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 Горейнов Д.В

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание №1

Создал пакет functions.

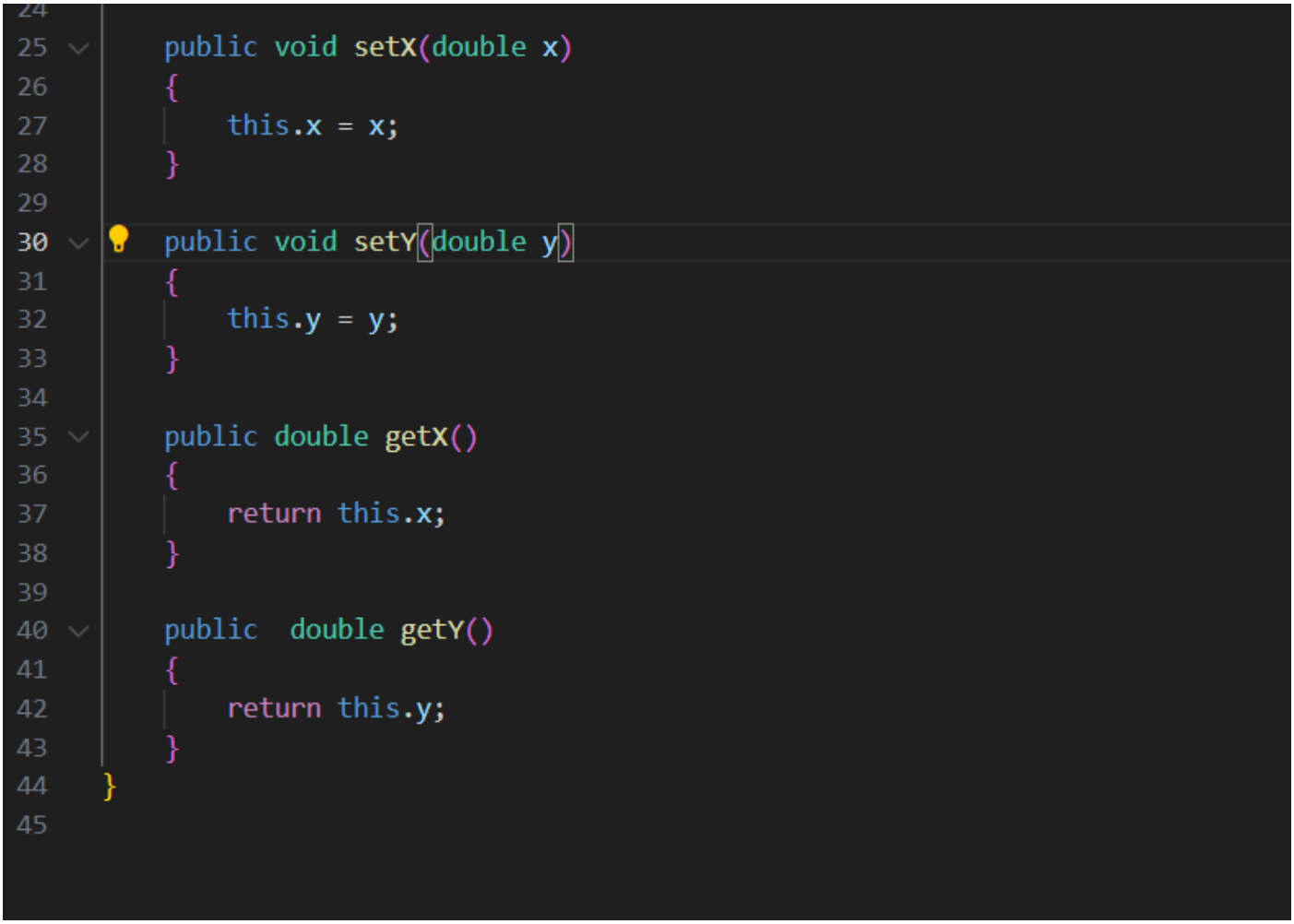
Задание №2

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

```java
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;
    }
```

фото 1

Также были созданы два геттер и сеттер метода для x и y, метод set должен задавать новые значения для точки, get выводи значения точки



```
    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;
    }
}
```

фото 2

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(points) содержащих координаты x с шагом step, а каждая координата y принимает стандартное значение равное 0.

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

```java
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]);
    }
}
```

фото 3

Задание №4

Создал два метода для возвращения левой и правой границ.

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

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

фото 4

Метод getFunctionValue позволяет узнать значение y в заданной точке x с помощью формулы для линейной интерполяции.

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

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)

```java
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;

}
```

фото 5

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

```java
55
56      public int getPointsCount() {
57          return pointsCount;
58      }
59
60      public FunctionPoint getPoint(int index) {
61          if(index<0||index>=pointsCount)
62          {
63              return null;
64          }
65          return points[index];
66      }
67
68      public void setPoint(int index, FunctionPoint point) {
69          if (index < 0 || index >= pointsCount)
70              return;
71          points[index] = new FunctionPoint(point);
72          leftX = points[0].getX();
73          rightX = points[pointsCount - 1].getX();
74      }
75
76      public double getPointX(int index) {
77
78          if (index < 0 || index >= pointsCount)
79              return Double.NaN;
80          return points[index].getX();
81      }
82
```

фото 6

```
84      public void setPointX(int index, double x) {
85          if (index < 0 || index > pointsCount)
86              return;
87          points[index].setX(x);
88          leftX = points[0].getX();
89          rightX = points[pointsCount - 1].getX();
90      }
91
92      public double getPointY(int index) {
93          if (index < 0 || index > pointsCount)
94              return Double.NaN;
95          return points[index].getY();
96      }
97
98      public void setPointY(int index, double y) {
99          points[index].setY(y);
100     }
101
```

Фото 7

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

```java
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;
    }


    }
```

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

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


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

}
```

фото 8 и 9

Задание №7

В main создал массив точек y, 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 y для X = 0.3 с помощью линейной интерполяции, также были проверенны методы setPoint и getPoint

```java
import functions.FunctionPoint;
import functions.TabulatedFunction;

public class Main {

    Run | Debug
    public static void main(String[] args)
    {
        double [] data = {2, 3, 6, 8, 4};
        FunctionPoint Point = new FunctionPoint(x:0.1, y:5);
        TabulatedFunction Function = new TabulatedFunction(leftX:0.0,rightX: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(index: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(x: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));

        }



    }

}
```

фото 10

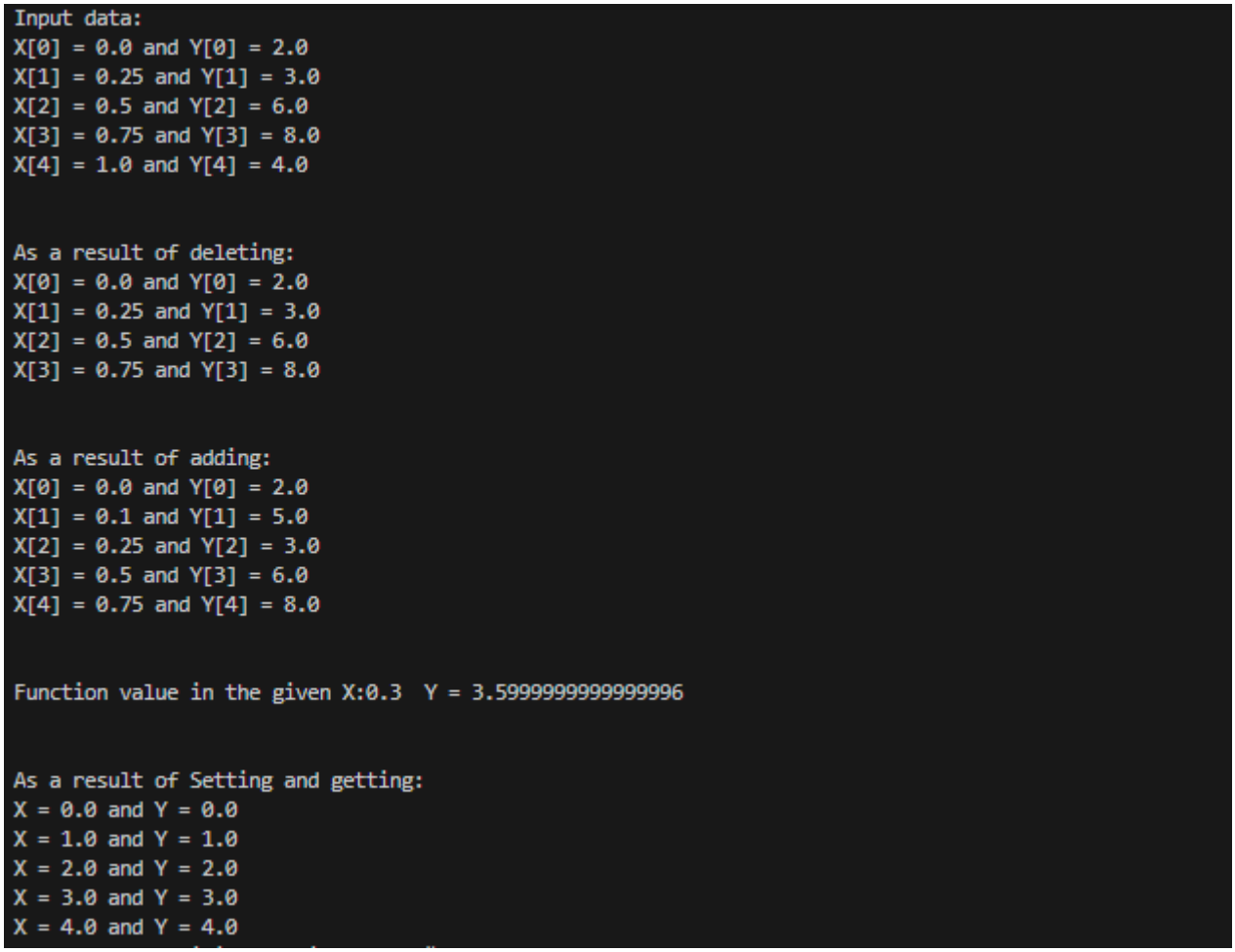

Input data:
X[0] = 0.0 and Y[0] = 2.0
X[1] = 0.25 and Y[1] = 3.0
X[2] = 0.5 and Y[2] = 6.0
X[3] = 0.75 and Y[3] = 8.0
X[4] = 1.0 and Y[4] = 4.0
As a result of deleting:
X[0] = 0.0 and Y[0] = 2.0
X[1] = 0.25 and Y[1] = 3.0
X[2] = 0.5 and Y[2] = 6.0
X[3] = 0.75 and Y[3] = 8.0
As a result of adding:
X[0] = 0.0 and Y[0] = 2.0
X[1] = 0.1 and Y[1] = 5.0
X[2] = 0.25 and Y[2] = 3.0
X[3] = 0.5 and Y[3] = 6.0
X[4] = 0.75 and Y[4] = 8.0
Function value in the given X:0.3 Y = 3.5999999999999996
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

фото 11