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 Ветров И.Р.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание 2

Создал класс FunctionPoint, с приватными полями x и y, также реализовал сеттер и геттор методы для x и y.

```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, y: 0);
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

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

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

Скрин 1.

Задание 3

Был создан класс TabulatedFunction, в котором был реализован массив типа FunctionPoint. Также в классе были реализованы конструкторы.

```
public class TabulatedFunction {
    private static final int DEFAULT_CAPACITY = 16;
    private FunctionPoint[] points;
    private int pointsCount;

    public TabulatedFunction(double leftX, double rightX, int pointsCount) {
        if (pointsCount < 2) {
            pointsCount = 2;
        }

        initArrays(Math.max(pointsCount * 2, DEFAULT_CAPACITY));
        this.pointsCount = pointsCount;

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

    public TabulatedFunction(double leftX, double rightX, double[] values) {
        int count = values.length;
        if (count < 2) {
            count = 2;
        }

        initArrays(Math.max(count * 2, DEFAULT_CAPACITY));
        this.pointsCount = count;

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

    private void initArrays(int capacity) {
        points = new FunctionPoint[capacity];
        for (int i = 0; i < capacity; i++) {
            points[i] = new FunctionPoint();
        }
    }
```

Скрин 2.

Задание 4

В классе TabulatedFunction были описаны методы возвращающие значение левой и правой границы области определения табулированной функции(double getleftDomainBorder() и double getrightDomainBorder() соответственно).

```
46      public double getLeftDomainBorder() {
47          return points[0].getX();
48      }
49
50      public double getRightDomainBorder() {
51          return points[pointsCount - 1].getX();
52      }
```

Скрин 3.

Также метод вычисление значения функции в точке X с помощью линейной интерполяции.

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

    for (int i = 0; i < pointsCount - 1; i++) {
        double x1 = points[i].getX();
        double x2 = points[i + 1].getX();

        if (x >= x1 && x <= x2) {
            if (x == x1) return points[i].getY();
            if (x == x2) return points[i + 1].getY();

            double y1 = points[i].getY();
            double y2 = points[i + 1].getY();
            return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
        }
    }

    return Double.NaN;
}
```

Скрин 4.

Задание 5

Были реализованы методы возвращающие количество точек и возвращающие копию точек, соответствующие переданному индексу(int getPointsCount и getPoint(int index) соответственно). Также метод заменяющий точку с заданным индексом на новую(void setpoint) и два сеттер и геттор метода возвращающие и устанавливающие новые значения X и Y.

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

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

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

    if (index > 0 && point.getX() <= points[index - 1].getX()) {
        return;
    }
    if (index < pointsCount - 1 && point.getX() >= points[index + 1].getX()) {
        return;
    }

    points[index].setX(point.getX());
    points[index].setY(point.getY());
}

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

Скрин 5.

```java
public void setPointX(int index, double x) {
    if (index < 0 || index >= pointsCount) {
        return;
    }

    if (index > 0 && x <= points[index - 1].getX()) {
        return;
    }
    if (index < pointsCount - 1 && x >= points[index + 1].getX()) {
        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) {
    if (index < 0 || index >= pointsCount) {
        return;
    }
    points[index].setY(y);
}
```

Скрин 6.

Задание 6

Были созданы методы добавляющие и удаляющие точки табулированной функции.

```
public void deletePoint(int index) {
    if (index < 0 || index >= pointsCount) {
        return;
    }
    if (pointsCount <= 2) {
    }
    if (index < pointsCount - 1) {
        System.arraycopy(points, index + 1, points, index, pointsCount - index - 1);
    }
    points[pointsCount - 1] = new FunctionPoint();
    pointsCount--;
}
public void addPoint(FunctionPoint point) {
    int insertIndex = 0;
    while (insertIndex < pointsCount && point.getX() > points[insertIndex].getX()) {
        insertIndex++;
    }
    if (insertIndex < pointsCount &&
        Math.abs(point.getX() - points[insertIndex].getX()) < 1e-10) {
        return;
    }
    if (pointsCount == points.length) {
        int newCapacity = points.length * 2;
        FunctionPoint[] newArray = new FunctionPoint[newCapacity];
        System.arraycopy(points, srcPos: 0, newArray, destPos: 0, pointsCount);
        for (int i = pointsCount; i < newCapacity; i++) {
            newArray[i] = new FunctionPoint();
        }
        points = newArray;
    }
    if (insertIndex < pointsCount) {
        System.arraycopy(points, insertIndex, points, insertIndex + 1, pointsCount - insertIndex);
    }
    points[insertIndex] = new FunctionPoint(point);
    pointsCount++;
}
```

Скрин 7.

Задание 7

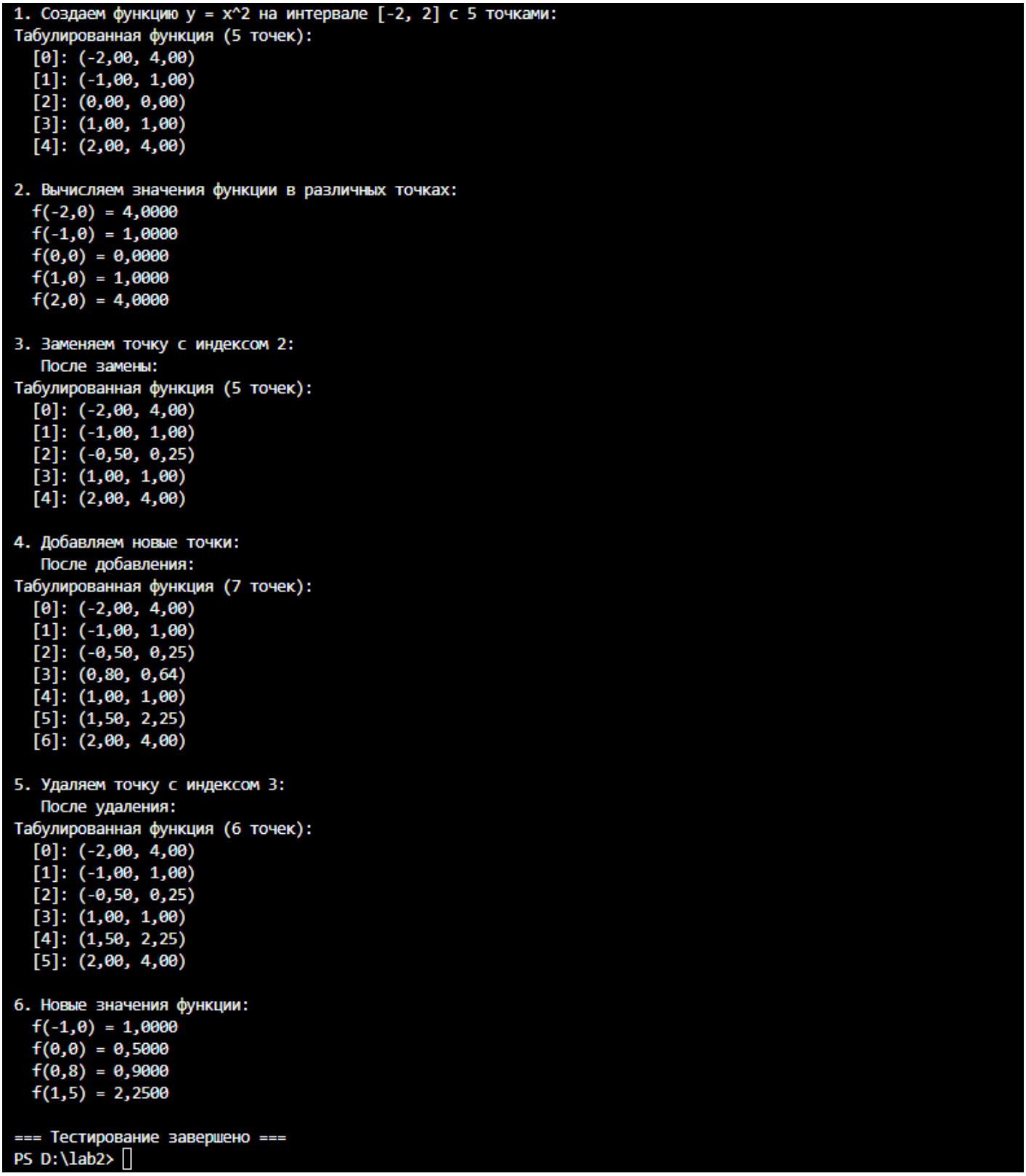
Был создан класс Main вне пакета functions для проверки работы написанных классов с функцией y=x^2.

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

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "=== Тестирование табулированной функции ===\n");
        System.out.println(x: "1. Создаем функцию y = x^2 на интервале [-2, 2] с 5 точками:");
        TabulatedFunction parabola = new TabulatedFunction(-2, rightX: 2, pointsCount: 5);
        for (int i = 0; i < parabola.getPointsCount(); i++) {
            double x = parabola.getPointX(i);
            parabola.setPointY(i, x * x);
        }
        parabola.printPoints();
        System.out.println(x: "\n2. Вычисляем значения функции в различных точках:");
        double[] testPoints = {-2,-1,0,1,2};
        for (double x : testPoints) {
            double y = parabola.getFunctionValue(x);
            System.out.printf(format: "  f(%.1f) = ", x);
            if (Double.isNaN(y)) {
                System.out.println(x: "не определено (вне области определения)");
            } else {
                System.out.printf(format: "%.4f%n", y);
            }
        }
        System.out.println(x: "\n3. Заменяем точку с индексом 2:");
        FunctionPoint newPoint = new FunctionPoint(-0.5, y: 0.25);
        parabola.setPoint(index: 2, newPoint);
        System.out.println(x: "   После замены:");
        parabola.printPoints();
        System.out.println(x: "\n4. Добавляем новые точки:");
        parabola.addPoint(new FunctionPoint(x: 1.5, y: 2.25));
        parabola.addPoint(new FunctionPoint(x: 0.8, y: 0.64));
        System.out.println(x: "   После добавления:");
        parabola.printPoints();
        System.out.println(x: "\n5. Удаляем точку с индексом 3:");
        parabola.deletePoint(index: 3);
        System.out.println(x: "   После удаления:");
        parabola.printPoints();
        System.out.println(x: "\n6. Новые значения функции:");
        for (double x : new double[]{-1, 0, 0.8, 1.5}) {
            double y = parabola.getFunctionValue(x);
            System.out.printf(format: "  f(%.1f) = %.4f%n", x, y);
        }

        System.out.println(x: "\n=== Тестирование завершено ===");
    }
}
```

Скрин 8.


1. Создаем функцию y = x^2 на интервале [-2, 2] с 5 точками:
Табулированная функция (5 точек):
[0]: (-2,00, 4,00)
[1]: (-1,00, 1,00)
[2]: (0,00, 0,00)
[3]: (1,00, 1,00)
[4]: (2,00, 4,00)
2. Вычисляем значения функции в различных точках:
f(-2,0) = 4,0000
f(-1,0) = 1,0000
f(0,0) = 0,0000
f(1,0) = 1,0000
f(2,0) = 4,0000
3. Заменяем точку с индексом 2:
После замены:
Табулированная функция (5 точек):
[0]: (-2,00, 4,00)
[1]: (-1,00, 1,00)
[2]: (-0,50, 0,25)
[3]: (1,00, 1,00)
[4]: (2,00, 4,00)
4. Добавляем новые точки:
После добавления:
Табулированная функция (7 точек):
[0]: (-2,00, 4,00)
[1]: (-1,00, 1,00)
[2]: (-0,50, 0,25)
[3]: (0,80, 0,64)
[4]: (1,00, 1,00)
[5]: (1,50, 2,25)
[6]: (2,00, 4,00)
5. Удаляем точку с индексом 3:
После удаления:
Табулированная функция (6 точек):
[0]: (-2,00, 4,00)
[1]: (-1,00, 1,00)
[2]: (-0,50, 0,25)
[3]: (1,00, 1,00)
[4]: (1,50, 2,25)
[5]: (2,00, 4,00)
6. Новые значения функции:
f(-1,0) = 1,0000
f(0,0) = 0,5000
f(0,8) = 0,9000
f(1,5) = 2,2500
=== Тестирование завершено ===
PS D:\lab2>

Скрин 9.