Лабораторная работа №2 по дисциплине «Объектно-ориентированное программирование»

Выполнил: Чичкин Данила Александрович

Группа: 6204-010302D

Оглавление

Задание 1

Организовал структуру проекта и создал директорию functions

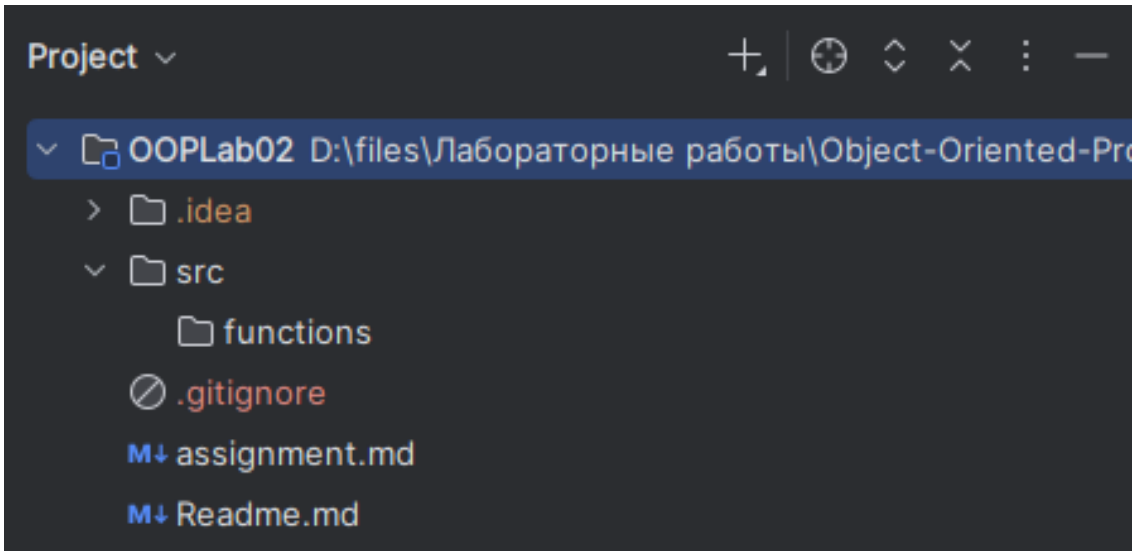

Рисунок 1 – Начальная структура проекта

Задание 2

В пакете functions я создал класс FunctionPoint с модификатором доступа public, так как доступ к классу должен осуществляться вне пакета. В классе реализовал:

- два приватных поля x, y с типом данных double;
- конструктор, создающий объект по точкам x и y;
- конструктор, создающий объект по точкам другого объекта;
- конструктор без параметров;
- методы доступа к полям get и set.

FunctionPoint.java

```
package functions;

public class FunctionPoint {
    private double x, y;

    // Конструктор, создающий объект по точкам x и y
    public FunctionPoint(double x, double y) {
        this.x = x;
        this.y = y;
    }

    // Конструктор, создающий объект по точкам другого объекта
    public FunctionPoint(FunctionPoint point) {
        this.x = point.x;
        this.y = point.y;
    }

    // Конструктор без парамертов
    public FunctionPoint() {
        x = 0;
        y = 0;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public void setX(double x) {
        this.x = x;
    }

    public void setY(double y) {
        this.y = y;
    }
}
```

Задание 3

В пакете functions я создал класс TabulatedFunction с модификатором доступа public, так как доступ к классу должен осуществляться вне пакета. В классе реализовал:

– приватное поле points типа FunctionPoint[] для хранения точек;

– конструктор TabulatedFunction(double leftX, double rightX, int pointsCount), создающий объект функции по заданным границам, а также количеству точек для табулирования. Точки создаются через равные интервалы по x, значения функции в точках равны 0;

– конструктор TabulatedFunction(double leftX, double rightX, double[] values), создающий объект функции по заданным границам, а также по значениям функции в виде массива. Точки создаются через равные интервалы по x, значения y берутся из массива;

Подробные комментарии присутствуют в файле с кодом, но в отчет я их не добавил, чтобы соблюсти условие об общем количестве символов (< 10 000)

TabulatedFunction.java

```
package functions;

public class TabulatedFunction {
    private FunctionPoint[] points;

    public TabulatedFunction(double leftX, double rightX, int
pointsCount) {
        points = new FunctionPoint[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < (pointsCount - 1); i++) {
            points[i] = new FunctionPoint(leftX + i*step, 0);
        }
        points[pointsCount-1] = new FunctionPoint(rightX, 0);
    }

    public TabulatedFunction(double leftX, double rightX, double[]
values) {
        int pointsCount = values.length;
        points = new FunctionPoint[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < (pointsCount - 1); i++) {
            points[i] = new FunctionPoint(leftX + i*step,
values[i]);
        }
        points[pointsCount-1] = new FunctionPoint(rightX,
values[pointsCount-1]);
    }
}
```

Задание 4

В классе TabulatedFunction я описал следующие методы:

– double getLeftDomainBorder() – возвращает значение левой границы области определения табулированной функции;

– double getRightDomainBorder() – возвращает значение правой границы области определения табулированной функции;

– double getFunctionValue(double x) – если x находится внутри области определения табулированной функции, метод возвращает значение функции в точке x, иначе возвращает значение неопределённости Double.NaN.

Третий метод сперва проверяет, находится ли точка x внутри области определения. Если да, то цикл for перебирает индексы массива точек начиная с 1 заканчивая последним. Если значение x меньше значения x рассматриваемой точки points[i], то значение функции в точке x можно вычислить, подставив значение x в формулу прямой от двух точек points[i-1] и points[i]:

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

Выразив значение функции y, получим формулу:

y = y1 + (y2-y1) * (x-x1) / (x2-x1)

Код методов:

```
public double getLeftDomainBorder() {
    return points[0].getX();
}

public double getRightDomainBorder() {
    return points[points.length - 1].getX();
}

public double getFunctionValue(double x) {
    if (x >= getLeftDomainBorder() && x <= getRightDomainBorder())
{
        for (int i = 1; i < points.length; i++) {
            if (x <= points[i].getX()) {
                double y1 = points[i-1].getY();
                double y2 = points[i].getY();
                double x1 = points[i-1].getX();
                double x2 = points[i].getX();
                return y1 + (y2-y1) * (x-x1) / (x2-x1);
            }
        }
    }
    return Double.NaN;
}
```

Задание 5

Добавил приватное поле size для хранения количества точек. Оно потребуется при выполнении задания 6. Вот примеры исправления в функциях:

```
public TabulatedFunction(double leftX, double rightX, int
pointsCount) {
    size = pointsCount;
    points = new FunctionPoint[size];
    double step = (rightX - leftX) / (size - 1);
    for (int i = 0; i < (size - 1); i++) {
        points[i] = new FunctionPoint(leftX + i*step, 0);
    }
    points[size-1] = new FunctionPoint(rightX, 0);
}
```

```
public double getRightDomainBorder() {
    return points[size - 1].getX();
}
```

В классе TabulatedFunction я описал следующие методы:

int getPointsCount() – возвращает количество точек;

FunctionPoint getPoint(int index) – возвращает копию точки, соответствующую переданному индексу;

void setPoint(int index, FunctionPoint point) – заменяет указанную точку табулированной функции на переданную;

double getPointX(int index) – возвращает значение абсциссы точки с указанным номером;

void setPointX(int index, double x) – изменяет значение абсциссы точки с указанным номером;

double getPointY(int index) – возвращает значение ординаты точки с указанным номером;

void setPointY(int index, double y) – изменяет значение ординаты точки с указанным номером;

Код вышеописанных функций:

```
public int getPointsCount() {
    return size;
}

public FunctionPoint getPoint(int index) {
    return new FunctionPoint(points[index]);
}

public void setPoint(int index, FunctionPoint point) {
```

```
    if (index > 0 && index < (size-1) && point.getX() >
points[index-1].getX() && point.getX() < points[index+1].getX()) {
        points[index] = new FunctionPoint(point);
    }
}

public double getPointX(int index) {
    return points[index].getX();
}

public void setPointX(int index, double x) {
    if (index > 0 && index < (size-1) && x > points[index-
1].getX() && x < points[index+1].getX()) {
        points[index].setX(x);
    }
}

public double getPointY(int index) {
    return points[index].getY();
}

public void setPointY(int index, double y) {
    points[index].setY(y);
}
```

Задание 6

В классе TabulatedFunction я описал метод deletePoint(), он удаляет точку со сдвигом элементов при size > 2. Также я описал метод addPoint(), он находит позицию для вставки, сдвигает элементы и добавляет точку, расширяя массив при заполнении:

```
public void deletePoint(int index) {
    if (size > 2) {
        System.arraycopy(points, index+1, points, index, size -
index - 1);
        size--;
    }
}

public void addPoint(FunctionPoint point) {
    if (size == points.length) {
        FunctionPoint[] tempPoints = new FunctionPoint[size * 2];
        System.arraycopy(points, 0, tempPoints, 0, size);
        points = tempPoints;
    }
    if (point.getX() > points[size-1].getX()) {
        points[size] = new FunctionPoint(point);
    } else {
        for (int i = 0; i < size; i++) {
            if (point.getX() < points[i].getX()) {
                System.arraycopy(points, i, points, i+1, size-i);
                points[i] = new FunctionPoint(point);
                break;
            }
        }
    }
    size++;
}
```

Задание 7

Создал класс Main с точкой входа программы. Создал экземпляр класса TabulatedFunction и задал для него значения функции $y = x^2 - 1$. Вывел в консоль саму функцию и область определения. Далее вывел значения функции на ряде точек, удалил, изменил и добавил по одной точке и вывел некоторые значения функции:

Main.java

```
import functions.*;

class Main {
    public static void main(String[] args) {
        // Функция y = x^2 - 1 на [-5; 5]
        double[] values = {24, 15, 8, 3, 0, -1, 0, 3, 8, 15, 24};
        TabulatedFunction func = new TabulatedFunction(-5, 5,
values);

        System.out.printf("%-21s %s%n", "Функция:", "y = x^2 -
1");
        System.out.printf("%-21s [%.2f; %.2f]%n%n", "Область
определения:", func.getLeftDomainBorder(),
func.getRightDomainBorder());

        System.out.println("До модификаций:");
        // Точки вне области определения
        System.out.printf("f(-6) = %.2f%n",
func.getFunctionValue(-6));
        System.out.printf("f(10) = %.2f%n",
func.getFunctionValue(10));

        // Точки внутри
        System.out.printf("f(0.1) = %.2f%n",
func.getFunctionValue(0.1));
        System.out.printf("f(0.2) = %.2f%n",
func.getFunctionValue(0.2));
        System.out.printf("f(0.3) = %.2f%n",
func.getFunctionValue(0.3));
        System.out.printf("f(-2.5) = %.2f%n",
func.getFunctionValue(-2.5));
        System.out.printf("f(0.3) = %.2f%n",
func.getFunctionValue(0.3));
        System.out.printf("f(3.7) = %.2f%n",
func.getFunctionValue(3.7));

        // Модификации
        func.deletePoint(3);
        func.addPoint(new FunctionPoint(-2, 10));
        func.setPointY(5, 100);

        System.out.println("\nПосле модификаций:");
        System.out.printf("f(-2.5) = %.2f%n",
func.getFunctionValue(-2.5));
        System.out.printf("f(0.3) = %.2f%n",
func.getFunctionValue(0.3));
```

```
        System.out.printf("f(3.7) = %.2f%n",
func.getFunctionValue(3.7));
    }
}
```

В результате выполнения программа вывела в консоль:

```
Функция:               y = x^2 - 1
Область определения:   [-5,00; 5,00]

До модификаций:
f(-6) = NaN
f(10) = NaN
f(0.1) = -0,90
f(0.2) = -0,80
f(0.3) = -0,70
f(-2.5) = 5,50
f(0.3) = -0,70
f(3.7) = 12,90

После модификаций:
f(-2.5) = 9,00
f(0.3) = 70,00
f(3.7) = 12,90
```