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 Волков Д.Н.

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(){
        x = 0;
        y = 0;
    }
```

Скрин 1

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

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

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

    public double getX(){
        return x;
    }

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

Скрин 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).

```
package functions;

public class TabulatedFunction{
    private FunctionPoint[] points;
    private double leftX;
    private double rightX;
    private int pointsCount;

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

Скрин 3

P.S Нужно будет добавить проверки :f

Задание №4

Создал два метода для возвращения левой и правой границ.

```
public double getLeftDomainBorder(){
    return leftX;
}

public double getRightDomainBorder(){
    return rightX;
}
```

Скрин 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;
    }
    else{
        int i = 0;
        double value;
        while (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;
    }
}
```

Скрин 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).

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

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

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;
    }
    else{
        return points[index].getX();
    }
}
```

Скрин 6

```
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;
    }
    else{
        return points[index].getY();
    }
}

public void setPointY(int index, double y){
    if (index < 0 || index >= pointsCount){
        return;
    }
    points[index].setY(y);
}
```

Скрин 7

Задание №6

Создал 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. Снова есть обновление границ после внесенных изменений, addPoint позволяет добавить точку в массив.

```
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;
    }

    if (pointsCount == 0){
    leftX = Double.NaN;
    rightX = Double.NaN;
    }
    else{
        leftX = points[0].getX();
        rightX = points[pointsCount - 1].getX();
    }
}

public void addPoint(FunctionPoint point){
    int i = 0;
    while (point.getX() > points[i].getX()) ++i;
    if(points[i].getX() == point.getX()){
        setPointY(i, point.getY());
        return;
    }
    System.arraycopy(points, srcPos:0, points, destPos:0, i);
    if(i < pointsCount)
        System.arraycopy(points, i, points, i+1, pointsCount - i);
    setPoint (i, point);
    pointsCount++;
    leftX = points[0].getX();
    rightX = points[pointsCount - 1].getX();
}
```

Скрин 8

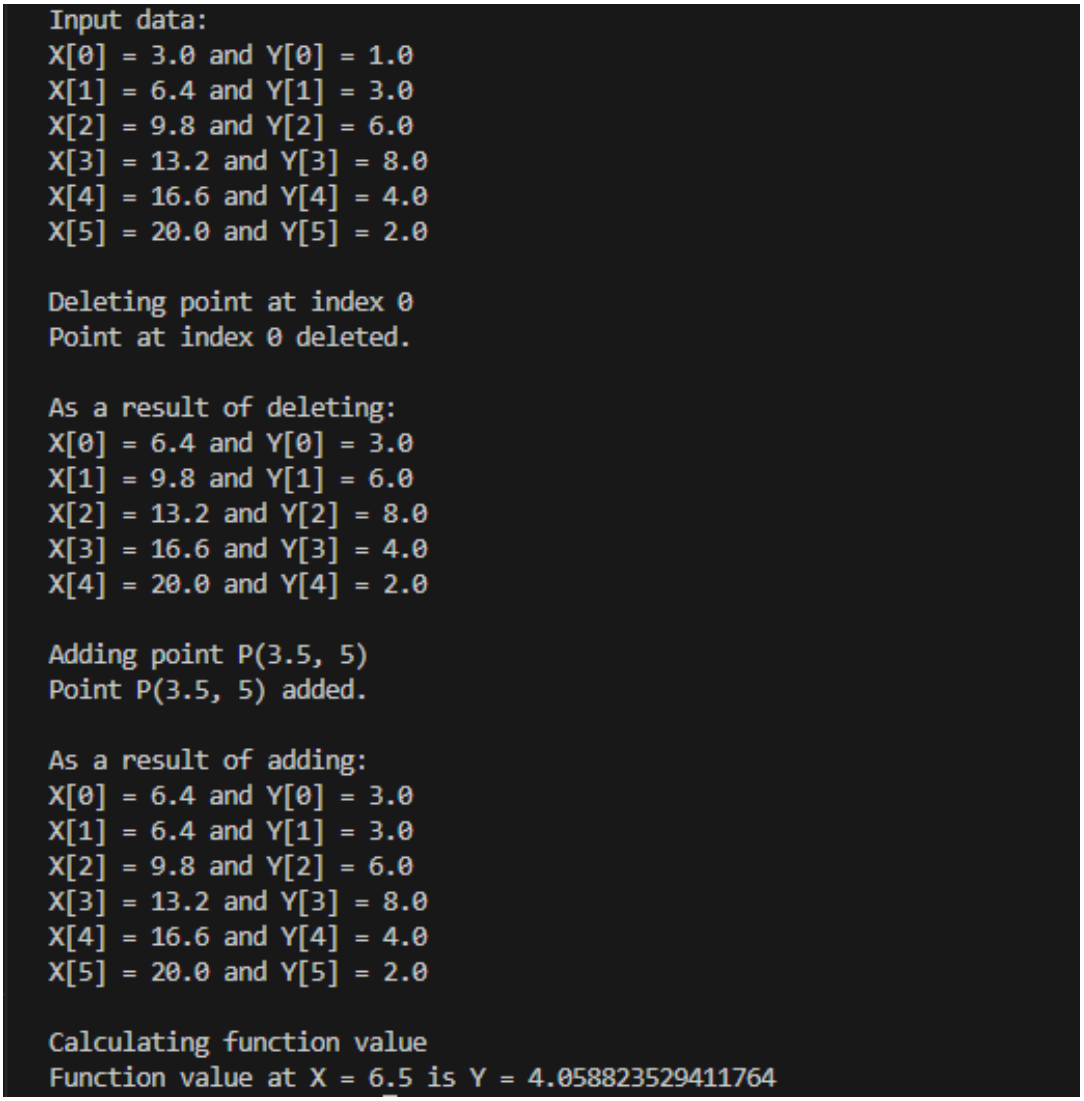
Задание №7

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 =3.5, Y=5, и нашел значение y для X = 6,5 с помощью линейной интерполяции.

```
import functions.*;

public class Main2 {

    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        double[] data = {1, 3, 6, 8, 4, 2};
        FunctionPoint P = new FunctionPoint(x:3.5, y:5);

        TabulatedFunction Function = new TabulatedFunction(leftX:3, rightX:20, data);

        System.out.println(x:"Input data:");
        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(x:"\nDeleting point at index 0 ");
        if (Function.getPointsCount() > 0) {
            Function.deletePoint(index:0);
            System.out.println(x:"Point at index 0 deleted.");
        }
        else {
            System.out.println(x:"Cannot delete, function has no points.");
        }

        System.out.println(x:"\nAs 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));
        }

        System.out.println(x:"\nAdding point P(3.5, 5)");
        Function.addPoint(P);
        System.out.println(x:"Point P(3.5, 5) added.");

        System.out.println(x:"\nAs 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(x:"\nCalculating function value");
        double ValueX = 6.5;
        double ValueY = Function.getFunctionValue(ValueX);
        System.out.println("Function value at X = " + ValueX + " is Y = " + ValueY);
    }
}
```

Скрин 9


Input data:
X[0] = 3.0 and Y[0] = 1.0
X[1] = 6.4 and Y[1] = 3.0
X[2] = 9.8 and Y[2] = 6.0
X[3] = 13.2 and Y[3] = 8.0
X[4] = 16.6 and Y[4] = 4.0
X[5] = 20.0 and Y[5] = 2.0
Deleting point at index 0
Point at index 0 deleted.
As a result of deleting:
X[0] = 6.4 and Y[0] = 3.0
X[1] = 9.8 and Y[1] = 6.0
X[2] = 13.2 and Y[2] = 8.0
X[3] = 16.6 and Y[3] = 4.0
X[4] = 20.0 and Y[4] = 2.0
Adding point P(3.5, 5)
Point P(3.5, 5) added.
As a result of adding:
X[0] = 6.4 and Y[0] = 3.0
X[1] = 6.4 and Y[1] = 3.0
X[2] = 9.8 and Y[2] = 6.0
X[3] = 13.2 and Y[3] = 8.0
X[4] = 16.6 and Y[4] = 4.0
X[5] = 20.0 and Y[5] = 2.0
Calculating function value
Function value at X = 6.5 is Y = 4.058823529411764

Скрин 10