# Лабораторная работа №2

**Заболотнов Николай Михайлович**

**6204-010302D**

# Task 1

Для начала создадим пакет function (рис 1).

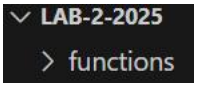


Рис 1

# Task 2

Создаем класс FunctionPoint с приватными полями x и y и с методами get…() и set…() для доступа к полям (рис 2).

```
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;
}
```

Рис 2

Для него напишем конструкторы (рис 3).

```
public FunctionPoint (double x, double y) {
    this.x = x;
    this.y = y;
}

public FunctionPoint (FunctionPoint point) {
    x = point.x;
    y = point.y;
}

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

Рис 3

# Task 3

Создаём класс TabulatedFunction с приватными полями: массивом точек и переменной количества точек, т. к. длина массива не обязана быть равна кол-ву точек. Напишем конструкторы этого класса (рис 4).

```
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, leftX += step)
        points[i] = new FunctionPoint(leftX, 0);
    PointsCount = pointsCount;
}

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

Рис 4

# Task 4

Добавим методы getLeftDomainBorder() и getRightDomainBorder() которые возвращают левую и правую границу области определения, т. к. точки в массиве расположены в порядке возрастания абсциссы точек, то это координаты x первой и последней почки массива (рис 5).

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

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

Рис 5

Напишем метод getFunctionValue(), который в случае если точка находится в области определения находит промежуток между координатами x соседних точек в, котором лежит аргумент, и ,подставляя точки в уравнение прямой, проходящей

через две точки, находит значение функции (рис 6). В противном случае неопределённость (поле NaN класса Double).

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

Рис 6

## Task 5

Реализуем методы: возвращающий количество точек, getPoint(), setPoint(), getPointX(), setPointX(), getPointY() и setPointY(), так чтобы при изменении точки и координаты x точка изменялась только в случае если координата x задаваемой точки лежит в интервале, определяемого значениями соседних точек табулированной функции (рис 7 и 8).

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

public void setPoint (int index, FunctionPoint point) {
    if ((index == 0 || index == PointsCount - 1)) {
        if (points[index].getX() != point.getX())
            return;
        points[index] = new FunctionPoint(point);
    }
    else
        if (point.getX() > points[index - 1].getX() && point.getX() < points[index + 1].getX())
            points[index] = new FunctionPoint(point);
}
```

Рис 7

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

public void setPointX (int index, double x) {
    if (index != 0 && index != PointsCount - 1 &&
     x > points[index - 1].getX() && x < points[index + 1].getX())
        points[index].setX(x);
}

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

public void setPointY (int index, double y) {
    points[index].setY(y);
}
```

Рис 8

## Task 6

В классе TabulatedFunction опишем методы deletePoint(), удаляющий элемент без создания нового массива, и addPoint(), добавляющий элемент создавая новый массив только в случае когда *кол-во точек + 1 > длины массива* (рис 9)

```
public void deletePoint (int index) {
    System.arraycopy(points, index + 1, points, index, PointsCount - index - 1);
    --PointsCount;
}

public void addPoint (FunctionPoint point) {
    int index = 0;
    if (point.getX() > points[PointsCount - 1].getX())
        index = PointsCount;
    else {
        while (point.getX() > points[index].getX()) {
            if (point.getX() == points[index].getX()) return;
            ++index;
        }
    }
    if (PointsCount + 1 > points.length) {
        FunctionPoint[] newPoints = new FunctionPoint[PointsCount * 2 + 1];
        System.arraycopy(points, 0, newPoints, 0, index);
        newPoints[index] = new FunctionPoint(point);
        System.arraycopy(points, index, newPoints, index + 1, PointsCount - index);
        points = newPoints;
    }
    else {
        System.arraycopy(points, index, points, index + 1, PointsCount - index);
        points[index] = new FunctionPoint(point);
    }
    ++PointsCount;
}
```

Рис 9

## Task 7

Вне пакета напишем класс Main. В методе main() создадим экземпляр класса TabulatedFunction и проверим написанный ранее методы (рис 10 и 11).

```
public class Main {
    Run main | Debug main
    public static void main(String[] args) {
        double[] values = {1, 6, 11, 16, 21, 26, 31, 36, 41, 46};
        TabulatedFunction f = new TabulatedFunction(5, 50, values);
        System.out.printf("левая граница области определения f: %.2f\n", f.getLeftDomainBorder());
        System.out.printf("правая граница области определения f: %.2f\n", f.getRightDomainBorder());
        System.out.println("Значения функции:");
        for (double i = 3; i < 63; i += 10) {
            System.out.printf("f(%.2f) = %.2f\n", i, f.getFunctionValue(i));
        }
        System.out.println();
        System.out.printf("Кол-во точек: %d\n", f.getPointsCount());
        System.out.println("Точки:");
        f.outClass();
        System.out.println();
        System.out.println("Точки после изменения:");
        f.setPoint(3, new FunctionPoint(50, 50));
        f.setPoint(5, new FunctionPoint(28, 50));
        f.outClass();
        System.out.println();
        System.out.println("Точки после изменения x:");
        f.setPointX(6, 100);
        f.setPointX(2, 18);
        f.outClass();
        System.out.println();
        System.out.println("Точки после изменения y:");
```

Рис 10

```
        f.setPointY(6, 100);
        f.outClass();
        System.out.println();
        System.out.printf("x элемента с индексом %d: %.2f\n", 3, f.getPointX(3));
        System.out.printf("y элемента с индексом %d: %.2f\n", 4, f.getPointY(4));
        System.out.println();
        System.out.println("Точки после добавления и удаления точек:");
        f.addPoint(new FunctionPoint(1,1));
        f.deletePoint(2);
        f.outClass();
    }
```

Рис 11

## Результат (рис 12, 13 и 14):

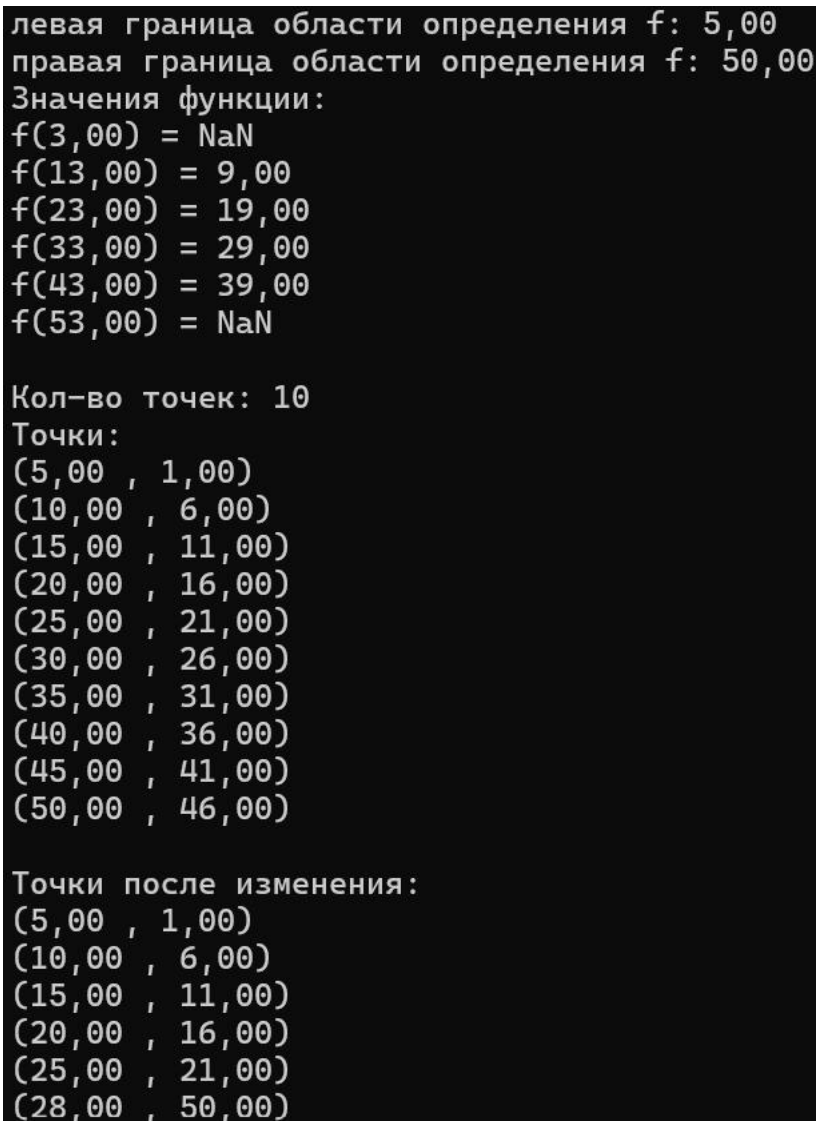
левая граница области определения f: 5,00
правая граница области определения f: 50,00
Значения функции:
f(3,00) = NaN
f(13,00) = 9,00
f(23,00) = 19,00
f(33,00) = 29,00
f(43,00) = 39,00
f(53,00) = NaN
Кол-во точек: 10
Точки:
(5,00 , 1,00)
(10,00 , 6,00)
(15,00 , 11,00)
(20,00 , 16,00)
(25,00 , 21,00)
(30,00 , 26,00)
(35,00 , 31,00)
(40,00 , 36,00)
(45,00 , 41,00)
(50,00 , 46,00)
Точки после изменения:
(5,00 , 1,00)
(10,00 , 6,00)
(15,00 , 11,00)
(20,00 , 16,00)
(25,00 , 21,00)
(28,00 , 50,00)

Рис 12

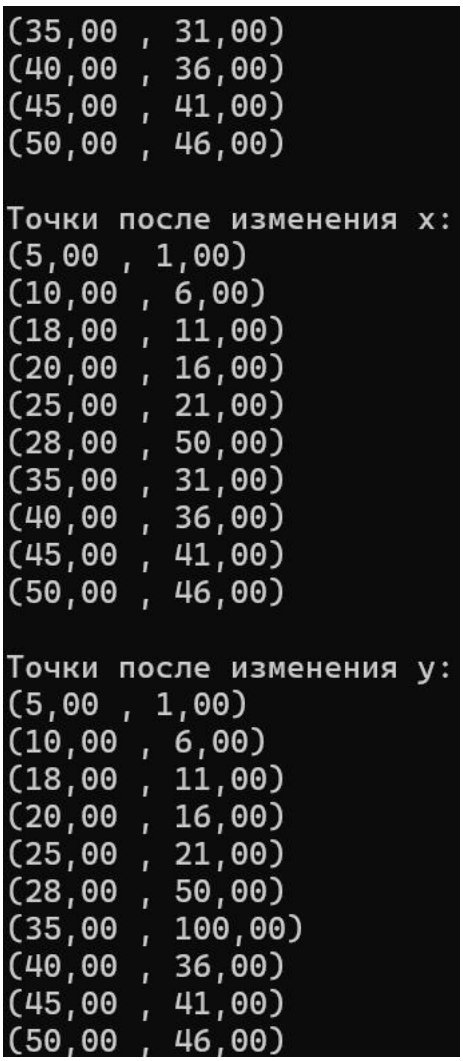
(35,00 , 31,00)
(40,00 , 36,00)
(45,00 , 41,00)
(50,00 , 46,00)
Точки после изменения x:
(5,00 , 1,00)
(10,00 , 6,00)
(18,00 , 11,00)
(20,00 , 16,00)
(25,00 , 21,00)
(28,00 , 50,00)
(35,00 , 31,00)
(40,00 , 36,00)
(45,00 , 41,00)
(50,00 , 46,00)
Точки после изменения y:
(5,00 , 1,00)
(10,00 , 6,00)
(18,00 , 11,00)
(20,00 , 16,00)
(25,00 , 21,00)
(28,00 , 50,00)
(35,00 , 100,00)
(40,00 , 36,00)
(45,00 , 41,00)
(50,00 , 46,00)

Рис 13

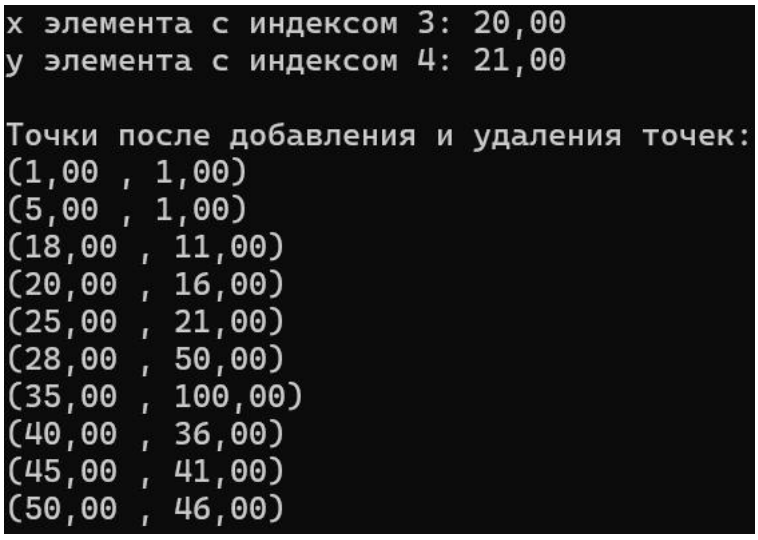
x элемента с индексом 3: 20,00
y элемента с индексом 4: 21,00
Точки после добавления и удаления точек:
(1,00 , 1,00)
(5,00 , 1,00)
(18,00 , 11,00)
(20,00 , 16,00)
(25,00 , 21,00)
(28,00 , 50,00)
(35,00 , 100,00)
(40,00 , 36,00)
(45,00 , 41,00)
(50,00 , 46,00)

Рис 14