

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА»

Институт «Информатики и кибернетики»

Специальность «Фотоника и оптоинформатика»

Отчет по лабораторной работе № 2

Выполнила: Гамаюнова Д.И.,

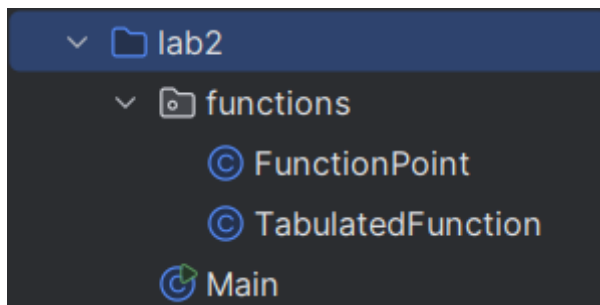
группа 6201-120303D

Проверил: преподаватель Борисов Д.С.

Самара 2025

Задание 1

Создаем пакет `functions`, в котором далее будут создаваться классы программы.



Задание 2

В пакете `functions` создаем класс `FunctionPoint`, объект которого описывает одну точку табулированной функции и хранит координаты `x` и `y`, предоставляя методы для работы с ними.

```
Main.java  FunctionPoint.java  TabulatedFunction.java
1  package functions;
2
3  public class FunctionPoint { 14 usages
4      private double x; 5 usages
5      private double y; 5 usages
6      public FunctionPoint(double x, double y) { 3 usages
7          this.x = x;
8          this.y = y;
9      }
10 @ public FunctionPoint(FunctionPoint point) { 3 usages
11     this.x = point.x;
12     this.y = point.y;
13 }
14 public FunctionPoint() { no usages
15     this(x: 0.0, y: 0.0);
16 }
17 public double getX() { 13 usages
18     return x;
19 }
20 public void setX(double x) { 1 usage
21     this.x = x;
22 }
23 public double getY() { 3 usages
24     return y;
25 }
26 public void setY(double y) { 1 usage
27     this.y = y;
28 }
29 }
```

Задание 3

В пакете `functions` создаем класс `TabulatedFunction`, объект которого описывает табулированную функцию, с помощью следующих методов

```
1 package functions;
2
3 public class TabulatedFunction { no usages
4     private FunctionPoint[] points; 30 usages
5     private int size; 17 usages
6     public TabulatedFunction(double leftX, double rightX, int pointsCount) { no usages
7         if (pointsCount < 2) {
8             pointsCount = 2;
9         }
10        this.points = new FunctionPoint[pointsCount];
11        this.size = pointsCount;
12        double step = (rightX - leftX) / (pointsCount - 1);
13        for (int i = 0; i < pointsCount; i++) {
14            double x = leftX + i * step;
15            points[i] = new FunctionPoint(x, y: 0.0);
16        }
17    }
18 @ public TabulatedFunction(double leftX, double rightX, double[] values) { no usages
19        int n = values.length;
20        if (n < 2) {
21            n = 2;
22        }
23        this.points = new FunctionPoint[n];
24        this.size = n;
25        double step = (rightX - leftX) / (n - 1);
26        for (int i = 0; i < n; i++) {
27            double x = leftX + i * step;
28            points[i] = new FunctionPoint(x, values[i]);
29        }
30    }
```

Задание 4

В классе `TabulatedFunction` описываем методы, необходимые для работы с функцией.

```
31     public double getLeftDomainBorder() { 1 usage
32         return points[0].getX();
33     }
34     public double getRightDomainBorder() { 1 usage
35         return points[size - 1].getX();
36     }
37     public double getFunctionValue(double x) { no usages
38         if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
39             return Double.NaN;
40         }
41         for (int i = 0; i < size - 1; i++) {
42             double x0 = points[i].getX();
43             double x1 = points[i + 1].getX();
44             double y0 = points[i].getY();
45             double y1 = points[i + 1].getY();
46
47             if (x == x0) return y0;
48             if (x > x0 && x < x1) {
49                 return y0 + (y1 - y0) * (x - x0) / (x1 - x0);
50             }
51         }
52         return Double.NaN;
53     }
```

Задание 5

В классе `TabulatedFunction` описываем методы, необходимые для работы с точками табулированной функции. Считаем, что нумерация точек начинается с нуля.

```
54 > public int getPointsCount() { return size; }
57 > public FunctionPoint getPoint(int index) { return new FunctionPoint(points[index]); }
60 public void setPoint(int index, FunctionPoint point) { 1 usage
61     if (index > 0 && point.getX() <= points[index - 1].getX()) {
62         return;
63     }
64     if (index < size - 1 && point.getX() >= points[index + 1].getX()) {
65         return;
66     }
67     points[index] = new FunctionPoint(point);
68 }
69 > public double getPointX(int index) { return points[index].getX(); }
72 public void setPointX(int index, double x) { no usages
73     if (index > 0 && x <= points[index - 1].getX()) {
74         return;
75     }
76     if (index < size - 1 && x >= points[index + 1].getX()) {
77         return;
78     }
79     points[index].setX(x);
80 }
81 > public double getPointY(int index) { return points[index].getY(); }
84 public void setPointY(int index, double y) { 1 usage
85     points[index].setY(y);
86 }
```

Задание 6

В классе `TabulatedFunction` описываем методы, изменяющие количество точек табулированной функции.

```
87     public void deletePoint(int index) { 1 usage
88         if (size <= 2) {
89             return;
90         }
91         for (int i = index; i < size - 1; i++) {
92             points[i] = points[i + 1];
93         }
94         points[size - 1] = null;
95         size--;
96     }
97     public void addPoint(FunctionPoint point) { 2 usages
98         if (size == points.length) {
99             FunctionPoint[] newPoints = new FunctionPoint[size * 2];
100             System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, size);
101             points = newPoints;
102         }
103         int insertIndex = 0;
104         double x = point.getX();
105         while (insertIndex < size && points[insertIndex].getX() < x) {
106             insertIndex++;
107         }
108         for (int i = size; i > insertIndex; i--) {
109             points[i] = points[i - 1];
110         }
111         points[insertIndex] = new FunctionPoint(point);
112         size++;
113     }
114 }
```

Задание 7

Создаем класс Main, содержащий точку входа программы.

```
1  import functions.FunctionPoint;
2  import functions.TabulatedFunction;
3
4  public class Main {
5      public static void main(String[] args) {
6          double left = -2;
7          double right = 3;
8          int count = 6;
9          TabulatedFunction func = new TabulatedFunction(left, right, count);
10         System.out.println("Функция f(x) = 2x² + 3x - 1");
11         System.out.println("Левая граница функции: " + func.getLeftDomainBorder());
12         System.out.println("Правая граница функции: " + func.getRightDomainBorder());
13
14         for (int i = 0; i < func.getPointsCount(); i++) {
15             double x = func.getPointX(i);
16             func.setPointY(i, 2*x*x + 3*x - 1);
17         }
18
19         System.out.println("\nПроверка значений функции:");
20         for (double x = -3; x <= 4; x += 0.5) {
21             System.out.println("f(" + x + ") = " + func.getFunctionValue(x));
22         }
23
24         double[] values = {1, -2, -1, 4, 13, 26};
25         TabulatedFunction func1 = new TabulatedFunction( leftX: -2, rightX: 3, values);
26
27         System.out.println("\nФункция создана с массивом значений:");
28         for (int i = 0; i < func1.getPointsCount(); i++) {
29             System.out.println("x=" + func1.getPointX(i) + ", y=" + func1.getPointY(i));
30         }
31
32         System.out.println("\nИзменяем значение y третьей точки:");
33         func.setPoint( index: 2, new FunctionPoint( x: 0, y: -1));
34         System.out.println("Новая точка 2: x = " + func.getPointX( index: 2) + ", y = " + func.getPointY( index: 2));
35
36         System.out.println("\nДобавляем новую точку (x=1.5, y=8):");
37         func.addPoint(new FunctionPoint( x: 1.5, y: 8));
38         for (int i = 0; i < func.getPointsCount(); i++) {
39             System.out.println("(" + func.getPointX(i) + "; " + func.getPointY(i) + ")");
40         }
41
42         System.out.println("\nНаходим значение y для 2.5 с помощью линейной интерполяции:");
43         double valueAt25 = func.getFunctionValue( x: 2.5);
44         System.out.println("f(2.5) = " + valueAt25);
45         func.addPoint(new FunctionPoint( x: 2.5, valueAt25));
46
47         System.out.println("Добавлена новая точка (2.5; " + valueAt25 + ")");
48
49         System.out.println("\nУдаляем четвертую точку:");
50         func.deletePoint( index: 3);
51         for (int i = 0; i < func.getPointsCount(); i++) {
52             System.out.println("(" + func.getPointX(i) + "; " + func.getPointY(i) + ")");
53         }
54     }
```

Проверяем работу написанных классов:

```
"C:\Program Files\jdk-25\bin\java.exe"
```

```
Функция  $f(x) = 2x^2 + 3x - 1$ 
```

```
Левая граница функции: -2.0
```

```
Правая граница функции: 3.0
```

```
Проверка значений функции:
```

```
f(-3.0) = NaN
```

```
f(-2.5) = NaN
```

```
f(-2.0) = 1.0
```

```
f(-1.5) = -0.5
```

```
f(-1.0) = -2.0
```

```
f(-0.5) = -1.5
```

```
f(0.0) = -1.0
```

```
f(0.5) = 1.5
```

```
f(1.0) = 4.0
```

```
f(1.5) = 8.5
```

```
f(2.0) = 13.0
```

```
f(2.5) = 19.5
```

```
f(3.0) = NaN
```

```
f(3.5) = NaN
```

```
f(4.0) = NaN
```

```
Функция создана с массивом значений:
```

```
x=-2.0, y=1.0
```

```
x=-1.0, y=-2.0
```

```
x=0.0, y=-1.0
```

```
x=1.0, y=4.0
```

```
x=2.0, y=13.0
```

```
x=3.0, y=26.0
```

```
Изменяем значение y третьей точки:
```

```
Новая точка 2: x = 0.0, y = -1.0
```

```
Добавляем новую точку (x=1.5, y=8):
```

```
(-2.0; 1.0)
```

```
(-1.0; -2.0)
```

```
(0.0; -1.0)
```

```
(1.0; 4.0)
```

```
(1.5; 8.0)
```

```
(2.0; 13.0)
```

```
(3.0; 26.0)
```

```
Находим значение y для 2.5 с помощью линейной интерполяции:
```

```
f(2.5) = 19.5
```

```
Добавлена новая точка (2.5; 19.5)
```

```
Удаляем четвертую точку:
```

```
(-2.0; 1.0)
```

```
(-1.0; -2.0)
```

```
(0.0; -1.0)
```

```
(1.5; 8.0)
```

```
(2.0; 13.0)
```

```
(2.5; 19.5)
```

```
(3.0; 26.0)
```

```
Process finished with exit code 0
```