МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Отчет по лабораторной работе №2**

по дисциплине

«Языки программирования для мобильной и веб-разработки»

на тему: «Наследование в Java»

**Вариант 13**

Выполнил:

Студент группы

КТмо1-12

Макаров С. В.

Проверил:

Доцент кафедры МОП ЭВМ

Лутай В.Н.

Таганрог 2023

Содержание

[1 Цель работы 3](#_Toc147148491)

[2 Постановка задачи 4](#_Toc147148492)

[3 Выполнение работы 4](#_Toc147148493)

[3.1 Диаграмма классов 5](#_Toc147148494)

[3.2 Спецификация классов 5](#_Toc147148495)

[3.3 Листинг программы 6](#_Toc147148496)

[Вывод 13](#_Toc147148497)

# Цель работы

Целью выполнения данной лабораторной работы является ознакомление с технологией построения классов, разработки методов и обработки исключений, построение иерархии классов в Java.

# Постановка задачи

Создать класс Pair (пара значений) с методами, реализующими арифметические операции сложения и вычитания. На его основе реализовать классы Fractional (дробное) и Complex (комплексное число). В классе Fractional вещественное число представляется в виде двух целых, в которых хранятся целая и дробная часть числа соответственно, в Complex – вещественная и мнимая часть комплексного числа.

# Выполнение работы

## Диаграмма классов

На рисунке 1 приведена диаграмма классов разработанного приложения.

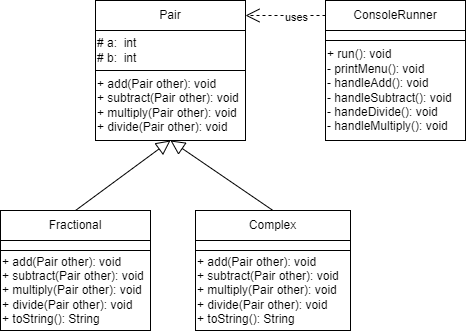


Рисунок 1 – Диаграмма классов

## Спецификация классов

Ниже приведена спецификация классов разработанного приложения.

public abstract class Pair {

    protected int a;

    protected int b;

    protected Pair(int a, int b);

    public abstract void add(Pair other);

    public abstract void subtract(Pair other);

    public abstract void multiply(Pair other);

    public abstract void divide(Pair other);

}

public class Complex extends Pair {

    public Complex(int real, int imaginary);

    @Override

    public void add(Pair complex);

    @Override

    public void subtract(Pair complex);

    @Override

    public void multiply(Pair complex);

    @Override

    public void divide(Pair complex);

    @Override

    public String toString();

}

public class Fractional extends Pair {

    public Fractional(int numerator, int denominator);

    @Override

    public void add(Pair other);

    @Override

    public void subtract(Pair other);

    @Override

    public void multiply(Pair fractional);

    @Override

    public void divide(Pair other);

    @Override

    public String toString();

    public static int validateDenominator(int denominator);

}

## Листинг программы

Ниже приведен полный листинг программного кода с разбиением по классам.

public class Point {

    private double x;

    private double y;

    public Point(double x, double y) {

        this.x = x;

        this.y = y;

    }

    public double getX() {

        return this.x;

    }

    public double getY() {

        return this.y;

    }

    public void setX(double value) {

        this.x = value;

    }

    public void setY(double value) {

        this.y = value;

    }

    public void move(double offsetX, double offsetY) {

        this.x += offsetX;

        this.y += offsetY;

    }

    @Override

    public String toString() {

        return String.format("[%.2f, %.2f]", this.x, this.y);

    }

    public static double magniture(Point from, Point to) {

        var subX = to.x - from.x;

        var subY = to.y - from.y;

        return Math.sqrt(subX \* subX + subY \* subY);

    }

}

import java.security.InvalidParameterException;

public class Triangle implements Comparable<Triangle> {

    private final Point[] points;

    public Triangle(Point a, Point b, Point c) {

        points = new Point[] { a, b, c };

        checkTriangleExisting(points);

    }

    public double findPerimeter() {

        var m = findMagnitudes(this.points);

        double a = m[0], b = m[1], c = m[2];

        return a + b + c;

    }

    public double findArea() {

        var m = findMagnitudes(this.points);

        double a = m[0], b = m[1], c = m[2];

        var p = (a + b + c) / 2;

        return Math.sqrt(p \* (p - a) \* (p - b) + (p - c));

    }

    public Point findCenter() {

        double x = 0, y = 0;

        for (Point point : points) {

            x += point.getX();

            y += point.getY();

        }

        return new Point(x / 3, y / 3);

    }

    public void move(Point offset) {

        for (Point point : points) {

            point.move(offset.getX(), offset.getY());

        }

    }

    public void scale(double factor) throws InvalidParameterException {

        if (factor <= 0) {

            throw new InvalidParameterException("Scale factor cannot be less than null");

        }

        for (Point point : points) {

            double x = point.getX(), y = point.getY();

            point.setX(x \* factor);

            point.setY(y \* factor);

        }

    }

    @Override

    public String toString() {

        return String.format("{\n  %s\n  %s\n  %s\n}\n", points[0], points[1], points[2]);

    }

    @Override

    public int compareTo(Triangle other) {

        var areaDiff = this.findArea() - other.findArea();

        return Double.compare(areaDiff, 0);

    }

    private static double[] findMagnitudes(Point[] points) {

        var a = Point.magniture(points[0], points[1]);

        var b = Point.magniture(points[1], points[2]);

        var c = Point.magniture(points[2], points[0]);

        return new double[] { a, b, c };

    }

    private static void checkTriangleExisting(Point[] points) throws InvalidParameterException {

        var m = findMagnitudes(points);

        double a = m[0], b = m[1], c = m[2];

        if ((a + b <= c) || (b + c <= a) || (c + a <= b)) {

            throw new InvalidParameterException("Triangle with this poinst doesn't exist");

        }

    }

}

import java.util.ArrayList;

import java.util.Scanner;

public class ConsoleRunner {

    private final Scanner sc;

    private Triangle triangle = null;

    public ConsoleRunner() {

        this.sc = new Scanner(System.in);

    }

    public void run() {

        try {

            this.triangle = readTriangle();

        } catch (Exception e) {

            System.err.println(e.getMessage());

            return;

        }

        printMenu();

        while (true) {

            try {

                loop();

            } catch (Exception e) {

                System.err.println(e.getMessage());

            }

        }

    }

    private void loop() {

        System.out.print(">>> ");

        int operation = sc.nextInt();

        switch (operation) {

            case 0:

                System.exit(0);

                return;

            case 1:

                printMenu();

                break;

            case 2:

                handlePrintTriangle();

                break;

            case 3:

                handlePrintCenter();

                break;

            case 4:

                handlePrintArea();

                break;

            case 5:

                handlePrintPerimeter();

                break;

            case 6:

                handleMove();

                break;

            case 7:

                handleScale();

                break;

            case 8:

                handleCompare();

                break;

            default:

                break;

        }

    }

    private void printMenu() {

        System.out.println("0 - exit; 1 - print menu;\n" +

                           "2 - print triangle; 3 - print center; 4 - print area; 5 - print perimeter;\n" +

                           "6 - move triangle; 7 - scale triangle; 8 - compare by area;\n");

    }

    private void handlePrintTriangle() {

        System.out.println(this.triangle);

    }

    private void handlePrintArea() {

        System.out.println(this.triangle.findArea());

    }

    private void handlePrintPerimeter() {

        System.out.println(this.triangle.findPerimeter());

    }

    private void handlePrintCenter() {

        System.out.println(this.triangle.findCenter());

    }

    private void handleMove() {

        System.out.println("Enter offset (x, y)");

        var x = sc.nextDouble();

        var y = sc.nextDouble();

        this.triangle.move(new Point(x, y));

    }

    private void handleScale() {

        System.out.println("Enter scale factor");

        var scale = sc.nextDouble();

        this.triangle.scale(scale);

    }

    private void handleCompare() {

        Triangle other = readTriangle();

        var compare = this.triangle.compareTo(other);

        if (compare > 0) {

            System.out.println("greater");

        }

        else if (compare == 0) {

            System.out.println("equal");

        }

        else {

            System.out.println("less");

        }

    }

    private Triangle readTriangle() {

        System.out.println("Введите координаты вершин треугольника:");

        ArrayList<Point> points = new ArrayList<Point>();

        for (int i = 0; i < 3; i++) {

            var x = sc.nextDouble();

            var y = sc.nextDouble();

            points.add(new Point(x, y));

        }

        return new Triangle(points.get(0), points.get(1), points.get(2));

    }

}

class Program {

    public static void main(String[] args) {

        new ConsoleRunner().run();

    }

}

Вывод

Выполнив лабораторную работу, мной были освоены принципы создания классов и их объектов в языке Java, описания их методов, а также обработки исключений и работы с вводом/выводом (файловым и консольным).