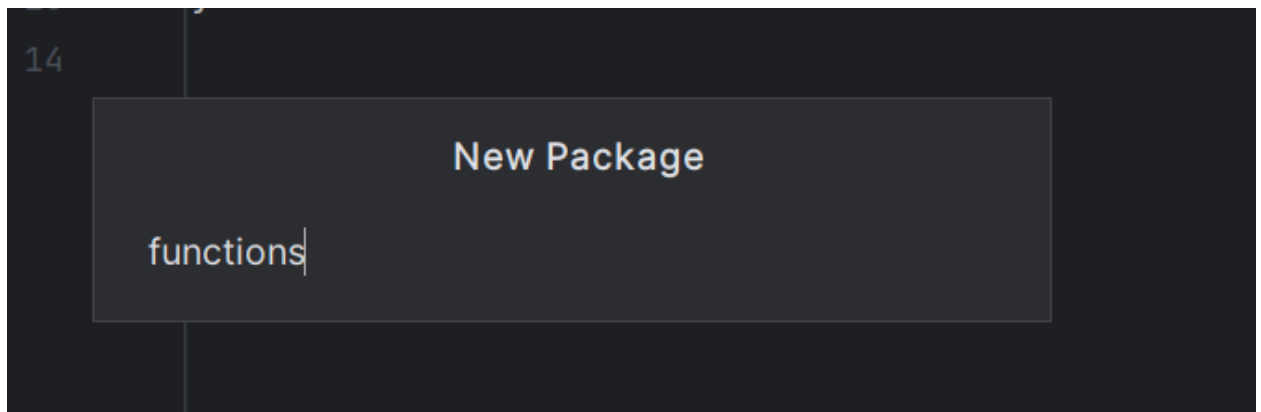
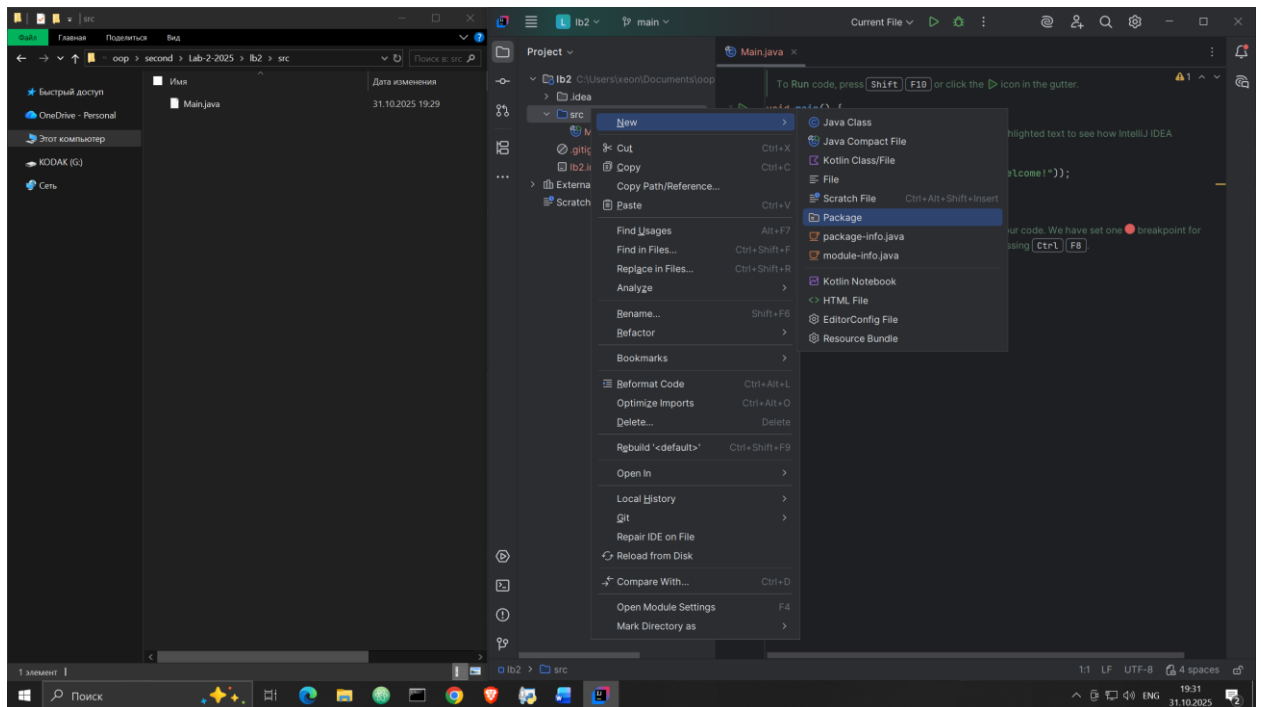


Отчёт по лабораторной работе №2
Тенигин Валерий 6204-010302D

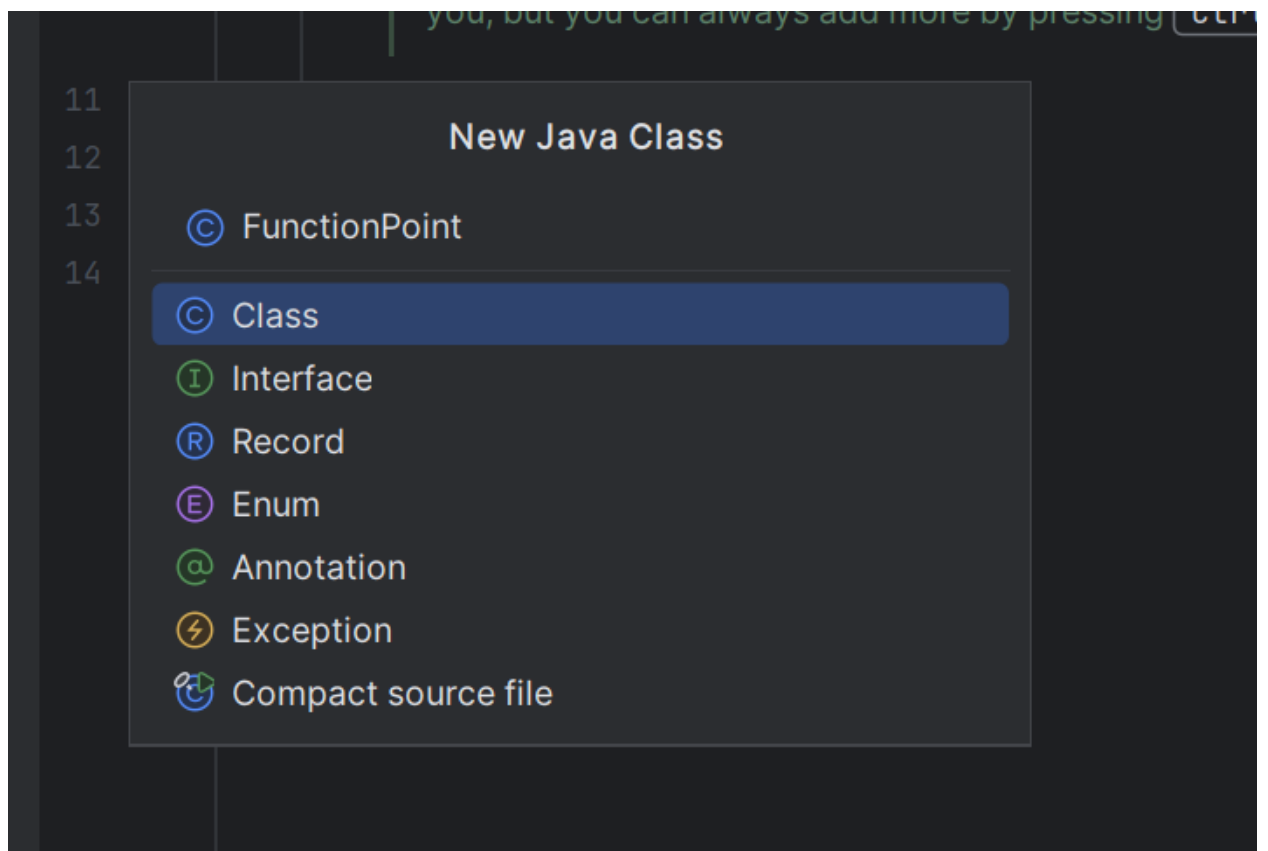
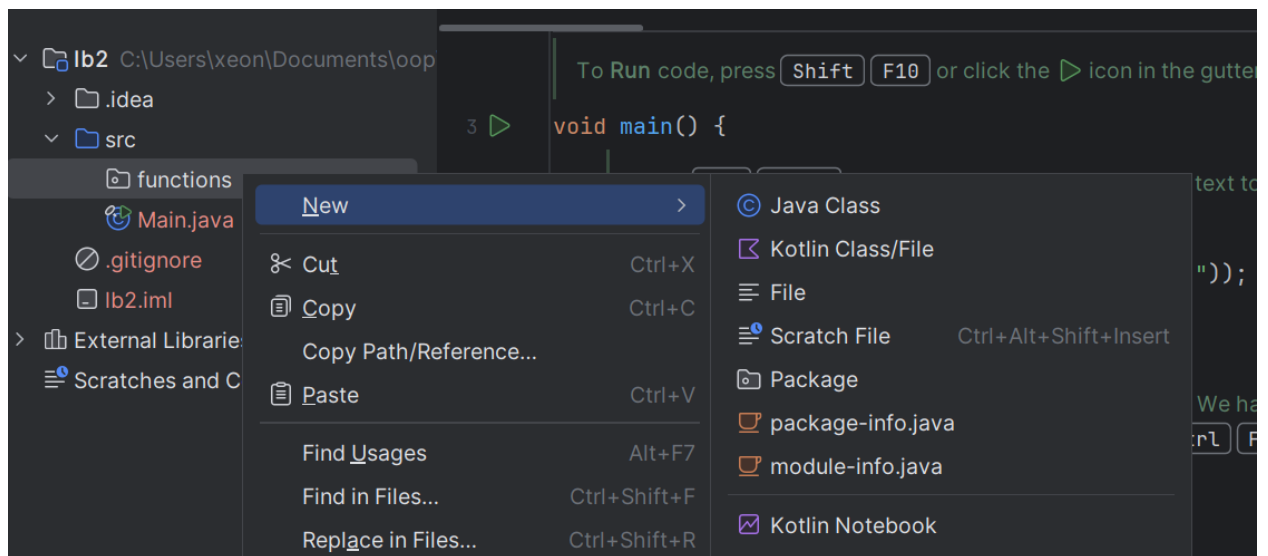
Задание 1

Создание пакета functions



Задание 2

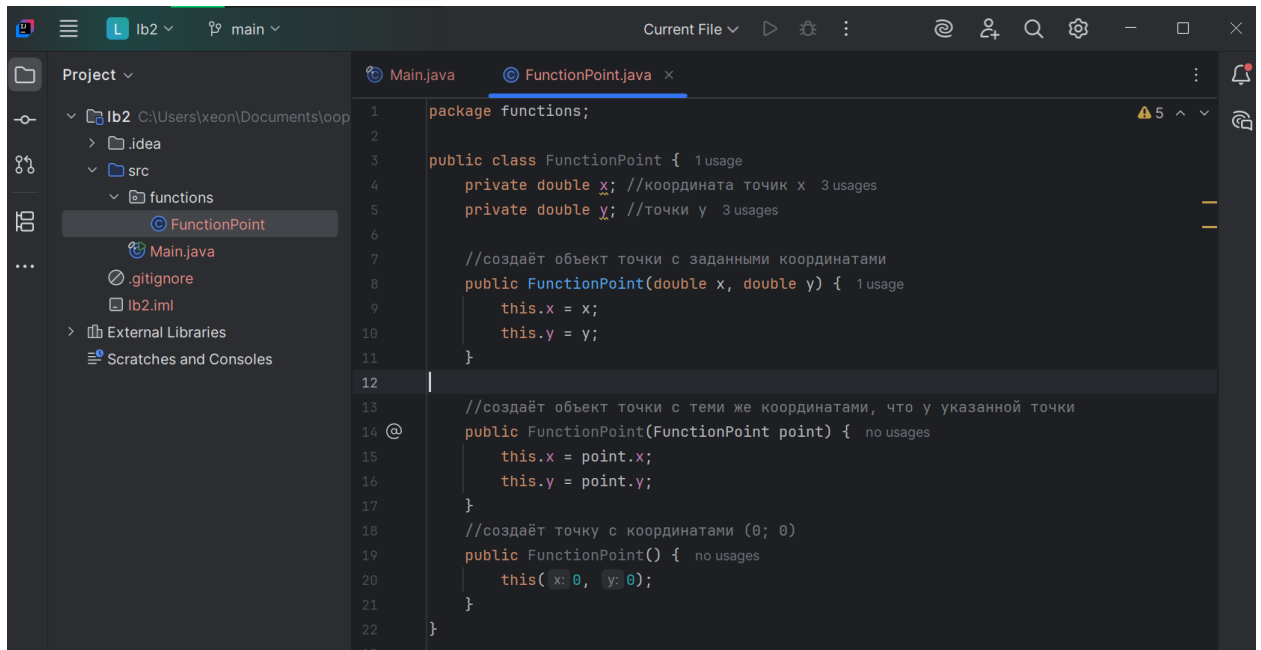
Внутри созданного пакета создаём класс FunctionPoint



В этом классе описываем следующие конструкторы:

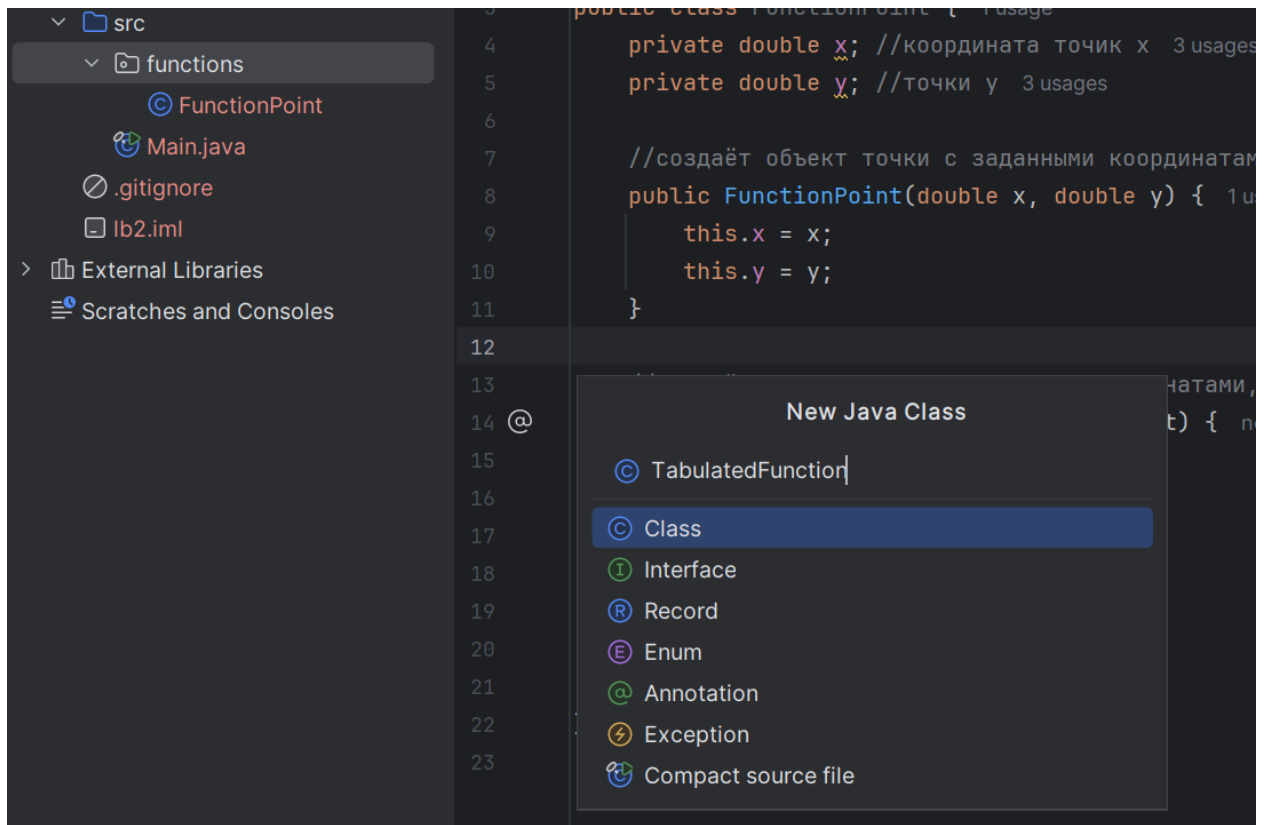
- `FunctionPoint(double x, double y)` – создаёт объект точки с заданными координатами;

- `FunctionPoint(FunctionPoint point)` – создаёт объект точки с теми же координатами, что у указанной точки;
- `FunctionPoint()` – создаёт точку с координатами (0; 0).



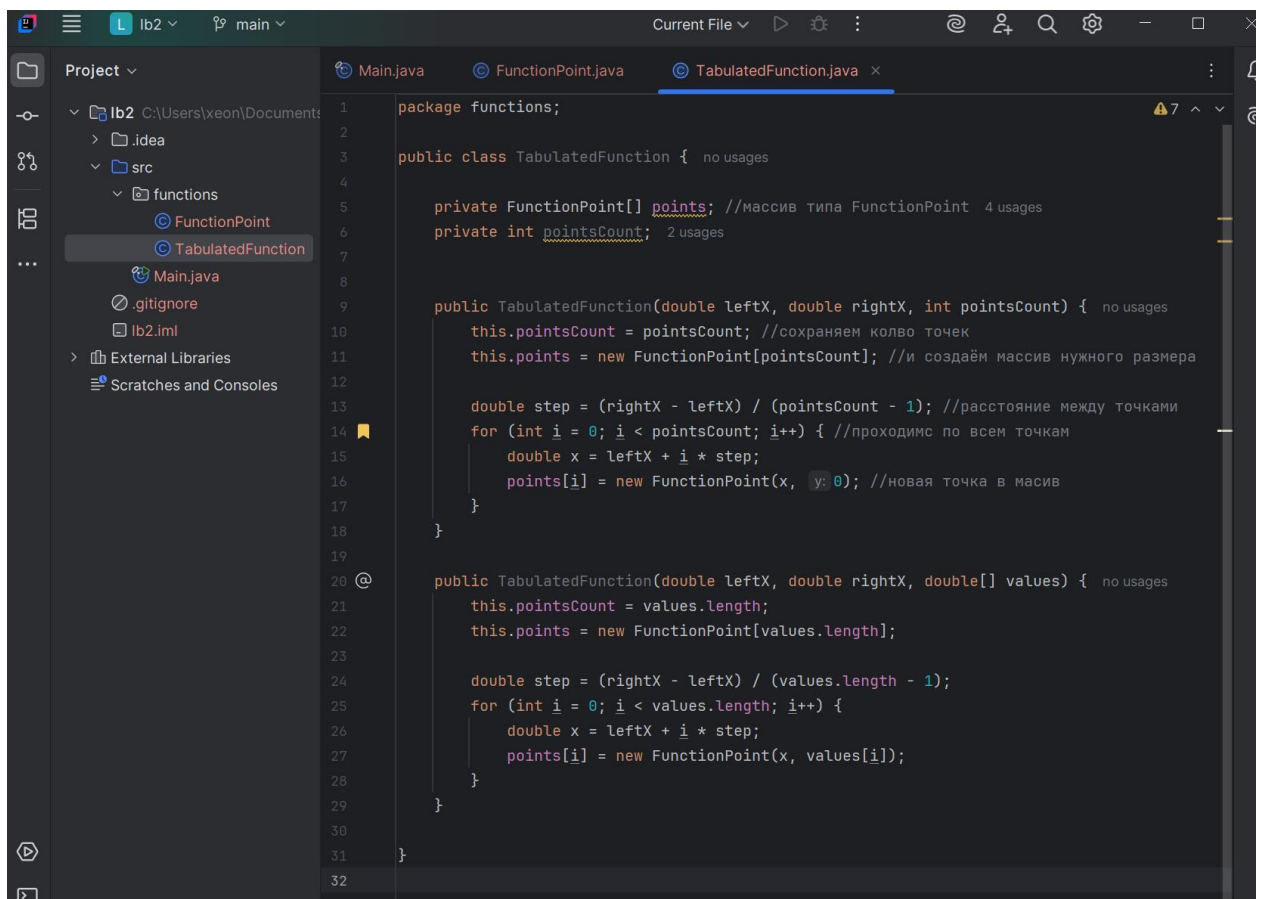
Задание 3

Создаём ещё один класс TabulatedFunction



Создаём следующие конструкторы:

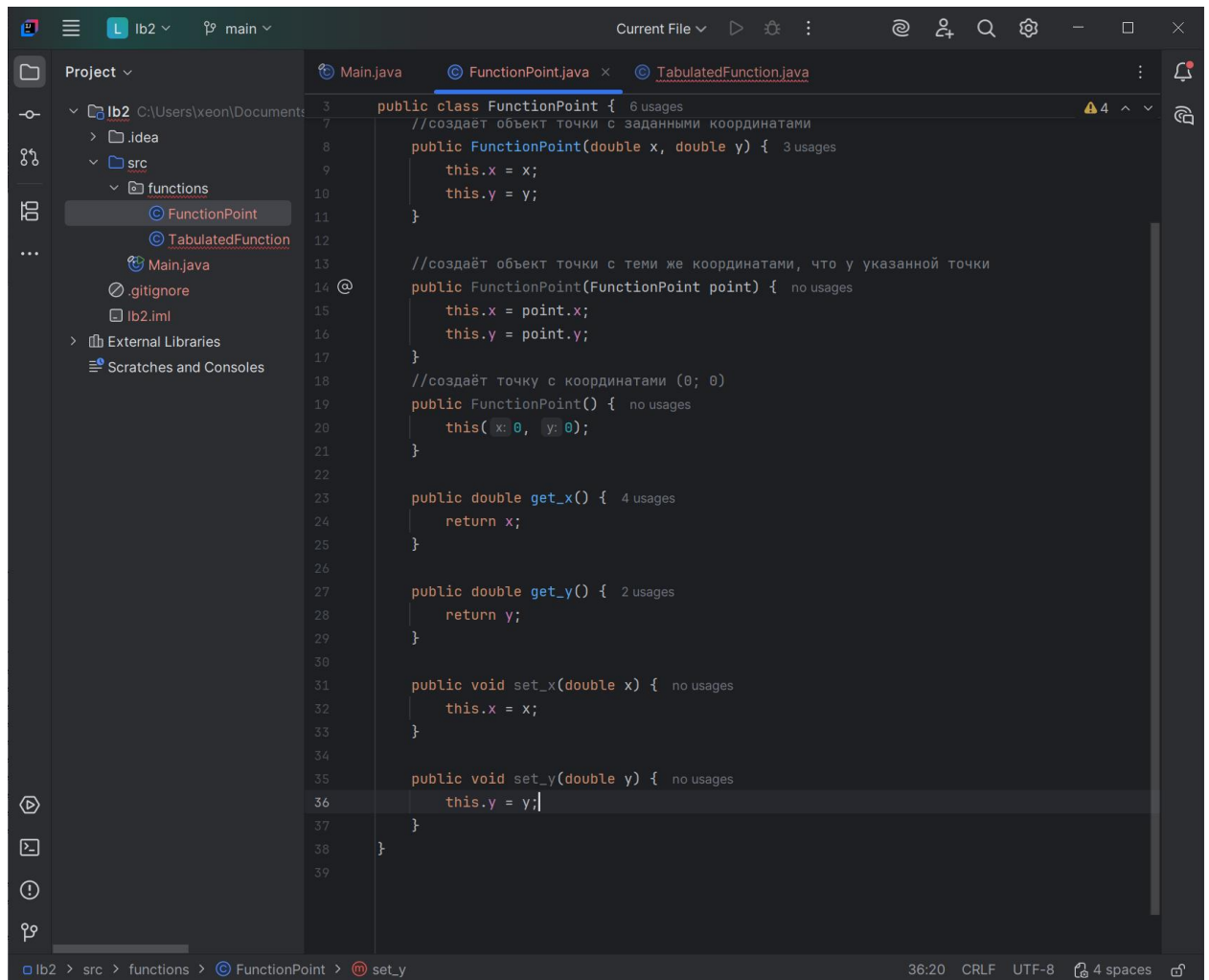
- `TabulatedFunction(double leftX, double rightX, int pointsCount)` – создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования (значения функции в точках при этом следует считать равными 0);
- `TabulatedFunction(double leftX, double rightX, double[] values)` – аналогичен предыдущему конструктору, но вместо количества точек получает значения функции в виде массива.



```
1 package functions;
2
3 public class TabulatedFunction { no usages
4
5     private FunctionPoint[] points; //массив типа FunctionPoint 4 usages
6     private int pointsCount; 2 usages
7
8
9     public TabulatedFunction(double leftX, double rightX, int pointsCount) { no usages
10         this.pointsCount = pointsCount; //сохраняем колво точек
11         this.points = new FunctionPoint[pointsCount]; //и создаём массив нужного размера
12
13         double step = (rightX - leftX) / (pointsCount - 1); //расстояние между точками
14         for (int i = 0; i < pointsCount; i++) { //проходим по всем точкам
15             double x = leftX + i * step;
16             points[i] = new FunctionPoint(x, 0); //новая точка в массив
17         }
18     }
19
20     public TabulatedFunction(double leftX, double rightX, double[] values) { no usages
21         this.pointsCount = values.length;
22         this.points = new FunctionPoint[values.length];
23
24         double step = (rightX - leftX) / (values.length - 1);
25         for (int i = 0; i < values.length; i++) {
26             double x = leftX + i * step;
27             points[i] = new FunctionPoint(x, values[i]);
28         }
29     }
30
31 }
32
```

Задание 4

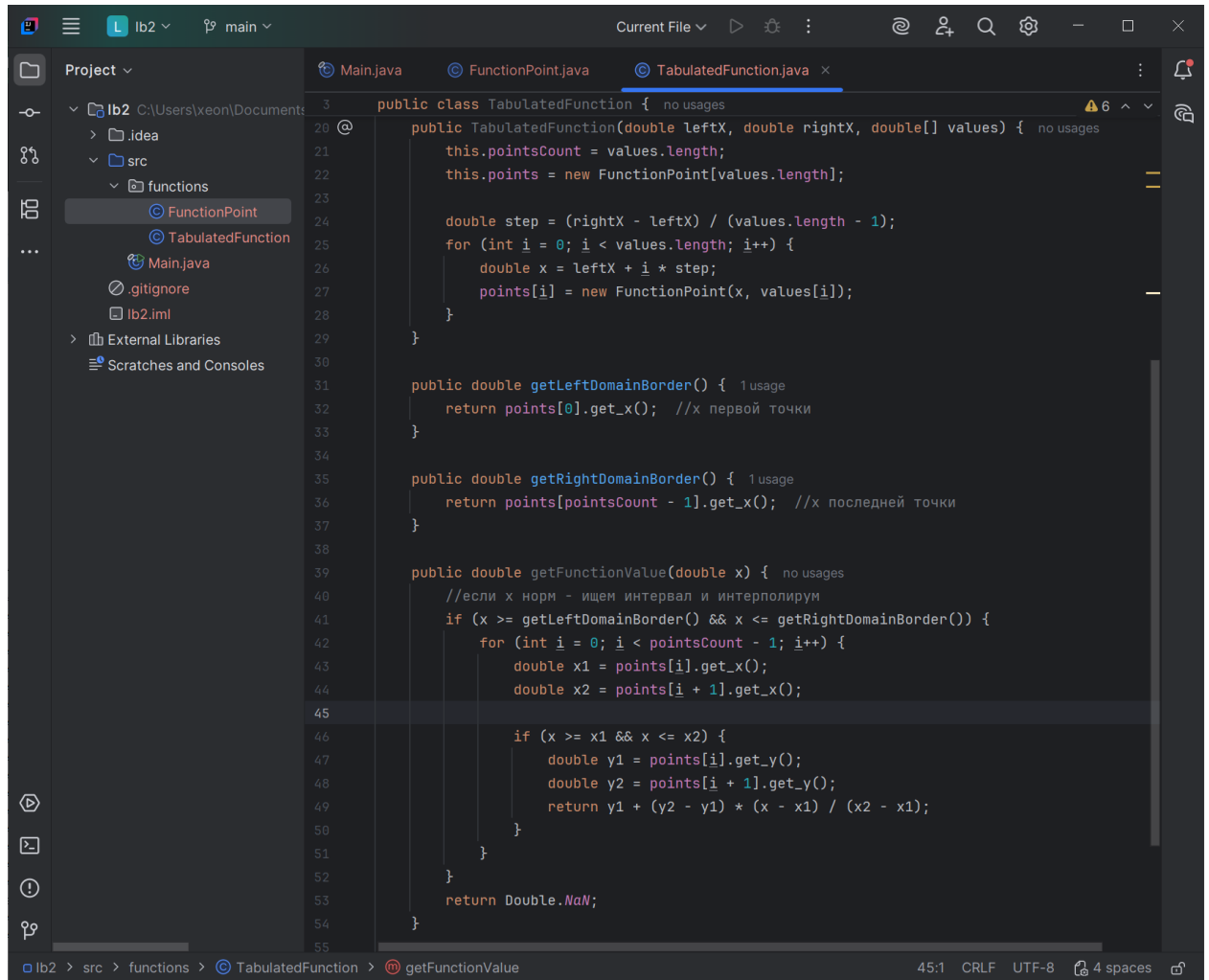
Добавил гетеры и сетеры в класс FunctionPoint



Длъявлены следующие методы:

- Метод `double getLeftDomainBorder()` должен возвращать значение левой границы области определения табулированной функции.
- метод `double getRightDomainBorder()` должен возвращать значение правой границы области определения табулированной функции.
- Метод `double getFunctionValue(double x)` должен возвращать значение функции в точке x , если эта точка лежит в области

определения функции



The screenshot shows an IDE window with the following components:

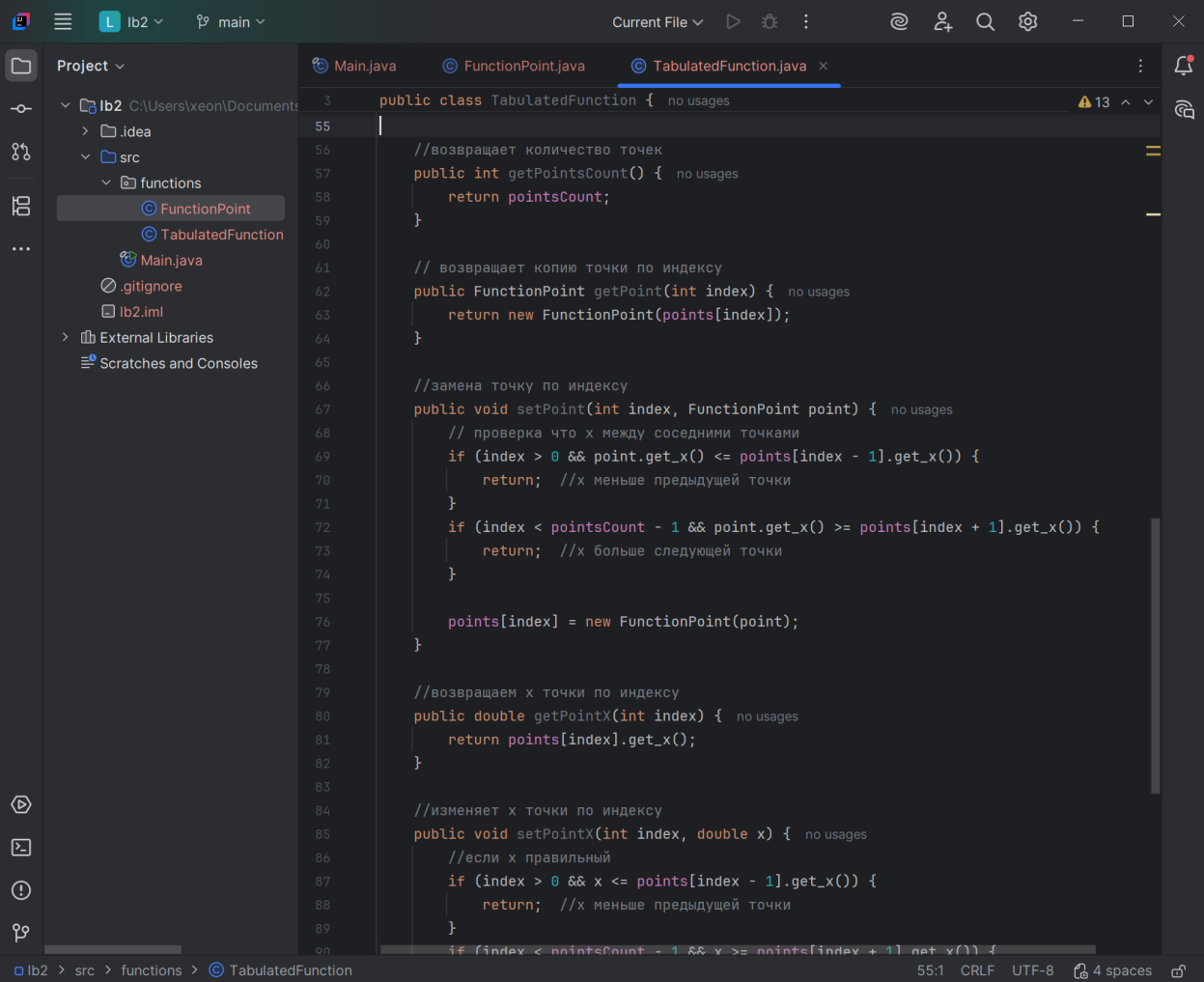
- Project View (Left):** Shows a project named 'lb2' with a 'src' directory containing a 'functions' package. The package contains 'FunctionPoint' and 'TabulatedFunction' classes. Other files like 'Main.java', '.gitignore', and 'lb2.iml' are also visible.
- Editor View (Right):** Displays the code for 'TabulatedFunction.java'. The code defines a public class with a constructor and several methods.

```
3 public class TabulatedFunction { no usages
20 public TabulatedFunction(double leftX, double rightX, double[] values) { no usages
21     this.pointsCount = values.length;
22     this.points = new FunctionPoint[values.length];
23
24     double step = (rightX - leftX) / (values.length - 1);
25     for (int i = 0; i < values.length; i++) {
26         double x = leftX + i * step;
27         points[i] = new FunctionPoint(x, values[i]);
28     }
29 }
30
31 public double getLeftDomainBorder() { 1 usage
32     return points[0].get_x(); //x первой точки
33 }
34
35 public double getRightDomainBorder() { 1 usage
36     return points[pointsCount - 1].get_x(); //x последней точки
37 }
38
39 public double getFunctionValue(double x) { no usages
40     //если x норм - ищем интервал и интерполируем
41     if (x >= getLeftDomainBorder() && x <= getRightDomainBorder()) {
42         for (int i = 0; i < pointsCount - 1; i++) {
43             double x1 = points[i].get_x();
44             double x2 = points[i + 1].get_x();
45
46             if (x >= x1 && x <= x2) {
47                 double y1 = points[i].get_y();
48                 double y2 = points[i + 1].get_y();
49                 return y1 + (y2 - y1) * (x - x1) / (x2 - x1);
50             }
51         }
52     }
53     return Double.NaN;
54 }
55 }
```

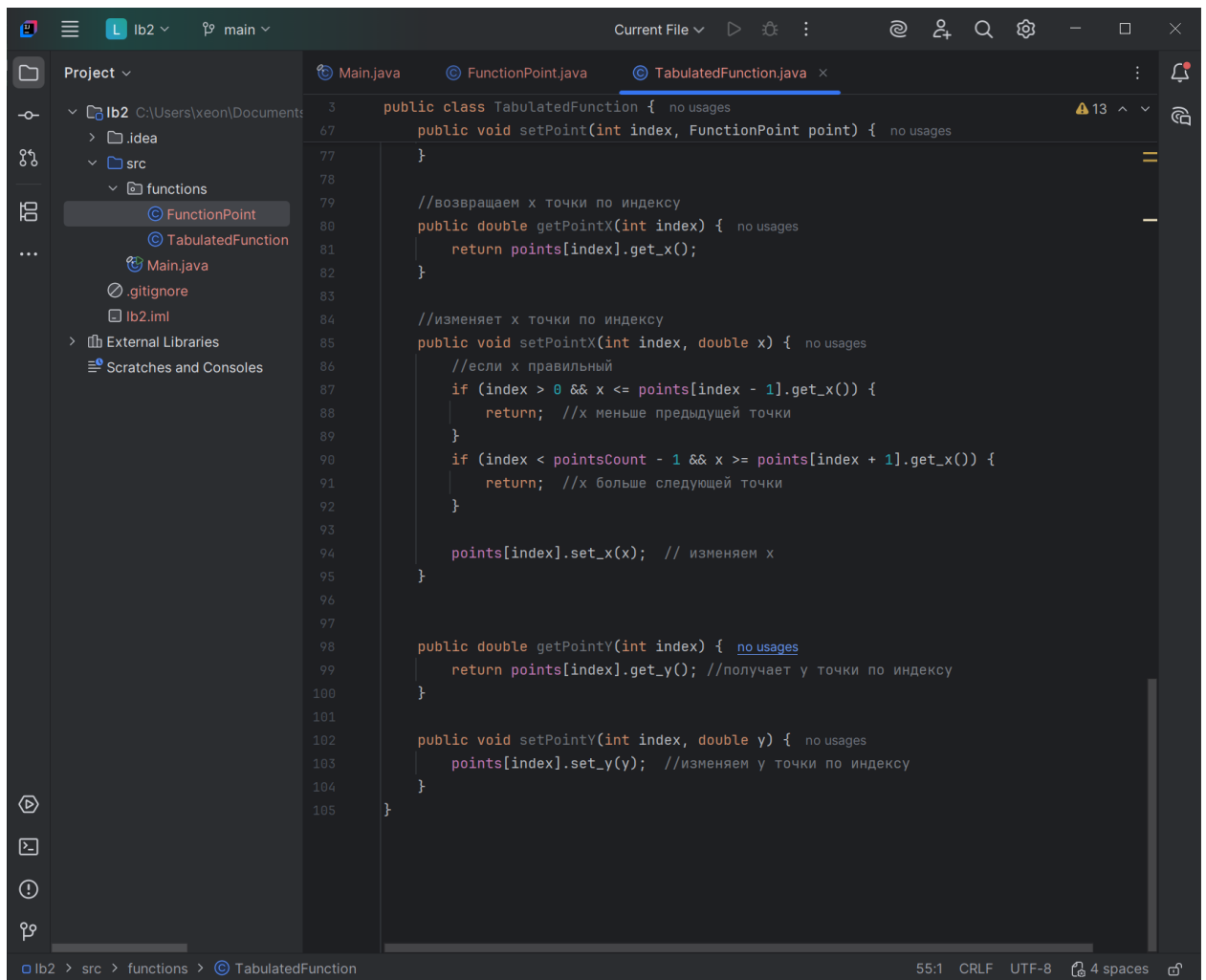
The status bar at the bottom indicates the file path 'lb2 > src > functions > TabulatedFunction > getFunctionValue', the cursor position '45:1', and the encoding 'CRLF UTF-8' with '4 spaces'.

Задание 5

Добавлены методы для работы с точками табулированной функции согласно заданию



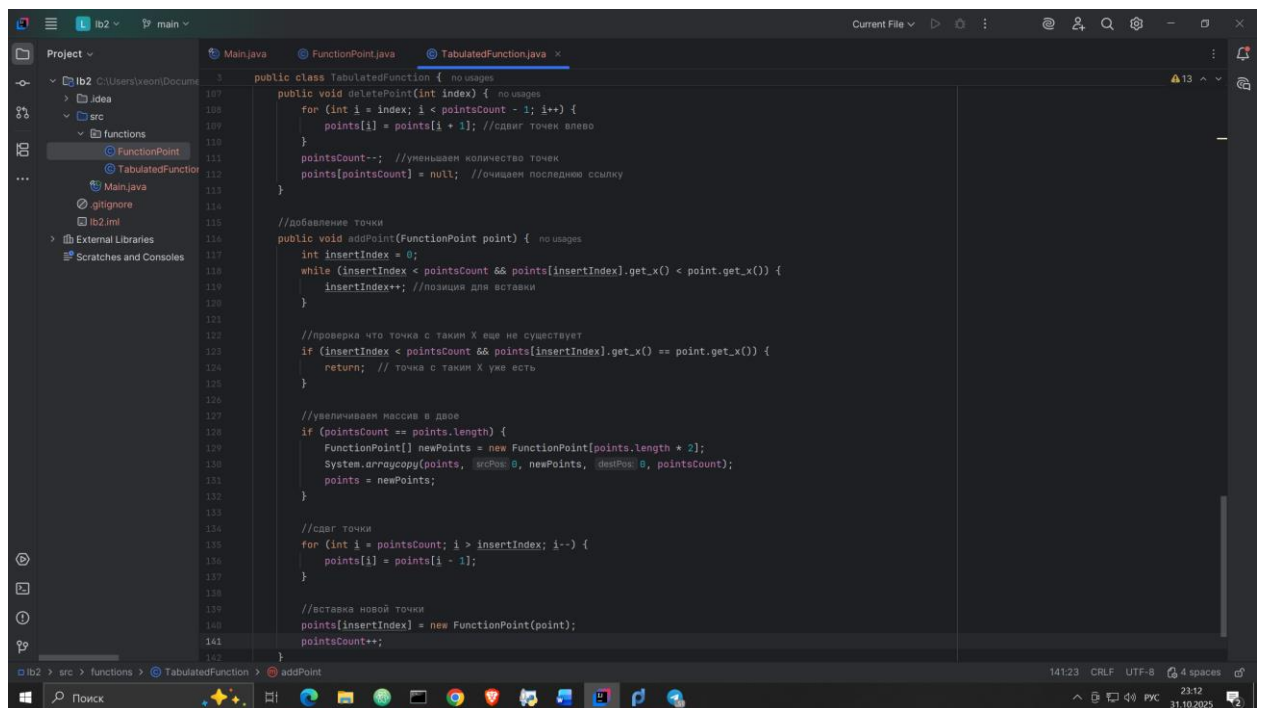
```
55 |  
56 |  
57 | //возвращает количество точек  
58 | public int getPointsCount() { no usages  
59 |     return pointsCount;  
60 | }  
61 |  
62 | // возвращает копию точки по индексу  
63 | public FunctionPoint getPoint(int index) { no usages  
64 |     return new FunctionPoint(points[index]);  
65 | }  
66 |  
67 | //замена точку по индексу  
68 | public void setPoint(int index, FunctionPoint point) { no usages  
69 |     // проверка что x между соседними точками  
70 |     if (index > 0 && point.get_x() <= points[index - 1].get_x()) {  
71 |         return; //x меньше предыдущей точки  
72 |     }  
73 |     if (index < pointsCount - 1 && point.get_x() >= points[index + 1].get_x()) {  
74 |         return; //x больше следующей точки  
75 |     }  
76 |     points[index] = new FunctionPoint(point);  
77 | }  
78 |  
79 | //возвращаем x точки по индексу  
80 | public double getPointX(int index) { no usages  
81 |     return points[index].get_x();  
82 | }  
83 |  
84 | //изменяет x точки по индексу  
85 | public void setPointX(int index, double x) { no usages  
86 |     //если x правильный  
87 |     if (index > 0 && x <= points[index - 1].get_x()) {  
88 |         return; //x меньше предыдущей точки  
89 |     }  
90 |     if (index < pointsCount - 1 && x >= points[index + 1].get_x()) {
```

Задание 6

Добавлены методы изменяющие количество точек табулированной функции:

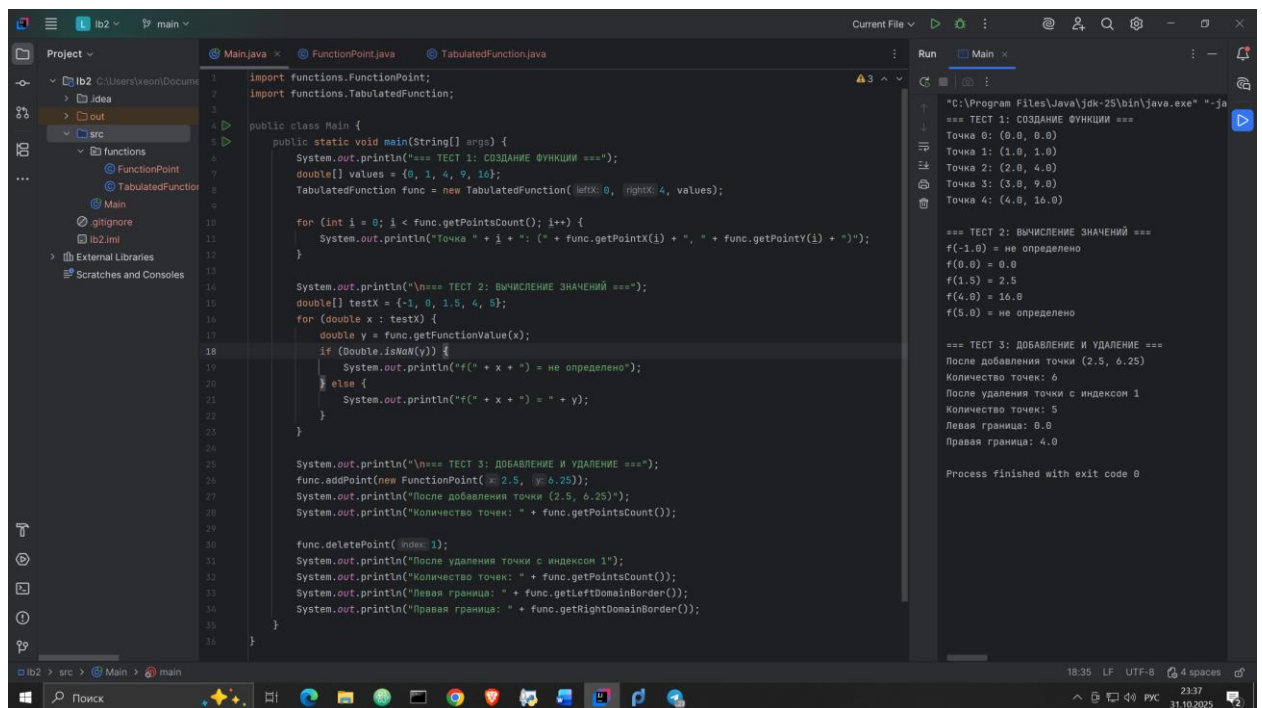
- Метод `void deletePoint(int index)` должен удалять заданную точку табулированной функции.
- Метод `void addPoint(FunctionPoint point)` должен добавлять новую точку табулированной функции.



```
1 public class TabulatedFunction { no usages
2
3     public void deletePoint(int index) { no usages
4         for (int i = index; i < pointsCount - 1; i++) {
5             points[i] = points[i + 1]; //сдвиг точек влево
6         }
7         pointsCount--; //уменьшаем количество точек
8         points[pointsCount] = null; //очищаем последнюю ссылку
9     }
10
11     //добавление точки
12     public void addPoint(FunctionPoint point) { no usages
13         int insertIndex = 0;
14         while (insertIndex < pointsCount && points[insertIndex].get_x() < point.get_x()) {
15             insertIndex++; //позиция для вставки
16         }
17
18         //проверка что точка с таким X еще не существует
19         if (insertIndex < pointsCount && points[insertIndex].get_x() == point.get_x()) {
20             return; // точка с таким X уже есть
21         }
22
23         //увеличиваем массив в два раза
24         if (pointsCount == points.length) {
25             FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
26             System.arraycopy(points, 0, newPoints, 0, pointsCount);
27             points = newPoints;
28         }
29
30         //сдвиг точки
31         for (int i = pointsCount; i > insertIndex; i--) {
32             points[i] = points[i - 1];
33         }
34
35         //вставка новой точки
36         points[insertIndex] = new FunctionPoint(point);
37         pointsCount++;
38     }
39 }
```

Задание 7

Проверка работы



The screenshot shows an IDE with a project named 'Ib2'. The 'src' directory contains three files: 'FunctionPoint.java', 'TabulatedFunction.java', and 'Main.java'. The 'Main.java' file is open, showing the following code:

```
1 import functions.FunctionPoint;
2 import functions.TabulatedFunction;
3
4 public class Main {
5     public static void main(String[] args) {
6         System.out.println("=== ТЕСТ 1: СОЗДАНИЕ ФУНКЦИИ ===");
7         double[] values = {0, 1, 4, 9, 16};
8         TabulatedFunction func = new TabulatedFunction(0, 4, values);
9
10        for (int i = 0; i < func.getPointsCount(); i++) {
11            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
12        }
13
14        System.out.println("\n=== ТЕСТ 2: ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ===");
15        double[] testX = {-1, 0, 1.5, 4, 5};
16        for (double x : testX) {
17            double y = func.getFunctionValue(x);
18            if (Double.isNaN(y)) {
19                System.out.println("f(" + x + ") = не определено");
20            } else {
21                System.out.println("f(" + x + ") = " + y);
22            }
23        }
24
25        System.out.println("\n=== ТЕСТ 3: ДОБАВЛЕНИЕ И УДАЛЕНИЕ ===");
26        func.addPoint(new FunctionPoint(2.5, 6.25));
27        System.out.println("После добавления точки (2.5, 6.25):");
28        System.out.println("Количество точек: " + func.getPointsCount());
29
30        func.deletePoint(1);
31        System.out.println("После удаления точки с индексом 1");
32        System.out.println("Количество точек: " + func.getPointsCount());
33        System.out.println("Левая граница: " + func.getLeftDomainBorder());
34        System.out.println("Правая граница: " + func.getRightDomainBorder());
35    }
36 }
```

The 'Run' window shows the output of the program:

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-ja
=== ТЕСТ 1: СОЗДАНИЕ ФУНКЦИИ ===
Точка 0: (0.0, 0.0)
Точка 1: (1.0, 1.0)
Точка 2: (2.0, 4.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

=== ТЕСТ 2: ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ===
f(-1.0) = не определено
f(0.0) = 0.0
f(1.5) = 2.5
f(4.0) = 16.0
f(5.0) = не определено

=== ТЕСТ 3: ДОБАВЛЕНИЕ И УДАЛЕНИЕ ===
После добавления точки (2.5, 6.25)
Количество точек: 6
После удаления точки с индексом 1
Количество точек: 5
Левая граница: 0.0
Правая граница: 4.0

Process finished with exit code 0
```

The status bar at the bottom indicates the file is 'Ib2', the editor is in 'src' and 'Main' files, and the main method is selected. The system tray shows the time as 18:35 on 31.10.2025.