ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №2

Студент Луковникова Е.Д

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание №1

Создала пакет functions.

Задание №2

В пакете functions создала класс FunctionPoint, объект которого описывает одну точку табулированной функции.

В классе описаны следующие конструкторы:

• FunctionPoint(double x, double y) – создаёт объект точки с заданными координатами;

• FunctionPoint(FunctionPoint point) – создаёт объект точки с теми же координатами, что у указанной точки;

• FunctionPoint() – создаёт точку с координатами (0; 0).

```
package functions;

public class FunctionPoint {
    private double x;
    private double y;

    public FunctionPoint(double x, double y){
        this.x = x;
        this.y = y;
    }

    public FunctionPoint(FunctionPoint point){
        this.x = point.x;
        this.y = point.y;
    }

    public FunctionPoint() {
        this.x = 0;
        this.y = 0;
    }

        public double getX() {
        return x;
    }
```

Скрин 1

Также создала по два геттер и сеттер метода для x и y.

```
    public double getX() {
        return x;
    }

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

    public double getY() {
        return y;
    }

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

Скрин 2

Задание №3

В пакете functions создала класс TabulatedFunction, объект которого описывает табулированную функцию.

Для хранения данных о точках используется массив типа FunctionPoint. Точки в нём упорядочены по значению координаты x.

В классе описаны следующие конструкторы:

• TabulatedFunction(double leftX, double rightX, int pointsCount) – создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования (значения функции в точках считать равными 0);

• TabulatedFunction(double leftX, double rightX, double[] values) – вместо количества точек получает значения функции в виде массива.

Точки создаются через равные интервалы по x.

```
package functions;
public class TabulatedFunction {
    private FunctionPoint[] points;
    private int pointsCount;
    private final double EPSILON_DOUBLE = 1e-10;

    public TabulatedFunction(double leftX, double rightX, int pointsCount){
        if (pointsCount < 2){
            throw new IllegalArgumentException(s: "Количество точек должно быть не менее 2");
        }

        if (leftX>=rightX){
            throw new IllegalArgumentException(s: "Правая граница должна быть больше левой");
        }

        this.pointsCount = pointsCount;
        this.points = new FunctionPoint[pointsCount+5];

        double h = (rightX-leftX)/(pointsCount-1);//h - шаг точек
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i*h;
            points[i] = new FunctionPoint(x,y: 0.0);
        }


    }
```

Скрин 3

```java
public TabulatedFunction(double leftX, double rightX, double[] values){
    if (values == null){
        throw new IllegalArgumentException(s: "Массив не может быть пустым");
    }

    if (values.length < 2){
        throw new IllegalArgumentException(s: "Массив должен содержать не менее 2");
    }

    if (leftX>=rightX){
        throw new IllegalArgumentException(s: "Правая граница должна быть больше левой");
    }

    this.pointsCount = values.length;
    this.points = new FunctionPoint[pointsCount+5];

     double h = (rightX-leftX)/(pointsCount-1);//h - шаг точек
    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + i*h;
        points[i] = new FunctionPoint(x,values[i]);
    }

}
```

Скрин 4

Задание №4

Создала два метода для возвращения левой и правой границ.

• Метод double getLeftDomainBorder() возвращает значение левой границы области определения табулированной функции.

• Метод double getRightDomainBorder() возвращает значение правой границы области определения табулированной функции.

• Метод double getFunctionValue(double x) возвращает значение функции в точке x, если эта точка лежит в области определения функции. В противном случае метод возвращает значение неопределённости. При расчёте значения функции использовала линейную интерполяцию. Воспользовалась уравнением прямой, проходящей через две заданные различающиеся точки.

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1} * (x - x_1)$$

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

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

public double getFunctionValue(double x){
    double M = 0;
    if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
        return Double.NaN;
    }

    for (int i = 0; i < pointsCount - 1; i++) {
        double x_1 = points[i].getX();
        double x_2 = points[i + 1].getX();

        if (x > x_1 && x < x_2) {
            double y1 = points[i].getY();
            double y2 = points[i + 1].getY();
            M = y1 + (y2 - y1) * (x - x_1) / (x_2 - x_1);
        }

        if(Math.abs(x - points[i].getX()) < EPSILON_DOUBLE){
            M = points[i].getY();
        }

    }

    return M;
}
```

Скрин 5

Задание №5

Создала методы позволяющие получать значение точки с помощью входящего индекса (getPoint позволяет создать и вернуть копию точки, getPointX получить значение по x, setPointX изменить значение x, аналогичные методы есть для y), каждый из методов имеет проверку для индекса, чтобы он не выходил за границы функции и не был равен 0.

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

public FunctionPoint getPoint(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    return new FunctionPoint(points[index]);
}

public void setPoint(int index, FunctionPoint point) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }

    if (point.getX() <= points[index - 1].getX()) {
        return;
    }
    if (point.getX() >= points[index + 1].getX()) {
        return;
    }

    points[index] = new FunctionPoint(point);
}

public double getPointX(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    return points[index].getX();
}
```

Скрин 6

```java
public void setPointX(int index, double x) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }


    if (x <= points[index - 1].getX()) {
        return;
    }
    if (x >= points[index + 1].getX()) {
        return;
    }

    points[index].setX(x);
}

public double getPointY(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    return points[index].getY();
}

public void setPointY(int index, double y) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    points[index].setY(y);
}
```

Скрин 7

Задание №6

Создала два метода, deletePoint позволяет удалить точку из массива, имеет в себе проверку входящего индекса, также, если индекс будет стоять на крайней позиции, то не будет задействована функция arraycopy, в другом случае массив будет копироваться и смещаться влево. Есть обновление границ после внесенных изменений, addPoint позволяет добавить точку в массив.

```
public void deletePoint(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    if (pointsCount < 3) {
        throw new IllegalStateException(s: "Невозможно удалить - нужно минимум две точки");
    }

    System.arraycopy(points, index + 1, points, index, pointsCount - index - 1);
    pointsCount--;
    points[pointsCount]=null;
}

public void addPoint(FunctionPoint point) {
    int insertIndex = 0;
    while (insertIndex < pointsCount && points[insertIndex].getX() < point.getX()) {
        insertIndex++;
    }


    if (insertIndex < pointsCount && points[insertIndex].getX() == point.getX()) {
        return;
    }


    if (pointsCount == points.length) {
        FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
        System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, pointsCount);
        points = newPoints;
    }


    System.arraycopy(points, insertIndex, points, insertIndex + 1, pointsCount - insertIndex);


    points[insertIndex] = new FunctionPoint(point);
    pointsCount++;
}
```

Скрин 8

Задание №7

В main создала массив значений Y = {1, 3, 4, 9, 16}, на основе которого инициализировала табулированную функцию с помощью конструктора TabulatedFunction, задав диапазон по X от 0 до 4. Затем протестировала вычисление значения функции в точке X = 3.3 с помощью линейной интерполяции. После этого изменила значение Y в точке с индексом 3 на 8 и координату X этой же точки на 3.5. Далее добавила новую точку с координатами (3, 5) в набор данных и удалил эту же точку. На каждом этапе выводила информацию о текущем состоянии функции для визуализации изменений.

```java
import functions.*;

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        //f(x) = x^2 на интервале [0, 4]
        double[] values = {1, 3, 4, 9, 16};
        TabulatedFunction function = new TabulatedFunction(leftX: 0, rightX: 4, values);

        System.out.println(x: "Исходная функция:");
        printFunctionInfo(function);


        System.out.println(x: "\nЗначения функции в различных точках:");
        testFuncVal(function);


        System.out.println(x: "\nПосле изменения точек:");
        test(function);


        System.out.println(x: "\nПосле добавления и удаления точек:");
        testAddDel(function);
    }

    private static void printFunctionInfo(TabulatedFunction function) {
        System.out.printf(format: "Область определения: [%.1f, %.1f]\n",
            function.getLeftDomainBorder(), function.getRightDomainBorder());
        System.out.println("Количество точек: " + function.getPointsCount());

        for (int i = 0; i < function.getPointsCount(); i++) {
            FunctionPoint point = function.getPoint(i);
            System.out.printf(format: "Точка %d: (%.1f, %.1f)\n", i, point.getX(), point.getY());
        }
    }

    private static void testFuncVal(TabulatedFunction function) {
            double x = 3.3;
        double y = function.getFunctionValue(x);
        System.out.println("Значение функции X = " + x + " Y = " + y);
    }
```

Скрин 9

```java
private static void test(TabulatedFunction function) {

    function.setPointY(index: 3, y: 8);
    System.out.println(x: "После изменения y в точке 3 на 8.0:");
    printFunctionInfo(function);


    function.setPointX(index: 3, x: 3.5);
    System.out.println(x: "После изменения x в точке 3 на 3.5:");
    printFunctionInfo(function);
}

private static void testAddDel(TabulatedFunction function) {

    function.addPoint(new FunctionPoint(x: 3, y: 5));
    System.out.println(x: "После добавления точки (3, 5):");
    printFunctionInfo(function);


    function.deletePoint(index: 3);
    System.out.println(x: "После удаления точки с индексом 3:");
    printFunctionInfo(function);
}
```

Скрин 10

```
Исходная функция:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 9,0)
Точка 4: (4,0, 16,0)

Значения функции в различных точках:
Значение функции X = 3.3 Y = 11.099999999999998

После изменения точек:
После изменения y в точке 3 на 8.0:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 8,0)
Точка 4: (4,0, 16,0)
После изменения x в точке 3 на 3.5:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,5, 8,0)
Точка 4: (4,0, 16,0)

После добавления и удаления точек:
После добавления точки (3, 5):
Область определения: [0,0, 4,0]
Количество точек: 6
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 5,0)
Точка 4: (3,5, 8,0)
Точка 5: (4,0, 16,0)
После удаления точки с индексом 3:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,5, 8,0)
Точка 4: (4,0, 16,0)
```

Скрин 11