((this.posB.x >= *min*(b.posA.x, b.posB.x)) && (this.posB.x <= *max*(b.posA.x, b.posB.x)) && (this.posB.y >= *min*(b.posA.y, b.posB.y)) && (this.posB.x <= *max*(b.posA.y, b.posB.y)))  
 return true;  
 if ((b.posA.x >= *min*(this.posA.x, this.posB.x)) && (b.posA.x <= *max*(this.posA.x, this.posB.x)) && (b.posA.y >= *min*(this.posA.y, this.posB.y)) && (b.posA.x <= *max*(this.posA.y, this.posB.y))) {  
 return true;  
 }  
 if ((b.posB.x >= *min*(this.posA.x, this.posB.x)) && (b.posB.x <= *max*(this.posA.x, this.posB.x)) && (b.posB.y >= *min*(this.posA.y, this.posB.y)) && (b.posB.x <= *max*(this.posA.y, this.posB.y)))  
 return true;  
 return false;  
 };  
  
 public Rectangle (Vector2d posA, Vector2d posB) {  
 this.posA = posA;  
 this.posB = posB;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o)  
 return true;  
 if (o == null || getClass() != o.getClass())  
 return false;  
 Rectangle rectangle = (Rectangle) o;  
 return Objects.*equals*(posA, rectangle.posA) && Objects.*equals*(posB, rectangle.posB);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(posA, posB);  
 }  
}

# Приложение Б. UnitTest.java

import app.Point;  
import app.Rectangle;  
import app.Task;  
import misc.CoordinateSystem2d;  
import misc.Vector2d;  
import org.junit.Test;  
  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.Set;  
  
*/\*\*  
 \* Класс тестирования  
 \*/*public class UnitTest {  
  
 */\*\*  
 \* Первый тест  
 \*/* @Test  
 public void test1() {  
 Rectangle r1 = new Rectangle(new Vector2d(0,0), new Vector2d(1,4));  
 Rectangle r2 = new Rectangle(new Vector2d(1,0), new Vector2d(1,-1));  
  
 assert r1.checkcross(r2);  
 }  
 */\*\*  
 \* Второй тест  
 \*/* @Test  
 public void test2() {  
 Rectangle r1 = new Rectangle(new Vector2d(-1,-1), new Vector2d(2,1));  
 Rectangle r2 = new Rectangle(new Vector2d(-1,-1), new Vector2d(2,1));  
  
 assert r1.checkcross(r2);  
 }  
 */\*\*  
 \* Третий тест  
 \*/* @Test  
 public void test3() {  
 Rectangle r1 = new Rectangle(new Vector2d(-1,-1), new Vector2d(2,1));  
 Rectangle r2 = new Rectangle(new Vector2d(-1,1), new Vector2d(2,-1));  
  
 assert r1.checkcross(r2);  
 }  
 */\*\*  
 \* Четвертый тест  
 \*/* @Test  
 public void test4() {  
 Rectangle r1 = new Rectangle(new Vector2d(-5,-4), new Vector2d(1,5));  
 Rectangle r2 = new Rectangle(new Vector2d(7,5), new Vector2d(7,5));  
  
 assert !r1.checkcross(r2);  
 assert !r2.checkcross(r1);  
 }  
 */\*\*  
 \* Пятый тест  
 \*/* @Test  
 public void test5() {  
 Rectangle r1 = new Rectangle(new Vector2d(0,0), new Vector2d(0,0));  
 Rectangle r2 = new Rectangle(new Vector2d(0,0), new Vector2d(0,0));  
  
 assert r1.checkcross(r2);  
 }  
  
 */\*\*  
 \* Шестой тест  
 \*/* @Test  
 public void test6() {  
 Rectangle r1 = new Rectangle(new Vector2d(-7,-7), new Vector2d(7,7));  
 Rectangle r2 = new Rectangle(new Vector2d(-5,-5), new Vector2d(5,5));  
  
 assert r1.checkcross(r2);  
 assert r2.checkcross(r1);  
 }  
}