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.

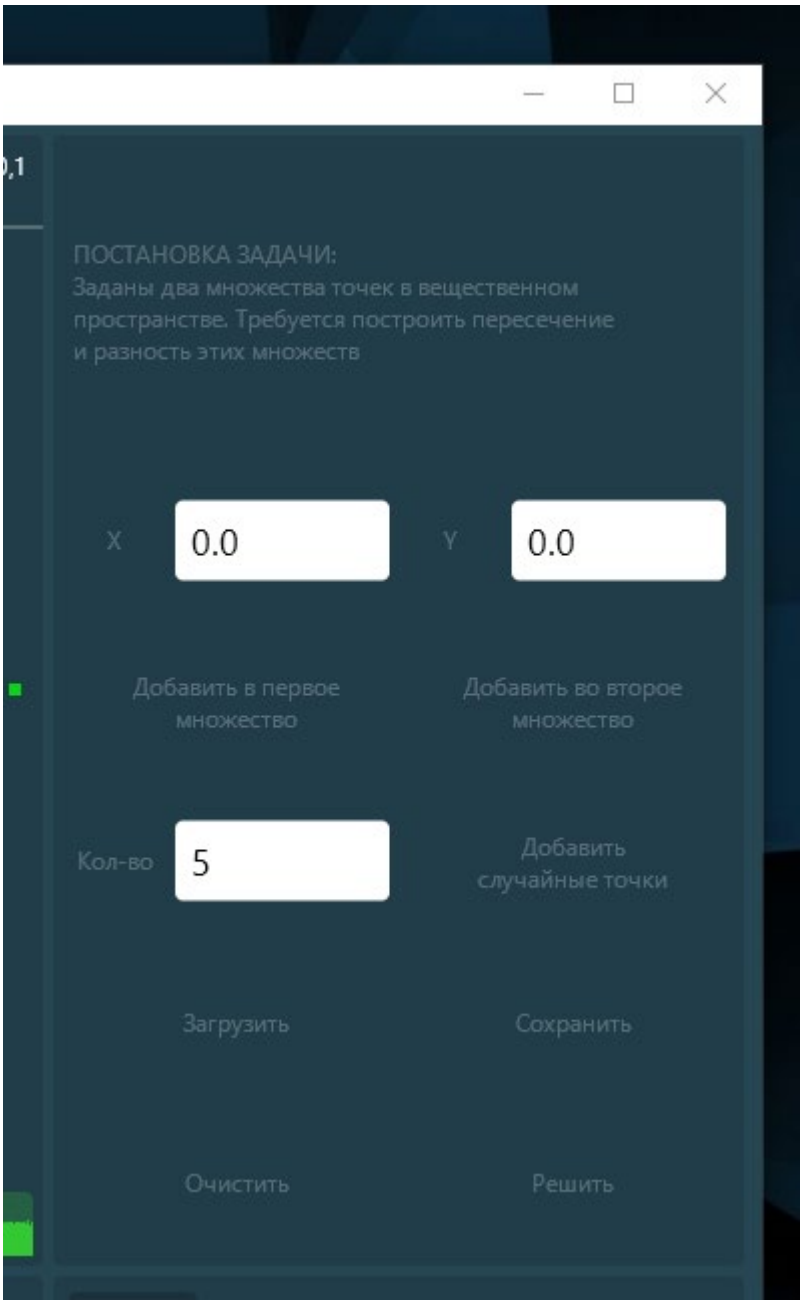
Санкт-Петербург – 2022 год

1. Постановка задачи

Заданы два множества точек в вещественном пространстве. Требуется построить пересечение и разность этих множеств.

2. Элементы управления

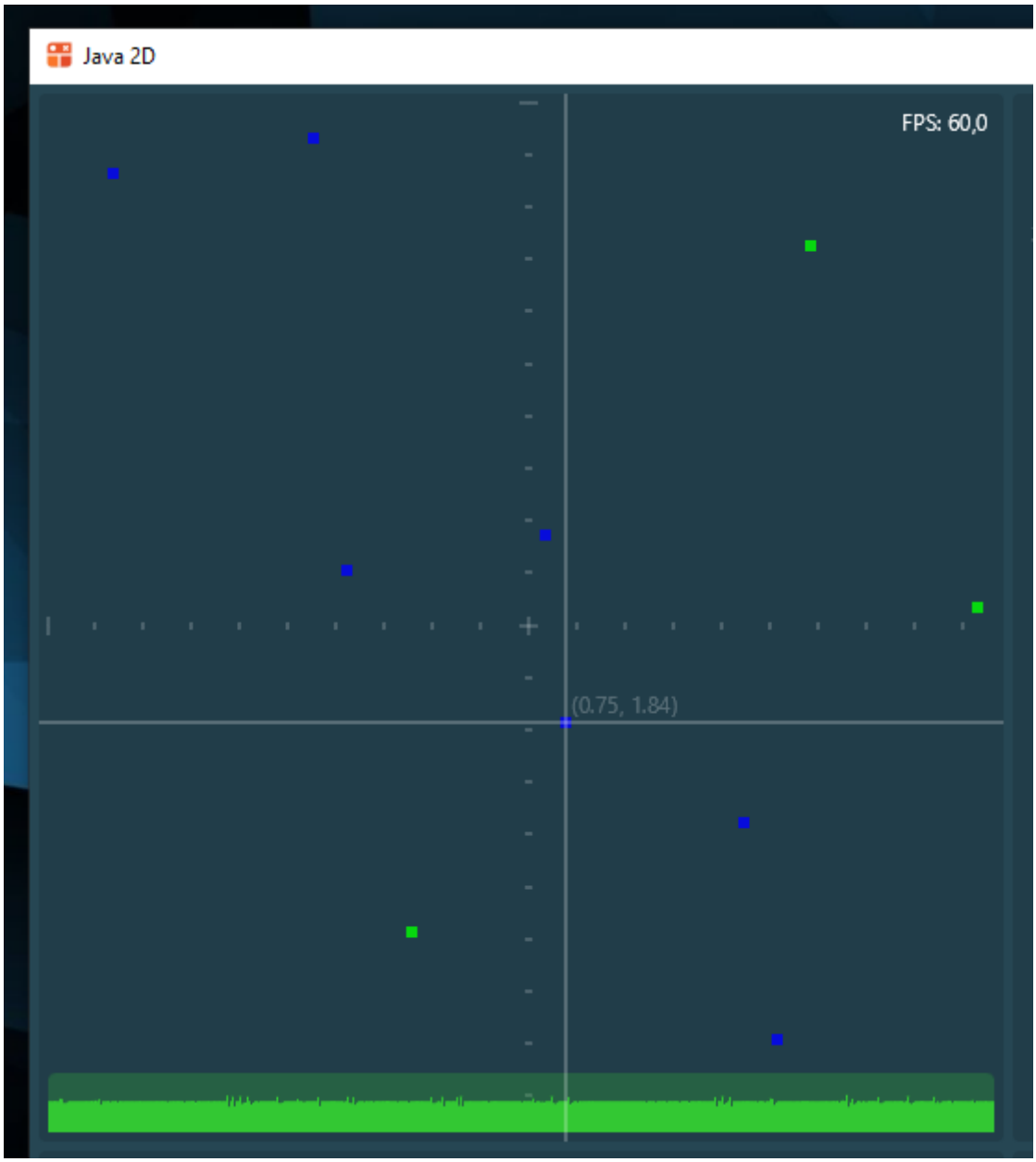
В рамках данной задачи необходимо было реализовать следующие элементы управления:



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, при клике правой - второму

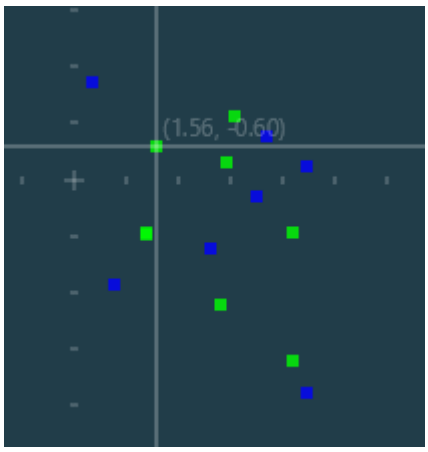
3. Структуры данных

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().**

```
/**
 * Решить задачу
 */
public void solve() {
    // очищаем списки
    crossed.clear();
    single.clear();

    // перебираем пары точек
    for (int i = 0; i < points.size(); i++) {
        for (int j = i + 1; j < points.size(); j++) {
            // сохраняем точки
            Point a = points.get(i);
            Point b = points.get(j);
            // если точки совпадают по положению
            if (a.pos.equals(b.pos) && !a.pointSet.equals(b.pointSet)) {
                if (!crossed.contains(a)) {
                    crossed.add(a);
                    crossed.add(b);
                }
            }
        }
    }

    /// добавляем вс
    for (Point point : points)
        if (!crossed.contains(point))
            single.add(point);

    // задача решена
    solved = true;
}
```

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

Точки первого множества: $\{(1,1);(-1,1);\ (2,1);(1,2)\}$

Точки второго множества: $\{(-1,1);(1,2)\}$

Точки пересечения: $\{(-1,1);(1,2)\}$

Точки разности: $\{(1,1);(2,1)\}$

Тест 2

Точки первого множества: $\{(1,1);(2,1);\ (2,2);(1,2)\}$

Точки второго множества: $\{\}$

Точки пересечения: $\{(-1,1);(1,2)\}$

Точки разности: $\{(1,1);(2,1);\ (2,2);(1,2)\}$

Тест 3

Точки первого множества: $\{(1,1);1,2)\}$

Точки второго множества: $\{(2,1);\ (2,2);\}$

Точки пересечения: $\{\}$

Точки разности: $\{(1,1);(2,1);\ (2,2);(1,2)\}$

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

```
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 Б. UnitTest.java

```
import app.Point;
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 {

    /**
     * Тест
     *
     * @param points        список точек
     * @param crossedCoords мн-во пересечений
     * @param singleCoords  мн-во разности
     */
    private static void test(ArrayList<Point> points, Set<Vector2d>
crossedCoords, Set<Vector2d> singleCoords) {
        Task task = new Task(new CoordinateSystem2d(10, 10, 20, 20), points);
        task.solve();
        // проверяем, что координат пересечения в два раза меньше, чем точек
        assert crossedCoords.size() == task.getCrossed().size() / 2;
        // проверяем, что координат разности столько же, сколько точек
        assert singleCoords.size() == task.getSingle().size();

        // проверяем, что все координаты всех точек пересечения содержатся в
множестве координат
        for (Point p : task.getCrossed()) {
            assert crossedCoords.contains(p.getPos());
        }

        // проверяем, что все координаты всех точек разности содержатся в
множестве координат
        for (Point p : task.getSingle()) {
            assert singleCoords.contains(p.getPos());
        }
    }


    /**
     * Первый тест
     */
    @Test
    public void test1() {
        ArrayList<Point> points = new ArrayList<>();

        points.add(new Point(new Vector2d(1, 1), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(-1, 1), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(-1, 1),
Point.PointSet.SECOND_SET));
        points.add(new Point(new Vector2d(2, 1), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(1, 2), Point.PointSet.SECOND_SET));
        points.add(new Point(new Vector2d(1, 2), Point.PointSet.FIRST_SET));
```

```
        Set<Vector2d> crossedCoords = new HashSet<>();
        crossedCoords.add(new Vector2d(1, 2));
        crossedCoords.add(new Vector2d(-1, 1));

        Set<Vector2d> singleCoords = new HashSet<>();
        singleCoords.add(new Vector2d(1, 1));
        singleCoords.add(new Vector2d(2, 1));

        test(points, crossedCoords, singleCoords);
    }

    /**
     * Второй тест
     */
    @Test
    public void test2() {
        ArrayList<Point> points = new ArrayList<>();

        points.add(new Point(new Vector2d(1, 1), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(2, 1), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(2, 2), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(1, 2), Point.PointSet.FIRST_SET));

        Set<Vector2d> crossedCoords = new HashSet<>();

        Set<Vector2d> singleCoords = new HashSet<>();
        singleCoords.add(new Vector2d(1, 1));
        singleCoords.add(new Vector2d(2, 1));
        singleCoords.add(new Vector2d(2, 2));
        singleCoords.add(new Vector2d(1, 2));

        test(points, crossedCoords, singleCoords);
    }

    /**
     * Третий тест
     */
    @Test
    public void test3() {
        ArrayList<Point> points = new ArrayList<>();

        points.add(new Point(new Vector2d(1, 1), Point.PointSet.FIRST_SET));
        points.add(new Point(new Vector2d(2, 1), Point.PointSet.SECOND_SET));
        points.add(new Point(new Vector2d(2, 2), Point.PointSet.SECOND_SET));
        points.add(new Point(new Vector2d(1, 2), Point.PointSet.FIRST_SET));

        Set<Vector2d> crossedCoords = new HashSet<>();

        Set<Vector2d> singleCoords = new HashSet<>();
        singleCoords.add(new Vector2d(1, 1));
        singleCoords.add(new Vector2d(2, 1));
        singleCoords.add(new Vector2d(2, 2));
        singleCoords.add(new Vector2d(1, 2));

        test(points, crossedCoords, singleCoords);
    }
}
```