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 Богданчиков М.А.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Цель лабораторной работы: Разработать набор классов для работы с функциями одной переменной, заданными в табличной форме.

Ход выполнения лабораторной работы

Задание 1: Создан пакет functions, в котором далее будут создаваться классы программы.

Задание 2: Создадим класс FunctionPoint, который содержит координаты X и Y точки. Класс имеет три конструктора: с параметрами, по умолчанию и конструктор, параметром которого является сам объект класса FunctionPoint. А также в классе есть геттеры и сеттеры для координат:

```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 = 0.0;
        this.y = 0.0;
    }
    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;
    }
}
```

Код 1

Задание 3:
Создан класс табулированной функции TabulatedFunction с двумя конструкторами, которые создают точки по равным интервалам на оси X. Они оба принимают по три параметра и в качестве первых двух параметров границы leftX – левая области определения и rightX – правая граница области определения, но один в качестве третьего параметра принимает количество точек pointsCount, а другой - массив значений функции values.

```
package functions;

public class TabulatedFunction {
    private FunctionPoint[] points;
    private int pointsCount;
    public TabulatedFunction(double leftX, double rightX, int pointsCount) {
        this.pointsCount = pointsCount;
        this.points = new FunctionPoint[pointsCount + 10];

        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, 0.0);
        }
    }
    public TabulatedFunction(double leftX, double rightX, double[] values) {

        this.pointsCount = values.length;
        this.points = new FunctionPoint[pointsCount + 10];

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

Код 2

Задание 4: в класс TabulatedFunction добавлены методы для работы с функцией: getLeftDomainBorder (возвращает значение левой границы области определения табулированной функции), getRightDomainBorder (возвращает значение правой границы области определения табулированной функции) и getFunctionValue (возвращает значение функции в точке x, если эта точка лежит в области определения функции). Также вписана погрешность для сравнений EPSILON = 1e-10.

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

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

    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() - EPSILON || x > getRightDomainBorder() + EPSILON) {
            return Double.NaN;
        }
        for (int i = 0; i < pointsCount; i++) {
            if (Math.abs(points[i].getX() - x) < EPSILON) {
                return points[i].getY();
            }
        }
        for (int i = 0; i < pointsCount - 1; i++) {
            double x1 = points[i].getX();
            double x2 = points[i + 1].getX();
            double y1 = points[i].getY();
            double y2 = points[i + 1].getY();
            return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
        }
    }
```

Код 3

Задание 5: в класс TabulatedFunction добавлены методы, необходимые для работы с точками табулированной функции: getPointsCount (возвращает число точек), getPoint (возвращает *копию* точки, соответствующей переданному индексу), setPoint (заменяет указанную точку табулированной функции на переданную), getPointX (возвращает значение абсциссы точки с указанным номером), setPointX (изменяет значение абсциссы точки с указанным номером), getPointY (возвращает значение ординаты точки с указанным номером), setPointY (изменяет значение ординаты точки с указанным номером).

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

public FunctionPoint getPoint(int index) {
    return new FunctionPoint(points[index]);
}

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

    double newX = point.getX();
    double leftBound = (index > 0) ? points[index - 1].getX() : -Double.MAX_VALUE;
    double rightBound = (index < pointsCount - 1) ? points[index + 1].getX() : Double.MAX_VALUE;

    if (newX > leftBound + EPSILON && newX < rightBound - EPSILON) {
        points[index] = new FunctionPoint(point);
    }
}
```

Код 4.1

```
public double getPointX(int index) {
    return points[index].getX();
}

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

    double leftBound = (index > 0) ? points[index - 1].getX() : -Double.MAX_VALUE;
    double rightBound = (index < pointsCount - 1) ? points[index + 1].getX() : Double.MAX_VALUE;

    if (x > leftBound + EPSILON && x < rightBound - EPSILON) {
        points[index].setX(x);
    }
}

public double getPointY(int index) {
    return points[index].getY();
}

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

Код 4.2

Задание 6: В классе TabulatedFunction добавлены методы, изменяющие количество точек табулированной функции: deletePoint (удаляет точку), addPoint (добавляет точку)

```
public void deletePoint(int index) {
    if (pointsCount <= 2 || index < 0 || index >= pointsCount) {
        return;
    }

    System.arraycopy(points, index + 1, points, index, pointsCount - index - 1);
    pointsCount--;
}

public void addPoint(FunctionPoint point) {
    if (pointsCount == points.length) {
        FunctionPoint[] newArray = new FunctionPoint[points.length * 2];
        System.arraycopy(points, 0, newArray, 0, pointsCount);
        points = newArray;
    }

    int insertIndex = 0;
    while (insertIndex < pointsCount && points[insertIndex].getX() < point.getX() - EPSILON) {
        insertIndex++;
    }

    if (insertIndex < pointsCount && Math.abs(points[insertIndex].getX() - point.getX()) < EPSILON) {
        points[insertIndex].setY(point.getY());
        return;
    }
    if (insertIndex < pointsCount) {
        System.arraycopy(points, insertIndex, points, insertIndex + 1, pointsCount - insertIndex);
    }
    points[insertIndex] = new FunctionPoint(point);
    pointsCount++;
}
```

Код 5

Задание 7: создан класс Main, с помощью которого будут проверяться работа написанных классов, и выведен результат его выполнения в консоль.

```
import functions.*;

public class Main {
    public static void main(String[] args) {
        System.out.println("=== Function test ===\n");

        int pointsCount = 5;
        TabulatedFunction parabola = new TabulatedFunction(-2, 2, pointsCount);

        for (int i = 0; i < parabola.getPointsCount(); i++) {
            double x = parabola.getPointX(i);
            parabola.setPointY(i, x*x);
        }

        System.out.println("y = x^2 on [-2, 2]");
        System.out.println("Points: " + parabola.getPointsCount());

        System.out.println("\nAll points:");
        for (int i = 0; i < parabola.getPointsCount(); i++) {
            System.out.printf("Point %d: x=%.2f, y=%.2f\n", i, parabola.getPointX(i), parabola.getPointY(i));
        }

        System.out.printf("\nDomain: [%.2f, %.2f]\n",
                parabola.getLeftDomainBorder(), parabola.getRightDomainBorder());
```

Код 6.1

```
        System.out.println("\nFunction values:");
        double[] testPoints = {-3, -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2, 3};
        for (double x : testPoints) {
            double y = parabola.getFunctionValue(x);
            if (Double.isNaN(y)) {
                System.out.printf("f(%.2f) = undefined\n", x);
            } else {
                System.out.printf("f(%.2f) = %.3f\n", x, y);
            }
        }

        System.out.println("\nChanging point:");
        int idx = 2;
        double newX = parabola.getPointX(idx);
        double newY = 10.0;
        System.out.printf("Change point %d to (%.2f, %.2f)\n",
                idx, newX, newY);
        parabola.setPoint(idx, new FunctionPoint(newX, newY));
        System.out.printf("f(%.2f) now = %.3f\n", newX, parabola.getFunctionValue(newX));

        System.out.println("\nTrying to change X coordinate:");
        int changeIdx = 1;
        double oldX = parabola.getPointX(changeIdx);
        double tryX = oldX + 0.3;
        System.out.printf("Try to change x of point %d from %.2f to %.2f\n", changeIdx, oldX, tryX);
        parabola.setPointX(changeIdx, tryX);
        double actualX = parabola.getPointX(changeIdx);
        if (Math.abs(tryX - actualX) < 1e-10) {
            System.out.printf("Changed, now x=%.2f\n", actualX);
        } else {
            System.out.printf("Not changed, x still %.2f\n", actualX);
        }
```

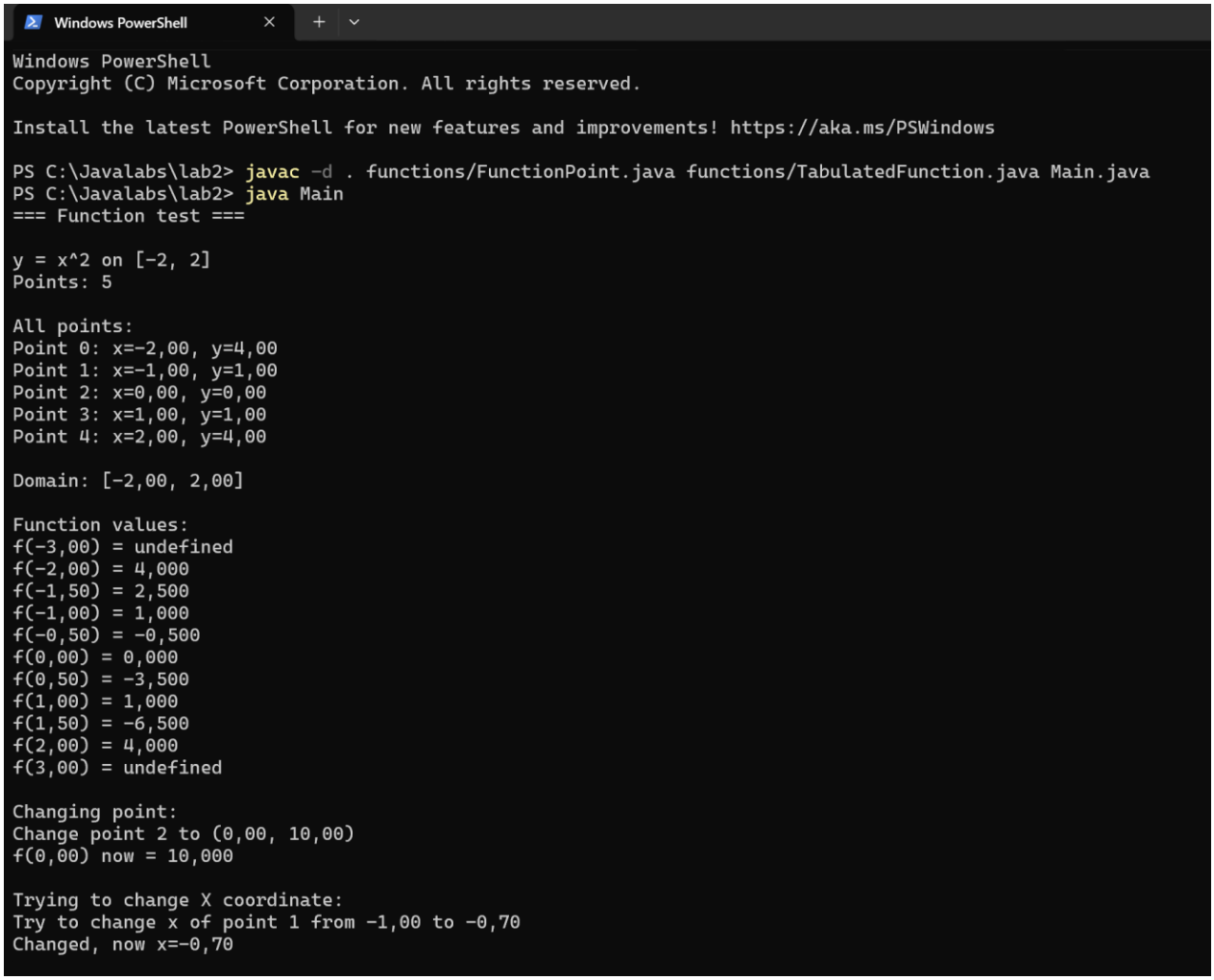
Код 6.2

```
System.out.println("\nChanging Y coordinate:");
int yIdx = 3;
double oldY = parabola.getPointY(yIdx);
double newYVal = 7.5;
System.out.printf("Change y of point %d from %.2f to %.2f\n", yIdx, oldY, newYVal);
parabola.setPointY(yIdx, newYVal);
System.out.printf("y of point %d now = %.2f\n", yIdx, parabola.getPointY(yIdx));

System.out.println("\nAdding point:");
FunctionPoint newPoint = new FunctionPoint(1.7, 3.0);
System.out.printf("Add (%.2f, %.2f)\n", newPoint.getX(), newPoint.getY());
parabola.addPoint(newPoint);
System.out.println("Points after add: " + parabola.getPointsCount());

System.out.println("\nPoints after adding:");
for (int i = 0; i < parabola.getPointsCount(); i++) {
    System.out.printf("Point %d: x=%.2f, y=%.2f\n", i, parabola.getPointX(i), parabola.getPointY(i));
}

System.out.printf("\nf(%.2f) = %.3f\n", newPoint.getX(), parabola.getFunctionValue(newPoint.getX()));

System.out.println("\nDeleting point:");
int delIdx = 2;
System.out.printf("Delete point %d (x=%.2f, y=%.2f)\n",
        delIdx, parabola.getPointX(delIdx), parabola.getPointY(delIdx));
parabola.deletePoint(delIdx);
System.out.println("Points after delete: " + parabola.getPointsCount());

System.out.println("\nRemaining points:");
for (int i = 0; i < parabola.getPointsCount(); i++) {
    System.out.printf("Point %d: x=%.2f, y=%.2f\n", i, parabola.getPointX(i), parabola.getPointY(i));
}
```

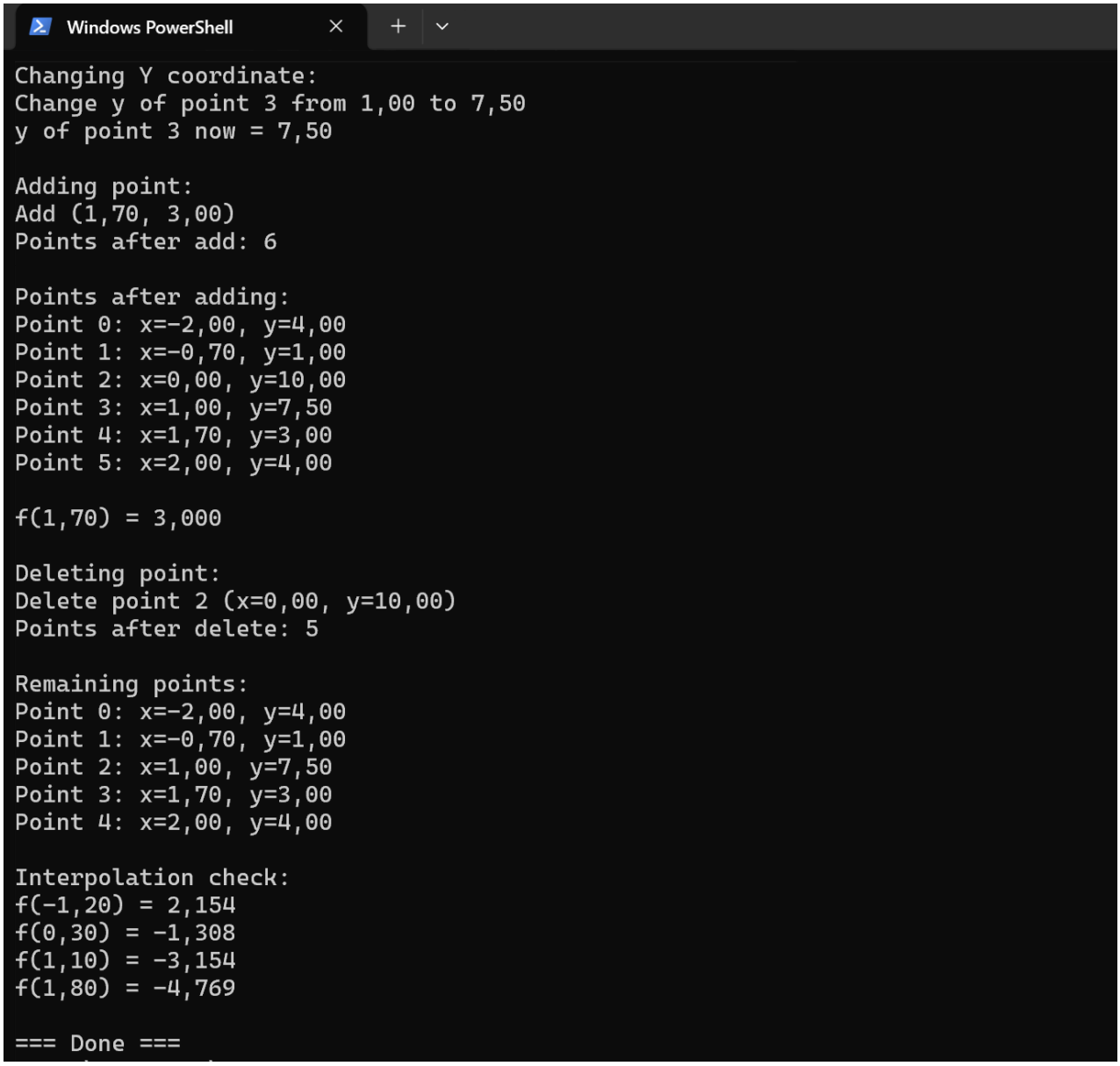
Код 6.3

```
    System.out.println("\nInterpolation check:");
    double[] interpPoints = {-1.2, 0.3, 1.1, 1.8};
    for (double x : interpPoints) {
        double y = parabola.getFunctionValue(x);
        if (Double.isNaN(y)) {
            System.out.printf("f(%.2f) = out of domain\n", x);
        } else {
            System.out.printf("f(%.2f) = %.3f\n", x, y);
        }
    }

    System.out.println("\n=== Done ===");
}
```

Код 6.4



Реализация 6.1



Реализация 6.2