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.

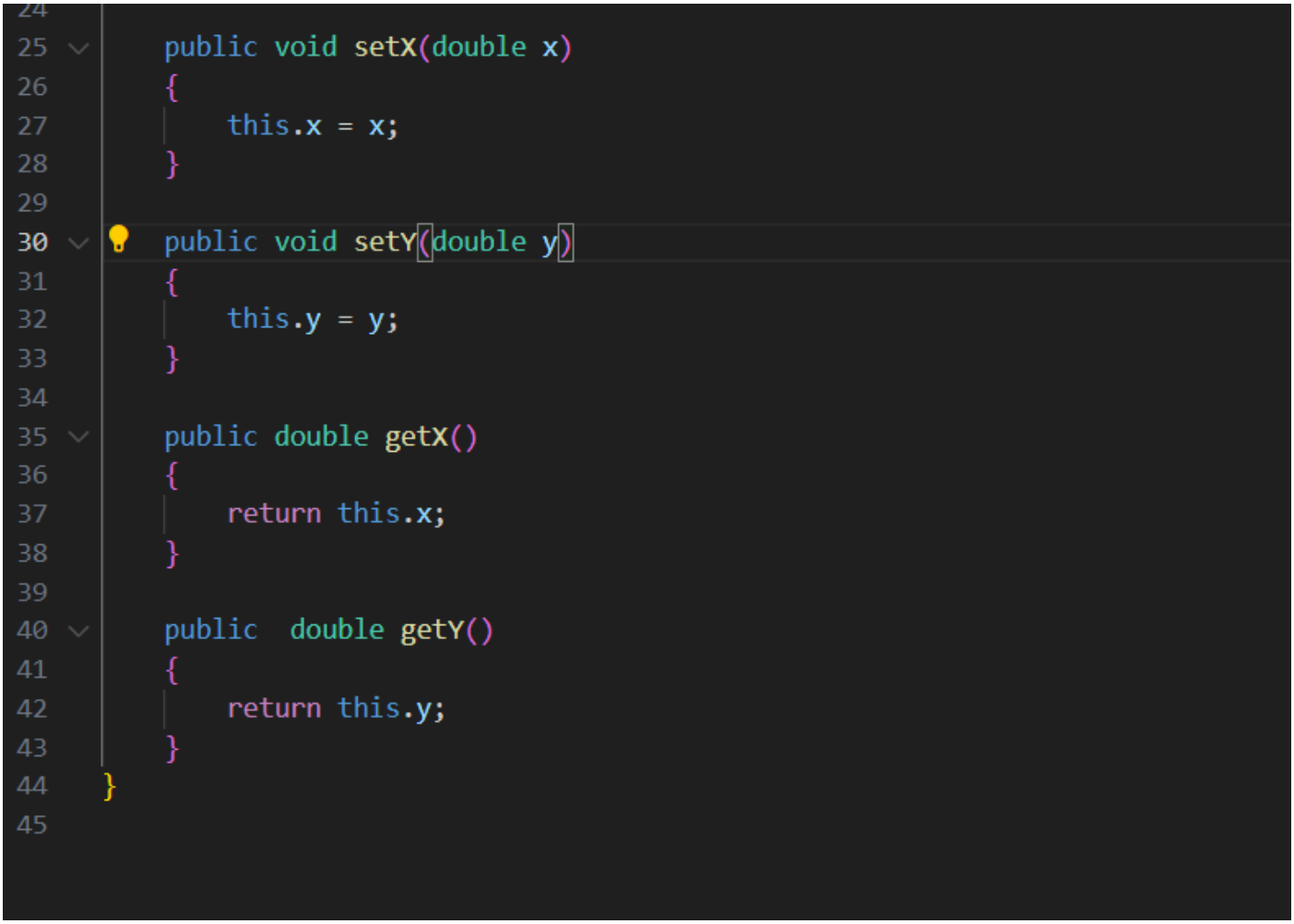
Задание №2

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.



```
    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 (если задается количество точек и если задается массив точек по оси y).

```java
package functions;
public class TabulatedFunction {
    private double leftX, rightX;
    private FunctionPoint points[];
    private int pointsCount;

    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, y:0);
        }

    }

    public TabulatedFunction(double leftX, double rightX, double[] values) {
        this.leftX = leftX;
        this.rightX = rightX;
        this.pointsCount = values.length;
        this.points = new FunctionPoint[this.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 leftX;
    }

    public double getRightDomainBorder() {
        return rightX;
    }

```

фото 4

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

Цикл while будет продолжаться пока входное значение x больше, чем значение x на позиции i из табулированной функции.

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

```
public double getFunctionValue(double x) {
    if (x < leftX || x > rightX)
        return Double.NaN;
    if(x==leftX)
    {
        return points[0].getY();
    }
    if(x==rightX)
    {
        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, ну или не был меньше 0(оранжевые прямоугольники). Также в методах для изменения координат, есть обновление границ(зеленые прямоугольники).

```java
    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);
        leftX = points[0].getX();
        rightX = points[pointsCount - 1].getX();
    }

    public double getPointX(int index) {

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

```

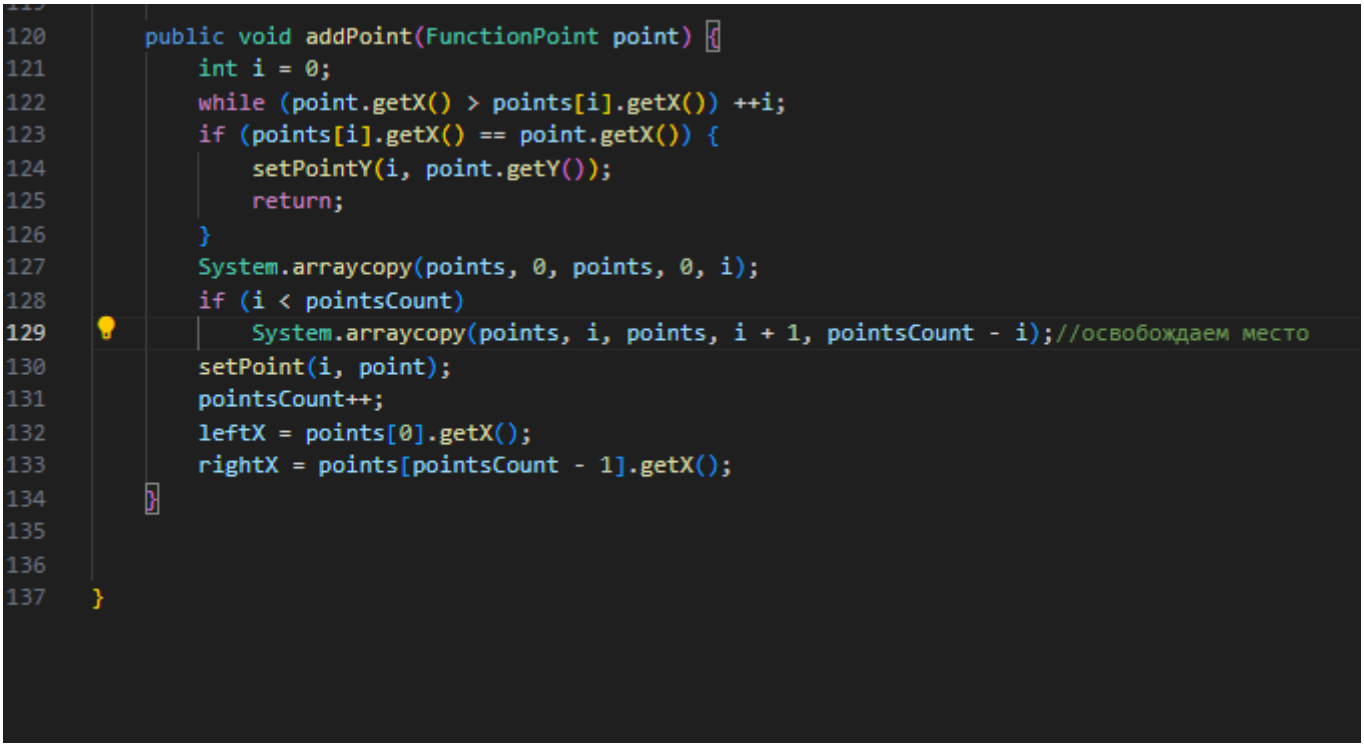
фото 6

```java
    public void setPointX(int index, double x) {
        if (index < 0 || index > pointsCount)
            return;
        points[index].setX(x);
        leftX = points[0].getX();
        rightX = points[pointsCount - 1].getX();
    }

    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);
    }

```

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.



```
120     public void addPoint(FunctionPoint point) {
121         int i = 0;
122         while (point.getX() > points[i].getX()) ++i;
123         if (points[i].getX() == point.getX()) {
124             setPointY(i, point.getY());
125             return;
126         }
127         System.arraycopy(points, 0, points, 0, i);
128         if (i < pointsCount)
129             System.arraycopy(points, i, points, i + 1, pointsCount - i);//освобождаем место
130         setPoint(i, point);
131         pointsCount++;
132         leftX = points[0].getX();
133         rightX = points[pointsCount - 1].getX();
134     }
135
136
137 }
```

фото 8

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 =7, Y=5, и нашел значение y для X = 6,5 с помощью линейной интерполяции.

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

public class Main {

    Run | Debug
    public static void main(String[] args)
    {
        double [] data = {1, 3, 6, 8, 4};
        FunctionPoint Point = new FunctionPoint(x:7, y:5);
        TabulatedFunction Function = new TabulatedFunction(leftX:0.0,rightX:10.0, 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:0);
        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  Y = " + Function.getFunctionValue(x:7.5));
    }

}
```

фото 9

```
Input data:
X[0] = 0.0 and Y[0] = 1.0
X[1] = 2.5 and Y[1] = 3.0
X[2] = 5.0 and Y[2] = 6.0
X[3] = 7.5 and Y[3] = 8.0
X[4] = 10.0 and Y[4] = 4.0


As a result of deleting:
X[0] = 2.5 and Y[0] = 3.0
X[1] = 5.0 and Y[1] = 6.0
X[2] = 7.5 and Y[2] = 8.0
X[3] = 10.0 and Y[3] = 4.0


As a result of adding:
X[0] = 2.5 and Y[0] = 3.0
X[1] = 5.0 and Y[1] = 6.0
X[2] = 7.0 and Y[2] = 5.0
X[3] = 7.5 and Y[3] = 8.0
X[4] = 10.0 and Y[4] = 4.0


Function value in the given X:6  Y = 5.5
```

фото 10