

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

Институт информатики и кибернетики  
Кафедра технической кибернетики

Отчет по лабораторной работе №2  
Дисциплина: «ООП»

Тема «Базовые конструкции»

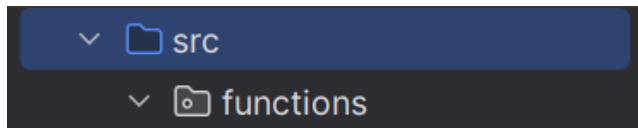
Выполнила: Степанова А.Д.  
Группа: 6201-120303D

Самара 2025

## Задание на лабораторную работу

### Задание 1

Я создала пакет functions



### Задание 2

В пакете functions создала класс FunctionPoint, объект которого должен описывать одну точку табулированной функции

```
① FunctionPoint.java × ② TabulatedFunction.java
1 package functions;
2
3 public class FunctionPoint { 12 usages
4
5     private double x; 4 usages
6     private double y; 4 usages
7
8     //Конструкторы
9     //Создает объект точки с заданными координатами
10    public FunctionPoint(double x, double y) { 5 usages
11        this.x = x;
12        this.y = y;
13    }
14
15    //Создает объект точки с теми же координатами, что и у указанной точки
16    @ public FunctionPoint(FunctionPoint point) { 2 usages
17        this.x = point.x;
18        this.y = point.y;
19    }
20
21    //Создает точку (0,0)
22    public FunctionPoint() { no usages
23        this(0, 0);
24    }
25
26    //Геттеры
27    public double getX() { 13 usages
28        return x;
29    }
30
31    public double getY() { 4 usages
32        return y;
33    }
34}
```

The code editor shows the 'FunctionPoint.java' file. The code defines a class 'FunctionPoint' with two private fields 'x' and 'y'. It has a constructor that takes 'x' and 'y' coordinates, a copy constructor that takes another 'FunctionPoint' object, and a default constructor that initializes the point at (0,0). It also includes two getter methods, 'getX()' and 'getY()'. Line numbers are visible on the left side of the code.

### Задание 3

В пакете functions создала класс TabulatedFunction, объект которого должен описывать табулированную функцию

```
1 package functions;
2
3 public class TabulatedFunction {
4     private FunctionPoint[] points;
5
6     //Конструкторы
7     //Создает объект табул. функции по заданным параметрам
8     public TabulatedFunction(double leftX, double rightX, int pointsCount) {
9
10         this.points = new FunctionPoint[pointsCount];
11
12         double step = (rightX - leftX) / (pointsCount - 1);
13
14         for (int i = 0; i < pointsCount; i++) {
15             double x = leftX + i * step;
16             points[i] = new FunctionPoint(x, y: 0);
17         }
18     }
19
20     //Создает объект табул. функции по заданным параметрам
21     public TabulatedFunction(double leftX, double rightX, double[] values) {
22
23         this.points = new FunctionPoint[values.length];
24
25         double step = (rightX - leftX) / (values.length - 1);
26
27         for (int i = 0; i < values.length; i++) {
28             double x = leftX + i * step;
29             points[i] = new FunctionPoint(x, values[i]);
30         }
31     }
32 }
```

## Задание 4

Кроме методов, которые требовались в задании, еще добавила методы сравнения чисел с плавающей точкой через эпсилон

```
37     public static final double EPSILON = 1e-10;  3 usages
38     //Сравнение чисел с плавающей точкой
39     //равенство
40     public static boolean doubleEq(double a, double b) {  4 usages
41         return Math.abs(a - b) < EPSILON;
42     }
43     //строго меньше
44     public static boolean doubleLess(double a, double b) {  2 usages
45         return a < b - EPSILON;
46     }
47     //строго больше
48     public static boolean doubleGreater(double a, double b) {  3 usages
49         return a > b + EPSILON;
50     }
51     //меньше или равно
52     public static boolean doubleLessOrEq(double a, double b) {  3 usages
53         return doubleLess(a,b) || doubleEq(a,b);
54     }
55     //больше или равно
56     public static boolean doubleGreaterOrEq(double a, double b) {  3 usages
57         return doubleGreater(a,b) || doubleEq(a,b);
```

```
59     public double getLeftDomainBorder() {  2 usages
60         return points[0].getX();
61     }
62     public double getRightDomainBorder() {  2 usages
63         return points[pointsCount - 1].getX();
64     }
65     //Возвращает значение функции в точке x, если точка лежит в обл. определения
66     public double getFunctionValue(double x) {  2 usages
67         if (doubleLess(x,getLeftDomainBorder()) || doubleGreater(x,getRightDomainBorder())) {
68             return Double.NaN;
69         }
70
71         for (int i = 0; i < pointsCount - 1; i++) {
72             if (doubleGreaterOrEq(x,points[i].getX()) && doubleLessOrEq(x,points[i + 1].getX())) {
73                 if (doubleEq(x,points[i].getX())) {
74                     return points[i].getY();
75                 }
76                 else if (doubleEq(x,points[i + 1].getX())) {
77                     return points[i + 1].getY();
78                 }
79             }
80         }
81     }
```

```
79             else {
80                 return interpolation(x, points[i], points[i + 1]);
81             }
82         }
83     }
84     return Double.NaN;
85 }
86
87 // Линейная интерполяция
88 @private double interpolation(double x, FunctionPoint p1, FunctionPoint p2) { 1 usage
89     return p1.getY() + (x - p1.getX()) * (p2.getY() - p1.getY()) / (p2.getX() - p1.getX());
90 }
```

## Задание 5

```
92 //Метод возвращает количество точек
93 > public int getPointsCount() { return pointsCount; }
94
95
96 //Метод возвращает копию точки, соответствующей данному индексу
97 > public FunctionPoint getPoint(int index) { return new FunctionPoint(points[index]); }
98
99
100 //Метод заменяет указанную точку на переданную
101 @public void setPoint(int index, FunctionPoint point) { 2 usages
102     double newX = point.getX();
103
104     if (index > 0) {
105         //если newX меньше левой границы
106         if (doubleLessOrEq(newX, points[index - 1].getX())) {
107             return;
108         }
109     }
110
111     if (index < pointsCount - 1) {
112         //если newX больше правой границы
113         if (doubleGreaterOrEq(newX, points[index + 1].getX())) {
114             return;
115         }
116     }
117 }
```

```
117     }
118     //если X лежит в нужном интервале, создаем новую точку
119     points[index] = new FunctionPoint(point);
120 }
121 //Метод возвращает значение абсциссы точки с указанным номером
122 public double getPointX(int index) { 20 usages
123     return points[index].getX();
124 }
125 //Метод возвращает значение ординаты точки с указанным номером
126 public double getPointY(int index) { 16 usages
127     return points[index].getY();
128 }
129 //Метод изменяет значение абсциссы точки с указанным номером
130 public void setPointX(int index, double x) { 2 usages
131     if (index > 0) {
132         //если x меньше левой границы
133         if (doubleLessOrEq(x, points[index - 1].getX())) {
134             return;
135         }
136     }
137     if (index < pointsCount - 1) {
138         if (index < pointsCount - 1) {
139             //если x больше правой границы
140             if (doubleGreaterOrEq(x, points[index + 1].getX())) {
141                 return;
142             }
143             //создаем новую точку с новым X и старым Y
144             points[index] = new FunctionPoint(x, getPointY(index));
145     }
146
147 //Метод изменяет значение ординаты точки с указанным номером
148 public void setPointY(int index, double y) { 3 usages
149     points[index] = new FunctionPoint(getPointX(index), y);
150 }
```

## Задание 6

Для выполнения задания 6 потребовалось добавить новые поля в класс TabulatedFunction

```
public class TabulatedFunction { 4 usages
    private FunctionPoint[] points; //массив точек 33 usages
    private int pointsCount; // количество точек 16 usages
    private int pointsLimit; // вместимость массива 8 usages
```

Теперь длина массива не совпадает с количеством точек табулированной функции, поэтому есть запасные места, чтобы потом добавлять точки или удалять

```
public TabulatedFunction(double leftX, double rightX, int pointsCount) { 1 usage
    this.pointsCount = pointsCount;
    this.pointsLimit = pointsCount + 10;
    this.points = new FunctionPoint[pointsLimit];

    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);
    }
}
```

Везде, где я использовала `points.length` в прошлых заданиях, я поменяла на `pointsCount` – реальное количество точек.

Методы удаления и добавления точек:

```
152     //Метод удаления точки функции
153     public void deletePoint(int index) { 1 usage
154         if (pointsCount <= 2) {
155             return; // нельзя удалять, т.к. мало точек
156         }
157         System.arraycopy(points, srcPos: index + 1, points, index, length: pointsCount - index - 1);
158         pointsCount--;
159         points[pointsCount] = null;
160     }
```

```

161     //Метод добавления новой точки
162     public void addPoint(FunctionPoint point) { 2 usages
163
164         int searchIndex = 0;
165         while (searchIndex < pointsCount && doubleGreater(point.getX(), points[searchIndex].getX())) {
166             searchIndex++;
167         }
168
169         if (pointsCount == pointsLimit) {
170             pointsLimit = pointsLimit * 2;
171             FunctionPoint[] newPoints = new FunctionPoint[pointsLimit];
172             System.arraycopy(points, [srcPos: 0, newPoints, [destPos: 0, pointsCount];
173             points = newPoints;
174         }
175         System.arraycopy(points, searchIndex, points, [destPos: searchIndex + 1, [length: pointsCount - searchIndex];
176         points[searchIndex] = new FunctionPoint(point);
177         pointsCount++;
178     }
179 }
```

## Задание 7

```

FunctionPoint.java   TabulatedFunction.java   Main.java
1 import functions.FunctionPoint;
2 import functions.TabulatedFunction;
3
4 public class Main {
5     private static final double EPSILON = 1e-10; 1 usage
6     //Сравнение чисел с плавающей точкой
7     public static boolean doubleEqual(double a, double b) { 4 usages
8         return Math.abs(a - b) < EPSILON;
9     }
10
11    public static void main(String[] args) {
12        System.out.println("Создание экземпляра класса через первый конструктор: ");
13        TabulatedFunction f = new TabulatedFunction( leftX: 0, rightX: 6, pointsCount: 10 );
14        double[] y_values = new double[f.getPointsCount()];
15        System.out.println("Функция f(x) = 2x + 1");
16        System.out.println("Область определения: [" + f.getLeftDomainBorder() + "; " + f.getRightDomainBorder() + "]");
17        System.out.println("Количество точек: " + f.getPointsCount());
18        System.out.println("\nТочки функции: ");
19
20        for (int i = 0; i < f.getPointsCount(); i++) {
21            f.setPointY(i, y: 2*f.getPointX(i) + 1);
22            System.out.printf("%d: (%.2f; %.2f)%n", i+1, f.getPointX(i), f.getPointY(i));
23            y_values[i] = f.getPointY(i);
24        }
25    }
26}
```

```

25     System.out.println("\nСоздание экземпляра класса через второй конструктор: ");
26     System.out.println("Точки функции: ");
27     TabulatedFunction f1 = new TabulatedFunction( leftX: 0, rightX: 6, y_values);
28     for (int i = 0; i < y_values.length; i++) {
29         System.out.printf("%d: (%.2f; %.2f)%n", i+1, f.getPointX(i), f.getPointY(i));
30     }
31     //Проверка метода getFunctionValue
32     System.out.println("\nЗначения функции f(x) в разных x: ");
33     double[] x_val = {-1.3, 18, 1, 6, 43, 3.1};
34     for (int i = 0; i < x_val.length; i++) {
35         if (Double.isNaN(f.getFunctionValue(x_val[i]))) {
36             System.out.println("Значение x = " + x_val[i] + " лежит вне области определения функции" );
37         }
38         else {
39             System.out.printf("f(%2f) = %.2f%n", x_val[i], f.getFunctionValue(x_val[i]));
40         }
41     }
42     //Проверка метода setPoint #1
43     System.out.println("\nЗамена точки №3 на точку (7; 15)");
44     System.out.printf("Исходная точка %d: (%.2f; %.2f)%n", 3, f.getPointX( index: 2), f.getPointY( index: 2));
45     FunctionPoint test1 = new FunctionPoint( x: 7, y: 15);
46     f.setPoint( index: 2, test1);
47     if (doubleEqual(f.getPointX( index: 2), test1.getX())) {
48         System.out.println("\nЗамена точки №3 прошла успешно");
49     }
50     else {
51         System.out.println("Замена не удалась, так как x не попал в интервал");
52     }
53     System.out.printf("Точка %d: (%.2f; %.2f)%n", 3, f.getPointX( index: 2), f.getPointY( index: 2));
54
55     //Проверка метода setPoint #2
56     System.out.println("\nЗамена точки №8 на точку (5; 11)");
57     System.out.printf("Исходная точка %d: (%.2f; %.2f)%n", 8, f.getPointX( index: 7), f.getPointY( index: 7));
58     FunctionPoint test2 = new FunctionPoint( x: 5, y: 11);
59     f.setPoint( index: 7, test2);
60     if (doubleEqual(f.getPointX( index: 7), test2.getX())) {
61         System.out.println("Замена точки №8 прошла успешно");
62     }
63     else {
64         System.out.println("Замена не удалась, так как x не попал в интервал");
65     }
66     System.out.printf("Точка %d: (%.2f; %.2f)%n", 8, f.getPointX( index: 7), f.getPointY( index: 7));
67
68     //Проверка метода setPointX #1
69     System.out.println("\nЗамена точки №2 на точку с координатой x = 1");
70     System.out.printf("Исходная точка %d: (%.2f; %.2f)%n", 2, f.getPointX( index: 1), f.getPointY( index: 1));
71     f.setPointX( index: 1, x: 1);
72     if (doubleEqual(f.getPointX( index: 1), b: 1)) {
73         f.setPointY( index: 1, y: 3);
74         System.out.println("Замена точки №2 прошла успешно");
75     }
76     else {
77         System.out.println("Замена не удалась, так как x не попал в интервал");
78     }
79     System.out.printf("Точка %d: (%.2f; %.2f)%n", 2, f.getPointX( index: 1), f.getPointY( index: 1));
80
81     //Проверка метода setPointX #2
82     System.out.println("\nЗамена точки №9 на точку с координатой x = 4.9");
83     System.out.printf("Исходная точка %d: (%.2f; %.2f)%n", 9, f.getPointX( index: 8), f.getPointY( index: 8));
84     f.setPointX( index: 8, x: 4.9);
85     if (doubleEqual(f.getPointX( index: 8), b: 4.9)) {
86         f.setPointY( index: 8, y: 4.9*2+1);
87         System.out.println("Замена точки №9 прошла успешно");
88     }
89     else {

```

```

89     else {
90         System.out.println("Замена не удалась, так как x не попал в интервал");
91     }
92     System.out.printf("Точка %d: (%.2f; %.2f)%n", 9, f.getPointX(index: 8), f.getPointY(index: 8));
93
94     System.out.println("\nТочки функции после замены: ");
95     for (int i = 0; i < f.getPointsCount(); i++) {
96         System.out.printf("%d: (%.2f; %.2f)%n", i+1, f.getPointX(i), f.getPointY(i));
97     }
98     //Проверка метода deletePoint
99     System.out.printf("\nУдаление точки №%d: (%.2f; %.2f)%n", 9, f.getPointX(index: 8), f.getPointY(index: 8));
100    f.deletePoint(index: 8);
101    System.out.println("Точки функции после удаления: ");
102    for (int i = 0; i < f.getPointsCount(); i++) {
103        System.out.printf("%d: (%.2f; %.2f)%n", i+1, f.getPointX(i), f.getPointY(i));
104    }
105
106    //Проверка метода addPoint
107    System.out.println("\nДобавление точек (7, 15) и (3, 7)");
108    FunctionPoint test_add1 = new FunctionPoint(x: 7, y: 15);
109    f.addPoint(test_add1);
110    FunctionPoint test_add2 = new FunctionPoint(x: 3, y: 7);
111    f.addPoint(test_add2);
112    System.out.println("Точки функции после добавления: ");
113    for (int i = 0; i < f.getPointsCount(); i++) {
114        System.out.printf("%d: (%.2f; %.2f)%n", i+1, f.getPointX(i), f.getPointY(i));
115    }
116}

```

## Вывод

```

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Prog
Создание экземпляра класса через первый конструктор:
Функция f(x) = 2x + 1
Область определения: [0.0; 6.0]
Количество точек: 10

```

Точки функции:

```

1: (0,00; 1,00)
2: (0,67; 2,33)
3: (1,33; 3,67)
4: (2,00; 5,00)
5: (2,67; 6,33)
6: (3,33; 7,67)
7: (4,00; 9,00)
8: (4,67; 10,33)
9: (5,33; 11,67)
10: (6,00; 13,00)

```

Создание экземпляра класса через второй конструктор:

Точки функции:

1: (0,00; 1,00)  
2: (0,67; 2,33)  
3: (1,33; 3,67)  
4: (2,00; 5,00)  
5: (2,67; 6,33)  
6: (3,33; 7,67)  
7: (4,00; 9,00)  
8: (4,67; 10,33)  
9: (5,33; 11,67)  
10: (6,00; 13,00)

Значения функции  $f(x)$  в разных  $x$ :

Значение  $x = -1.3$  лежит вне области определения функции

Значение  $x = 18.0$  лежит вне области определения функции

$f(1,00) = 3,00$

$f(6,00) = 13,00$

Значение  $x = 43.0$  лежит вне области определения функции

$f(3,10) = 7,20$

Замена точки №3 на точку (7; 15)

Исходная точка 3: (1,33; 3,67)

Замена не удалась, так как  $x$  не попал в интервал

Точка 3: (1,33; 3,67)

Замена точки №8 на точку (5; 11)

Исходная точка 8: (4,67; 10,33)

Замена точки №8 прошла успешно

Точка 8: (5,00; 11,00)

Замена точки №2 на точку с координатой  $x = 1$

Исходная точка 2: (0,67; 2,33)

Замена точки №2 прошла успешно

Точка 2: (1,00; 3,00)

Замена точки №9 на точку с координатой  $x = 4.9$

Исходная точка 9: (5,33; 11,67)

Замена не удалась, так как  $x$  не попал в интервал

Точка 9: (5,33; 11,67)

Точки функции после замены:

- 1: (0,00; 1,00)
- 2: (1,00; 3,00)
- 3: (1,33; 3,67)
- 4: (2,00; 5,00)
- 5: (2,67; 6,33)
- 6: (3,33; 7,67)
- 7: (4,00; 9,00)
- 8: (5,00; 11,00)
- 9: (5,33; 11,67)
- 10: (6,00; 13,00)

Удаление точки №9: (5,33; 11,67)

Точки функции после удаления:

- 1: (0,00; 1,00)
- 2: (1,00; 3,00)
- 3: (1,33; 3,67)
- 4: (2,00; 5,00)
- 5: (2,67; 6,33)
- 6: (3,33; 7,67)
- 7: (4,00; 9,00)
- 8: (5,00; 11,00)
- 9: (6,00; 13,00)

Добавление точек (7, 15) и (3, 7)

Точки функции после добавления:

- 1: (0,00; 1,00)
- 2: (1,00; 3,00)
- 3: (1,33; 3,67)
- 4: (2,00; 5,00)
- 5: (2,67; 6,33)
- 6: (3,00; 7,00)
- 7: (3,33; 7,67)
- 8: (4,00; 9,00)
- 9: (5,00; 11,00)
- 10: (6,00; 13,00)
- 11: (7,00; 15,00)