**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 10-2 класса

Фролов Т.С.

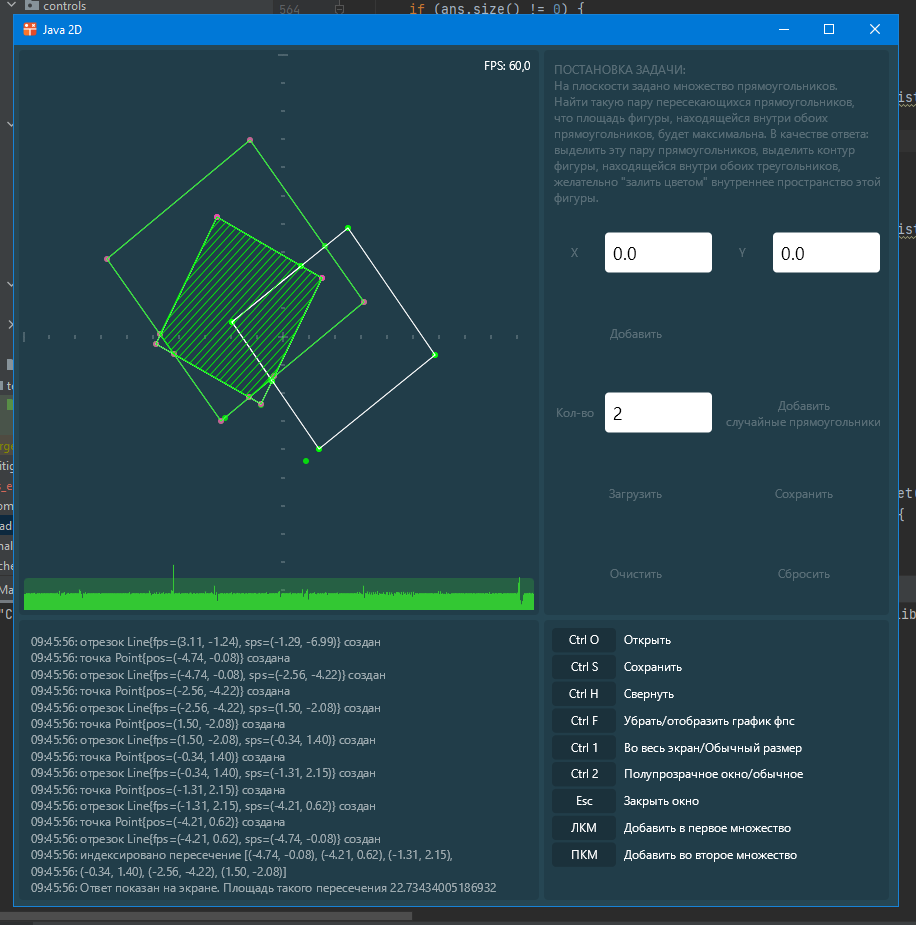
Преподаватель:

Клюнин А.О.

Санкт-Петербург – 2023 год

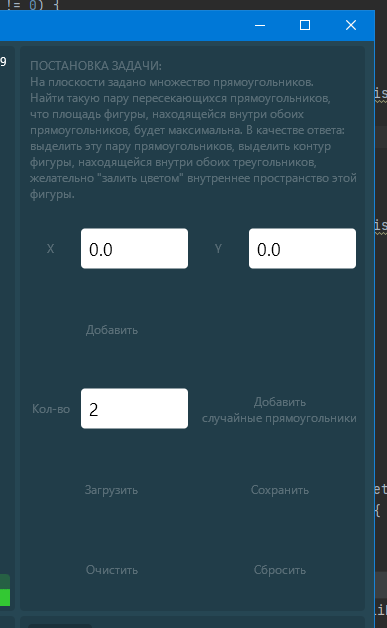
# 1. Постановка задачи

На плоскости задано множество прямоугольников. Найти такую пару пересекающихся прямоугольников, что площадь фигуры, находящейся внутри обоих прямоугольников, будет максимальна. В качестве ответа: выделить эту пару прямоугольников, выделить контур фигуры, находящейся внутри обоих прямоугольников, желательно "залить цветом" внутреннее пространство этой фигуры.



2. Элементы управления

В рамках данной задачи необходимо было реализовать следующие элементы управления:

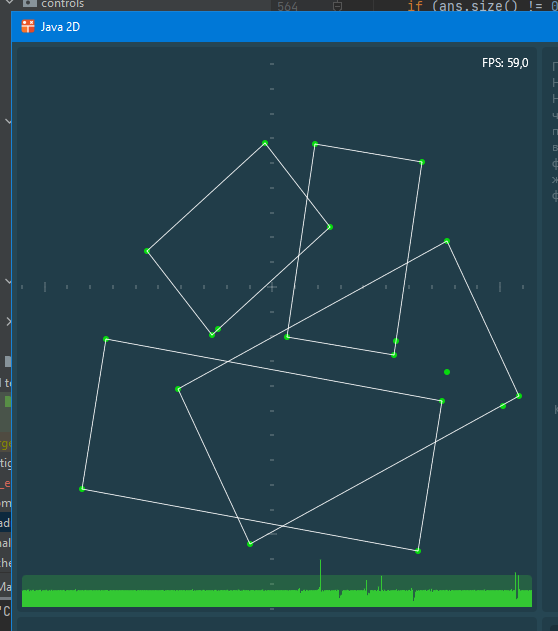


Для добавления прямоугольника необходимо ввести три точки: первые две определяют опорный отрезок, одна из сторон прямоугольника, третья определяет две остальные вершины, лежащими на одной прямой с ней.

Для добавления точки по координатам было создано два поля ввода: «X» и «Y». Чтобы добавить точку, была создана кнопка «Добавить».

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

Также программа позволяет добавлять точки, отрезки и прямоугольники с помощью клика мышью по области рисования



При клике кнопкой мыши по области рисования в месте клика создаётся точка, при втором клике – вторая точка, а, следовательно, первый отрезок прямоугольника. Ну и последняя точка задаёт оставшуюся часть прямоугольника

3. Структуры данных

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

В него были добавлены поле **pos**, соответствующее положению точки в пространстве задачи**.**

Для того чтобы хранить отрезки, был разработан класс **Line.java.** Его листинг приведён в приложении Б.

В него были добавлены поля **pointA и pointB**, соответствующее положению точек отрезка в пространстве задачи и коэффициенты прямой, содержащей этот отрезок **a, b** и **c.**

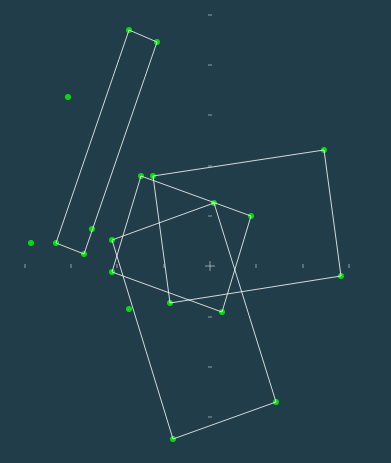
Для того чтобы хранить прямоугольники, был разработан класс **Rectangle.java.** Его листинг приведён в приложении В.

В него были добавлены поля **pointA, pointB, pointC** и **pointD**, соответствующее положению точек прямоугольника в пространстве задачи и его отрезки **AB, BC, CD** и **DA.**

4. Рисование

Чтобы нарисовать точку, использовалась команда рисования кругов **canvas.drawRRect().**

Чтобы нарисовать отрезок, использовалась команда рисования отрезков **canvas.drawLine().**



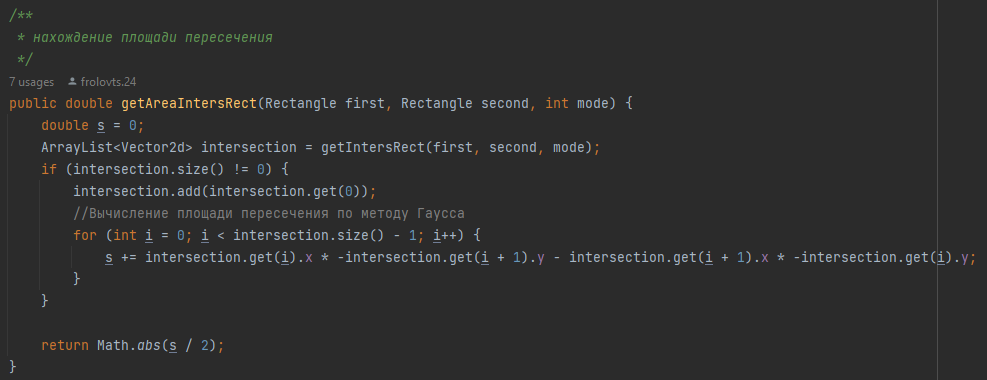
5. Решение задачи

Для решения поставленной задачи в классе **Task** был разработан метод **solve().** Его листинг приведён в приложении Г.

В нём перебираются пары прямоугольников и, если площадь их пересечения больше максимальной (по умолчанию 0), то обе эти прямоугольники добавляются в ответ.

Потом идёт их выделение и рисование пересечения, используя штриховку.

Для вычисления площади пересечения в классе **Task** был разработан метод **getAreaIntersRect().**



Он вычисляет площадь фигуры по [методу Гаусса](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B0_%D0%BF%D0%BB%D0%BE%D1%89%D0%B0%D0%B4%D0%B8_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0). Для этого ему нужно строго отсортированное множество точек, которое он получает от метода **getIntersRect(),** разработанный в классе **Task.** Его листинг приведен в приложении Д.

Этот метод находит множество точек: вершины прямоугольника, лежащие в другом прямоугольнике (точки объединения) и точки пересечения сторон прямоугольников (точки пересечения). В конце он возвращает отсортированное множество (т.к. порядок точек в методе Гаусса и в методе рисования фигуры пересечения важен) точек, используя метод **sortForCF(),** разработанный в классе **Task**. Его листинг приведен в приложении Е.

Этот же метод сортирует точки по хитрой схеме: сначала находится самая левая точка (с наименьшим x), потом все точки сортируются по тангенсу угла между вертикалью и прямой, соединяющей самую левую и обрабатываемую точку. Таким образом получается множество точек выпуклой фигуры.

6. Проверка

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

Тест 1

Этот тест проверяет, правильно ли метод **getIntersLines()** находит пересечение прямых.

Прямая 1: {(1, -4), (1, 2)}

Прямая 2: {(0, 0), (2, 2)}

Точка пересечения: (1, 1)

Тест 2

Этот тест проверяет, правильно ли метод **getIntersLines()** находит пересечение прямых.

Прямая 1: {(1, 1), (2, 2)}

Прямая 2: {(2, 1), (1, 2)}

Точка пересечения: (1.5, 1.5)

Тест 3

Этот тест проверяет, правильно ли метод **getDistance()** находит расстояние от точки до прямой.

Точка: (0, 0)

Прямая: {(2, 1), (1, 2)}

Расстояние: 2.1213

Тест 4

Этот тест проверяет, правильно ли метод **getIntersLines()** находит пересечение прямых.

Прямая 1: {(0, 2), (5, 0)}

Прямая 2: {(0, 0), (8, 4)}

Точка пересечения: (2.222, 1.111)

Тест 5

Этот тест проверяет, правильно ли метод **isIntersRect()** определяет принадлежность точки к прямоугольнику.

Точка: (0, 0)

Прямоугольник: {(-2, -1), (-2, 3), (5, 3), (5, -1)}

Принадлежит? **ДА**

Тест 6

Этот тест проверяет, правильно ли метод **isIntersRect()** определяет принадлежность точки к прямоугольнику.

Точка: (0, 0)

Прямоугольник: {( -0.39, -2.19), ( 3.89, 0.27), (2.20, 3.22), (-2.08, 0.76)}

Принадлежит? **ДА**

Тест 7

Этот тест проверяет, правильно ли метод **getAreaIntersRect()** находит площадь пересечения прямоугольников.

Прямоугольник 1: {( 3.69, 1.46), ( 2.65, -3.18), (-0.78, -2.41), ( 0.26, 2.23)}

Прямоугольник 2: {( 3.04, -5.37), ( 0.17, -2.22), (2.38, -0.20), (5.25, -3.35)}

Площадь пересечения: 4.98297

Тест 8

Этот тест проверяет, правильно ли решена задача.

Прямоугольник 1: {( 3.69, 1.46), ( 2.65, -3.18), (-0.78, -2.41), ( 0.26, 2.23)}

Прямоугольник 2: {( 3.04, -5.37), ( 0.17, -2.22), (2.38, -0.20), (5.25, -3.35)}

Прямоугольник 3: {( -1.56, -0.60), ( -1.16, 4.13), (5.84, 3.54), ( 5.44, -1.19)}

Точка пересечения 1 и 2: 4.98297

Точка пересечения 2 и 3: 0.53269

Точка пересечения 1 и 3: 9.7005

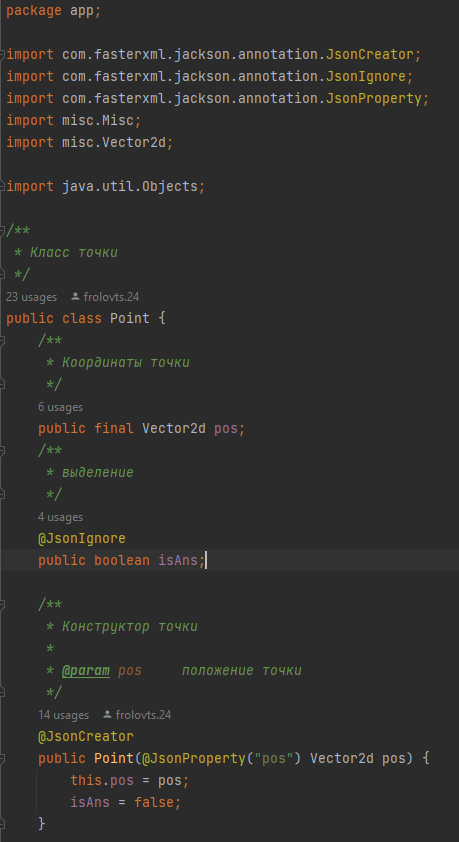
Ответ:

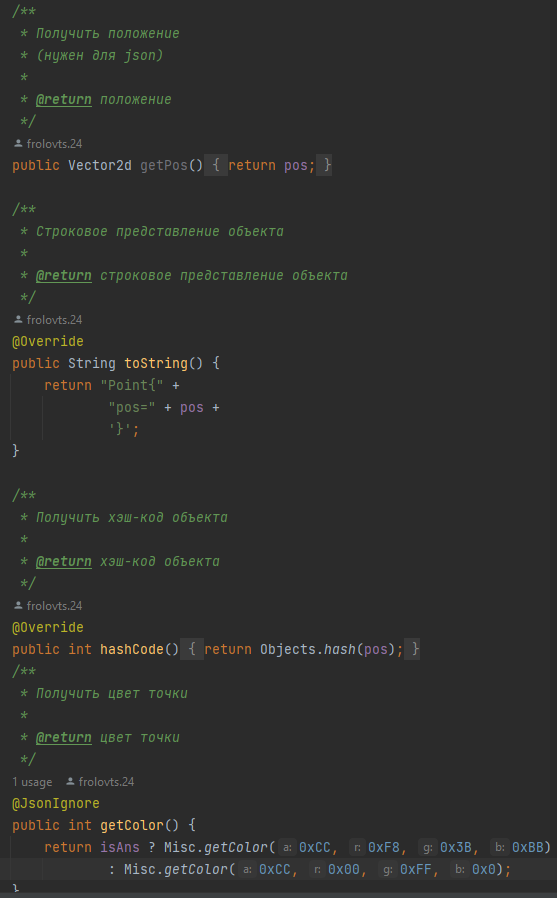
фигура: {( -0.396, -0.698), (3.139, -0.997), (3.69, 1.46), (0.26, 2.23)}

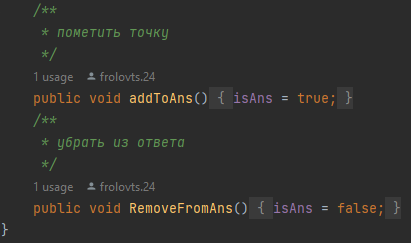
7. Заключение

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

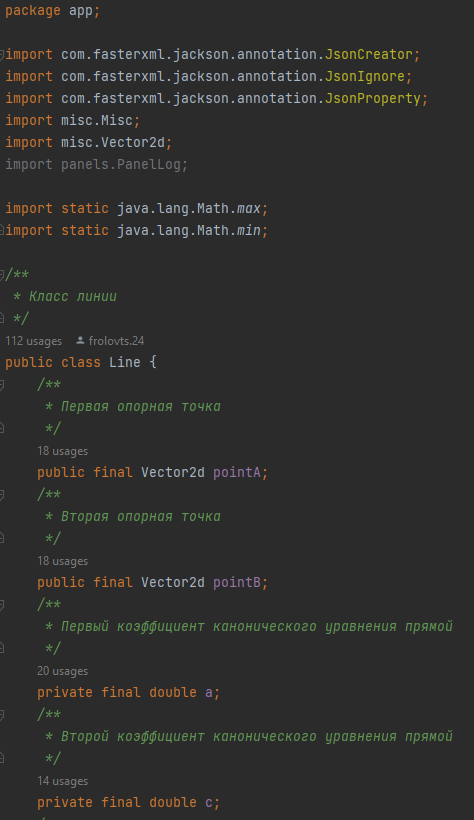
Приложение А. Point.java

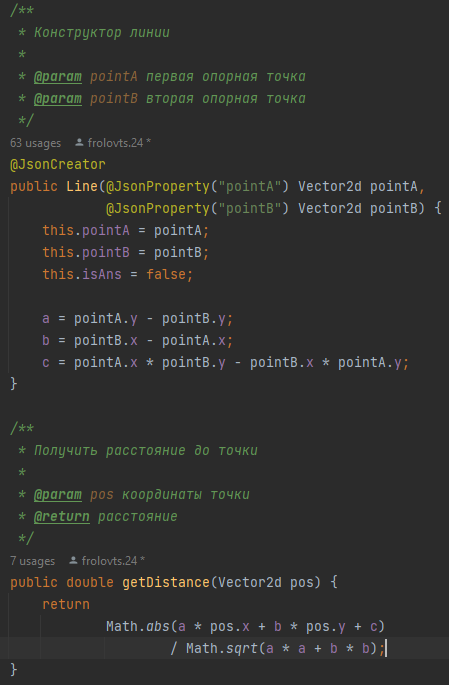


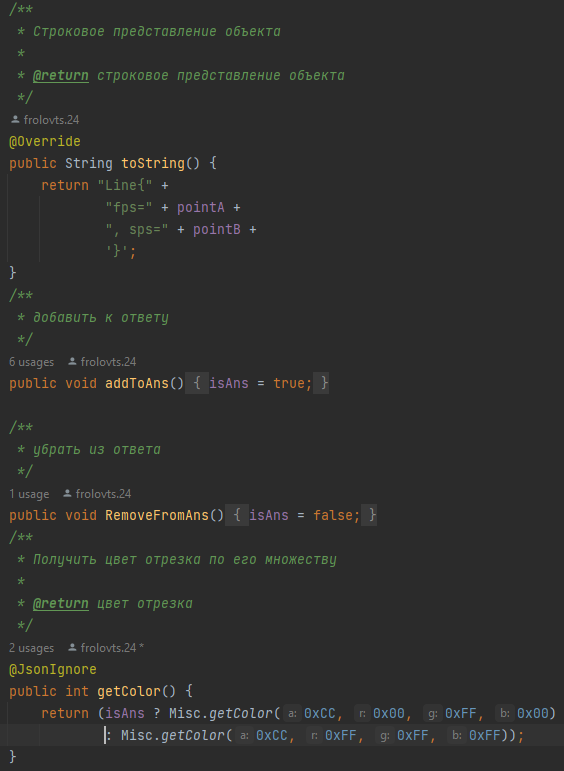


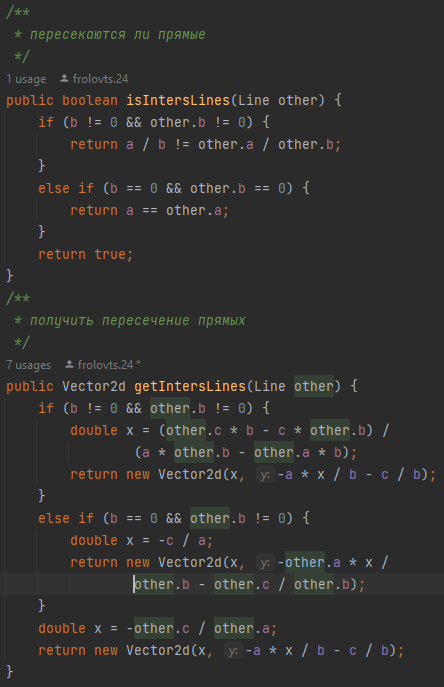


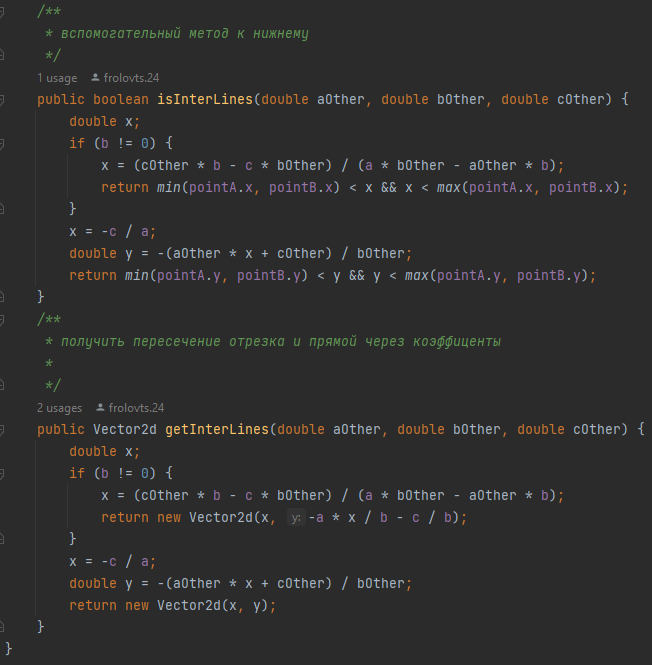
Приложение Б. Line.java





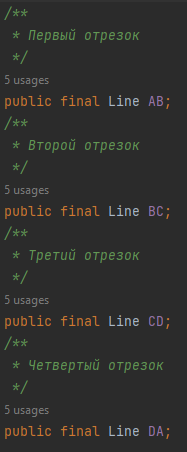


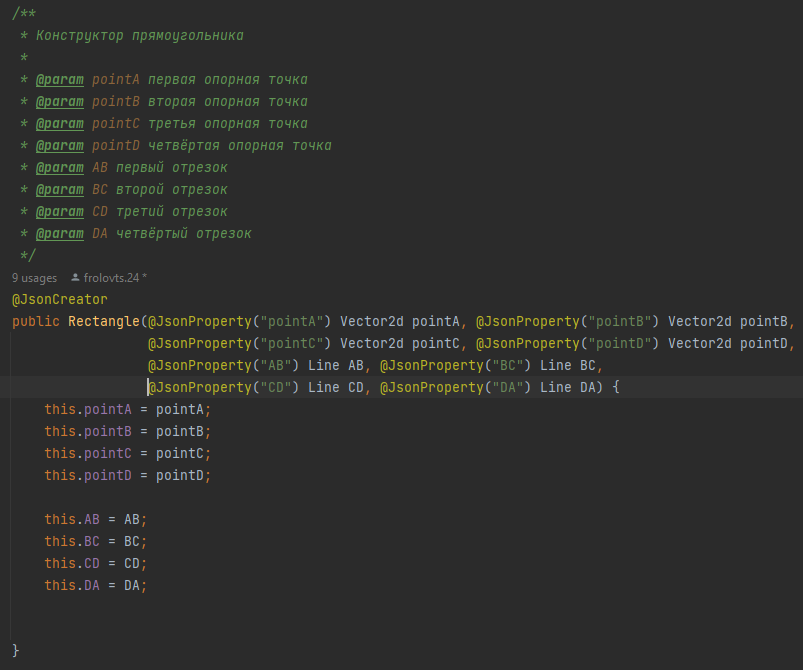


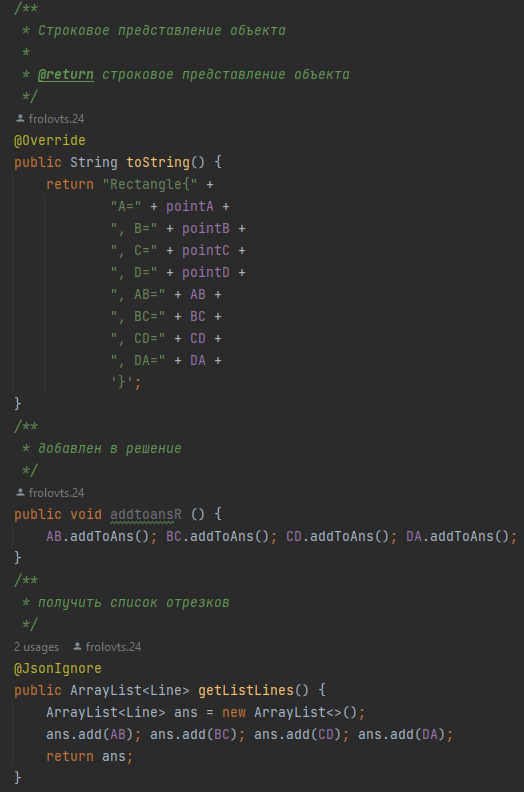


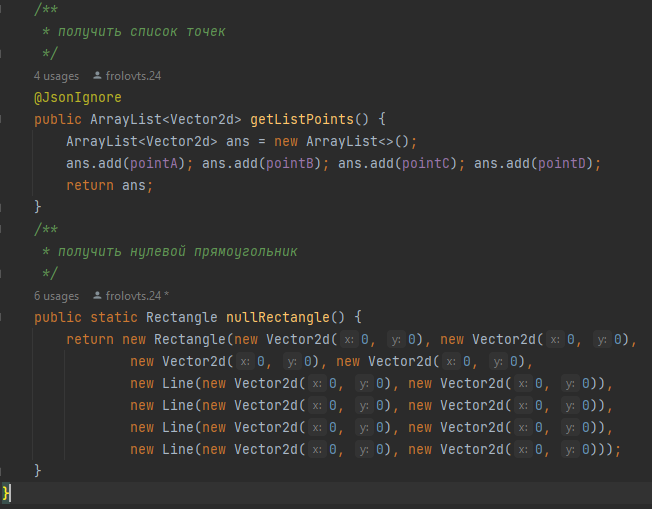
Приложение В. Rectangle.java





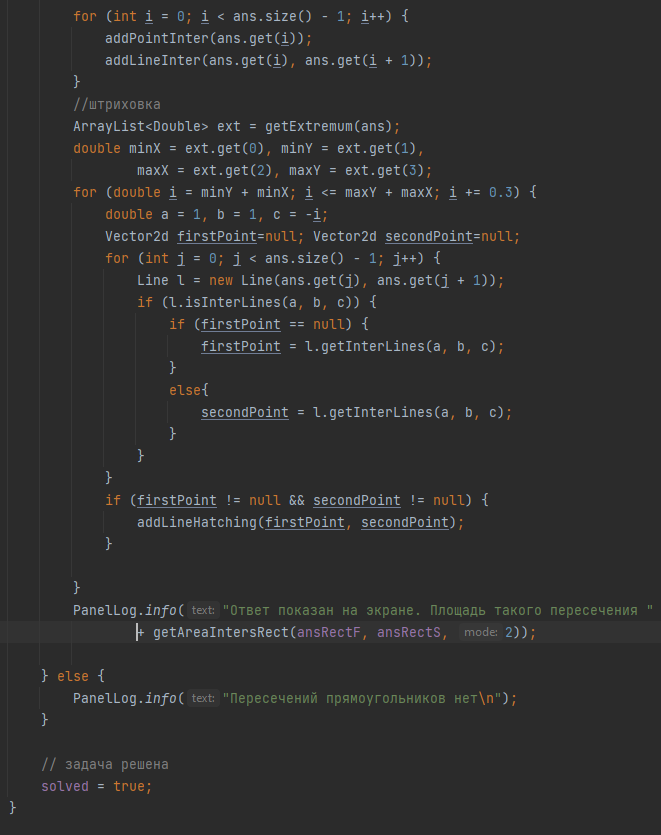




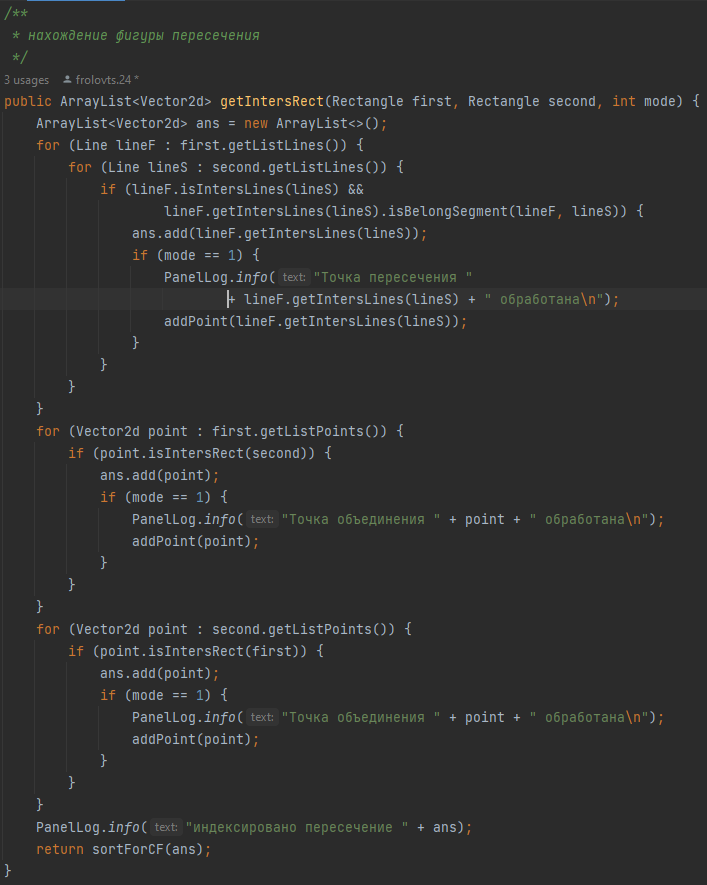


Приложение Г. Task.solve()

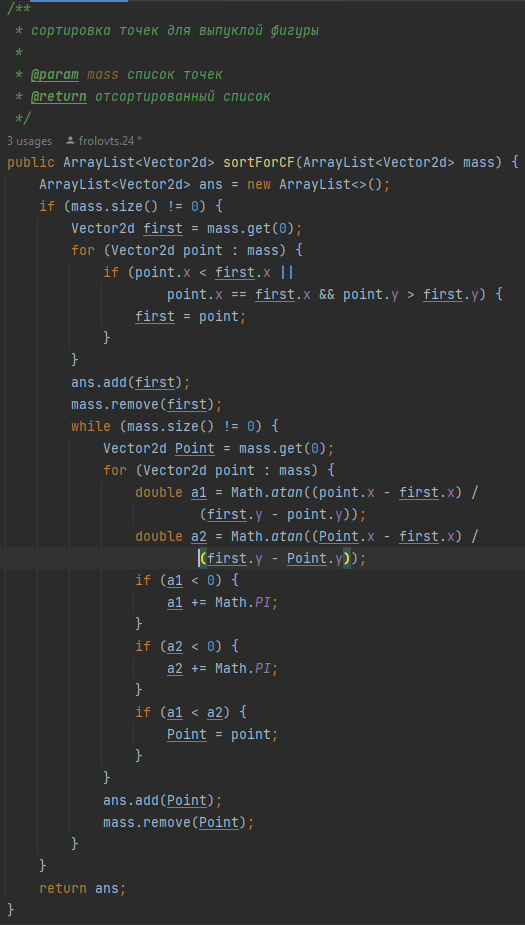




Приложение Д. Task. getIntersRect()



Приложение Е. Task.sortForCF()



Приложение Ж. UnitTest.java



