Министерство науки и высшего образования Российской Федерации федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» Институт информатики и кибернетики Кафедра технической кибернетики

Лабораторная работа №2

По дисциплине «Обьектно-ориентированное программирование»

Студент: Кораблев Д.С.

Группа: 6203-010302D

Самара, 2025

Содержание

Задание 1

Создал пакет function, в котором потом создадал классы программ.

Задание 2

Создал класс «FunctionPoint» с двумя приватными полями для точек по оси x и y, необходимые по заданию конструкторы.

```
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;
        }
```

Рисунок 1 - Класс «FunctionPoint»

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

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

        public double getX()
        {
            return this.x;
        }

        public  double getY()
        {
            return this.y;
        }
    }
```

Рисунок 2 - Геттеры и сеттеры в классе «FunctionPoint»

Задание 3

Создал конструкторы необходимые для создания сеточной (табулированной) функции в двух разных случаях. В первом случае, когда передаётся количество элементов и границы интервала, создаётся массив точек содержащих координаты X с шагом, а каждая координата У принимает стандартное значение равное нулю.

Во втором случае, в функцию передаются границы интервала, а также массив значений по У(values). Далее точно также создаётся массив точек содержащий координаты X и У (значения которые мы передали с помощью массива values).

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

}

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

Рисунок 3 - Функции табулирования

Задание 4

Создал два метода для возвращения левой и правой границ.

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

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

Рисунок 4 - Методы возвращения левой и правой границ

Метод «getFunctionValue» позволяет узнать значение У в заданной точке X с помощью формулы для линейной интерполяции.

В методе присутствуют несколько проверок:

1. Попадает ли точка, значение который мы бы хотели узнать, в заданный интервал;
2. Проверки значений X на левой и правой границе, с помощью сравнения модуля разности с заданным эпсилон (взято рекомендованное значение, в зависимости от необходимой точности можно поменять на другое).

```
public double getFunctionValue(double x)
 {
    double epcilon=1e-9;
    double leftX=points[0].getX();
    double rightX=points[pointsCount-1].getX();
    if (x < leftX|| x > rightX)
        return Double.NaN;
    if(Math.abs(x-leftX)<epcilon)
    {
        return points[0].getY();
    }
    if(Math.abs(x-rightX)<epcilon)
    {
        return points[pointsCount-1].getY();
    }
    int i = 0;
    double value=0;
    for (i=0;x > points[i].getX();++i)
    {
    value = points[i].getY() + (points[i + 1].getY() - points[i].getY()) * (x - points[i].getX()) / (points[i + 1].getX() - points[i].getX());
    }
    return value;

}
```

Рисунок 5 - Метод «getFunctionValue»

Задание 5

Создал 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 У), каждый из методов имеет проверку для индекса, чтобы он не выходил за границы сеточной функции.

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

public FunctionPoint getPoint(int index) {
    if(index<0||index>=pointsCount)
    {
        return null;
    }
    return points[index];
}

public void setPoint(int index, FunctionPoint point) {
    if (index < 0 || index >= pointsCount)
        return;
    points[index] = new FunctionPoint(point);
}

public double getPointX(int index) {

    if (index < 0 || index >= pointsCount)
        return Double.NaN;
    return points[index].getX();
}
```

```
public void setPointX(int index, double x) {
    if (index < 0 || index > pointsCount)
        return;
    points[index].setX(x);

}

public double getPointY(int index) {
    if (index < 0 || index > pointsCount)
        return Double.NaN;
    return points[index].getY();
}

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

Рисунок 6 - Методы получения значений точки с помощью входящего индекса

Задание 6

Создал два метода, «deletePoint» позволяет удалить точку из массива, имеет в себе проверку входящего индекса, массив будет копироваться и смещаться влево, «addPoint» позволяет добавить точку в массив.

В методе «deletePoint» реализованы проверки:

1. Выходит ли индекс за пределы массива;

2. Если индекс совпадает с левой границей, то мы уменьшаем «pointsCount» на 1, затем передаём в правую границу null.

В остальных случаях используется System.arraycopy

System.arraycopy() — это быстрый встроенный метод Java для копирования элементов из одного массива в другой. Он очень часто используется для расширения массивов или копирования части массива.

Что передаётся в функцию:

1. Исходный массив;

2. Стартовая позиция в исходном массиве, с которой начинаем копировать;

3. Массив, в который копируем;

4. Стартовая позиция в целевом массиве, куда вставляем;

5. Количество элементов для копирования.

```
  public void deletePoint(int index){
    if (index < 0 || index >= pointsCount){
            return;
        }

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


        }

    public void addPoint(FunctionPoint point) {
        int i = 0;
        double epcilon=1e-9;

    while (i < pointsCount && Math.abs(point.getX()-points[i].getX())<epcilon) {
        i++;
    }

    // Если точка с таким же x уже есть, обновляем y
    if (i < pointsCount && Math.abs(point.getX()-points[i].getX())<epcilon) {
        setPointY(i, point.getY());
        return;
    }


    System.arraycopy(points, i, points, i + 1, pointsCount - i);
    points[i] = new FunctionPoint(point);
    pointsCount++;
}
```

Рисунок 7 - Методы «deletePoint» и «addPoint»

Задание 7

В main создал массив точек У, для которых создал сеточную(табулированную) функцию с помощью «TabulatedFunction», далее удалил одну точку стоящую на 1 месте в массиве, добавил точку с координатами X =0.1, Y=5, и нашел значение У для X = 0.3 с помощью линейной интерполяции, также были проверены методы «setPoint» и «getPoint»

```
public static void main(String[] args)
{
    double [] data = {5, 2, 8, 2, 1};
    FunctionPoint Point = new FunctionPoint(0.1, 5);
    TabulatedFunction Function = new TabulatedFunction(0.0,1.0, data);
    System.out.println("Input data:");
    for (int i = 0; i < Function.getPointsCount(); i++){
        System.out.println("X["+ i + "] = " + Function.getPointX(i) + " and Y["+ i + "] = " + Function.getPointY(i));
    }
    Function.deletePoint(4);
    System.out.println("\n");
    System.out.println("As a result of deleting:");
    for (int i = 0; i < Function.getPointsCount(); i++){
        System.out.println("X["+ i + "] = " + Function.getPointX(i) + " and Y["+ i + "] = " + Function.getPointY(i));
    }
    Function.addPoint(Point);
    System.out.println("\n");
    System.out.println("As a result of adding:");
    for (int i = 0; i < Function.getPointsCount(); i++){
        System.out.println("X["+ i + "] = " + Function.getPointX(i) + " and Y["+ i + "] = " + Function.getPointY(i));
    }
    System.out.println("\n");
    System.out.println("Function value in the given X:0.3  Y = " + Function.getFunctionValue(0.3));

    System.out.println("\n");
    System.out.println("As a result of Setting and getting:");
    for(int i=0; i<Function.getPointsCount();i++)
    {
        Function.setPointX(i,i);
        Function.setPointY(i,i);
    }

    for(int i=0; i<Function.getPointsCount();i++)
    {
        System.out.println("X = " + Function.getPointX(i) + " and Y = " + Function.getPointY(i));

    }
```

Рисунок 8 - Метод «main»

```
Input data:
X[0] = 0.0 and Y[0] = 5.0
X[1] = 0.25 and Y[1] = 2.0
X[2] = 0.5 and Y[2] = 8.0
X[3] = 0.75 and Y[3] = 2.0
X[4] = 1.0 and Y[4] = 1.0


As a result of deleting:
X[0] = 0.0 and Y[0] = 5.0
X[1] = 0.25 and Y[1] = 2.0
X[2] = 0.5 and Y[2] = 8.0
X[3] = 0.75 and Y[3] = 2.0


As a result of adding:
X[0] = 0.1 and Y[0] = 5.0
X[1] = 0.0 and Y[1] = 5.0
X[2] = 0.25 and Y[2] = 2.0
X[3] = 0.5 and Y[3] = 8.0
X[4] = 0.75 and Y[4] = 2.0


Function value in the given X:0.3  Y = 3.1999999999999997


As a result of Setting and getting:
X = 0.0 and Y = 0.0
X = 1.0 and Y = 1.0
X = 2.0 and Y = 2.0
X = 3.0 and Y = 3.0
X = 4.0 and Y = 4.0
```

Рисунок 9 - Результат