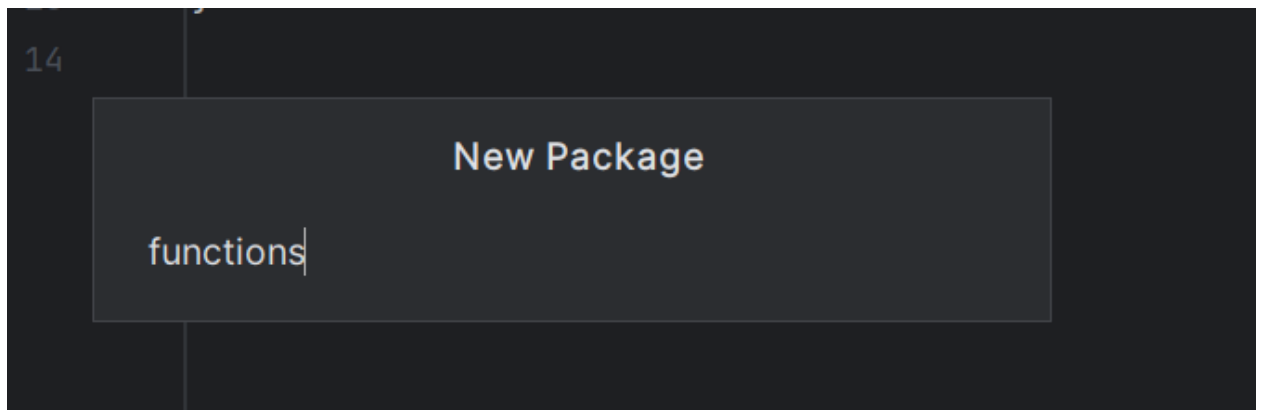
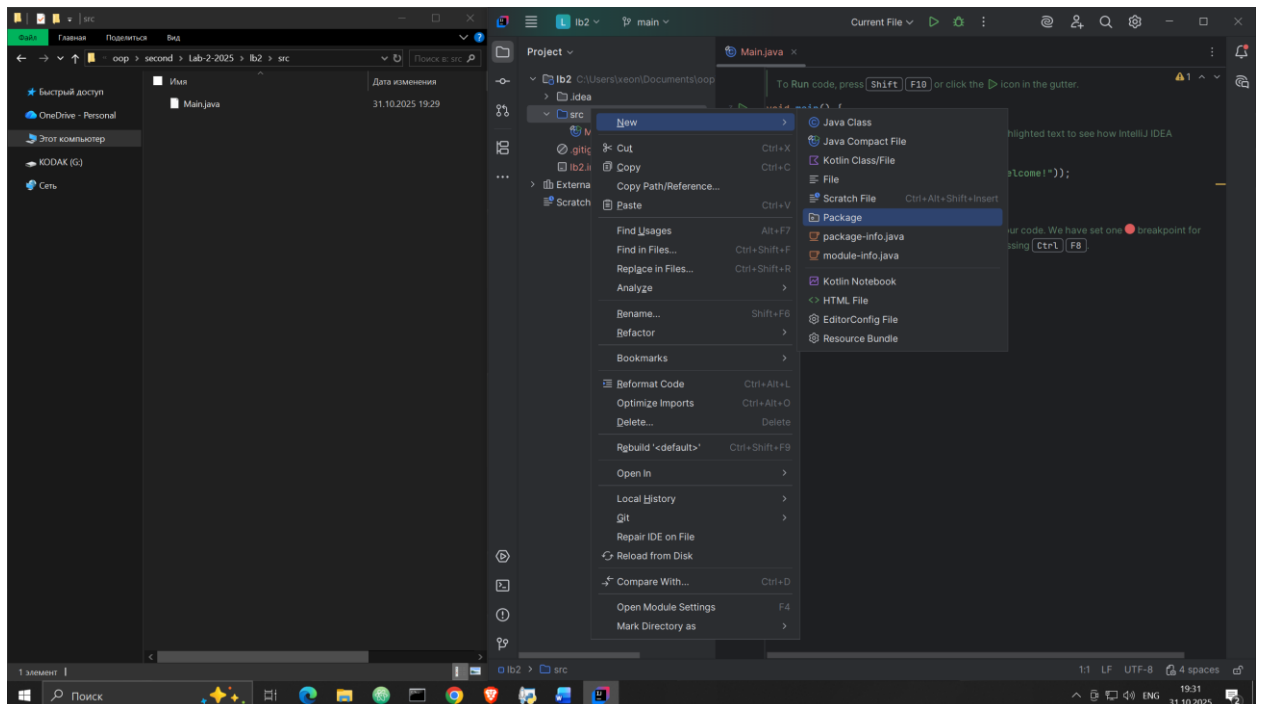


Отчёт по лабораторной работе №2
Тенигин Валерий 6204-010302D

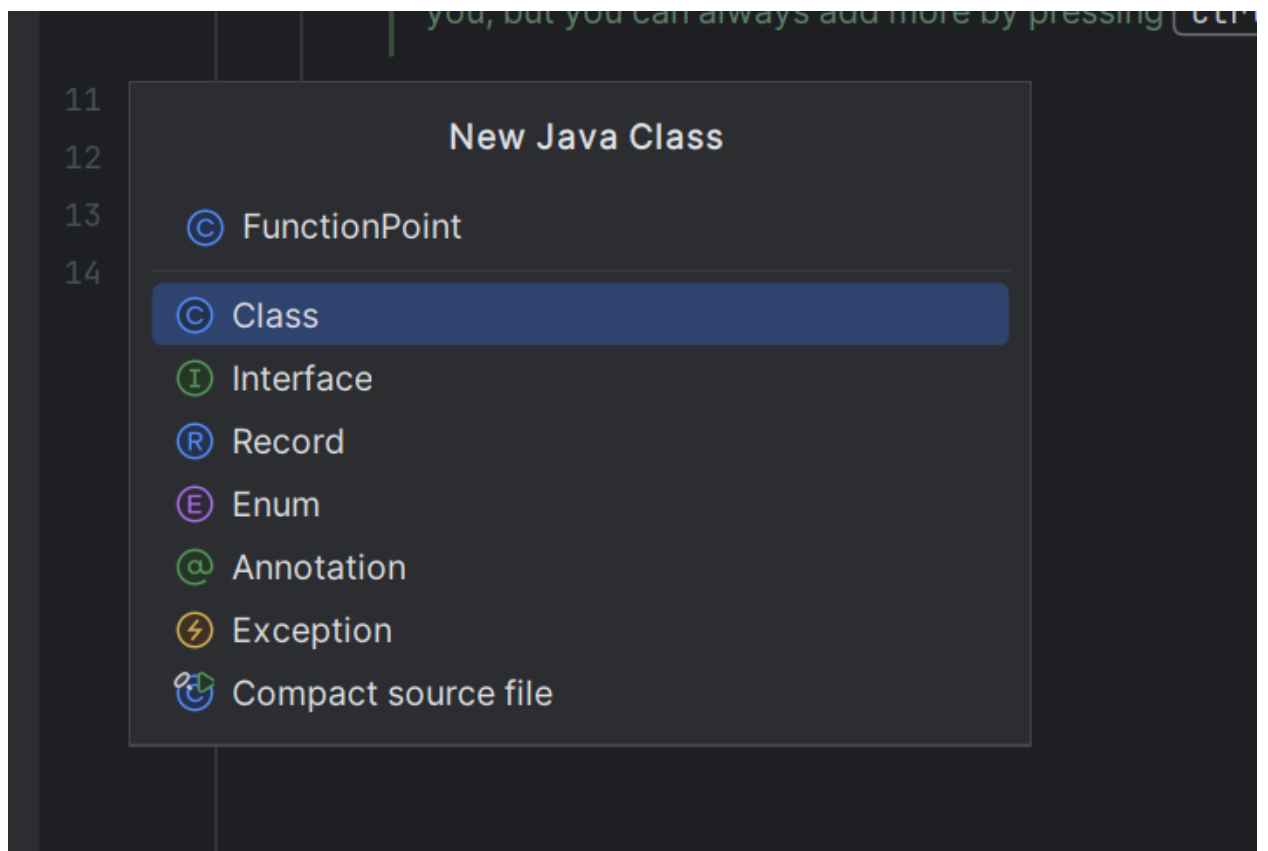
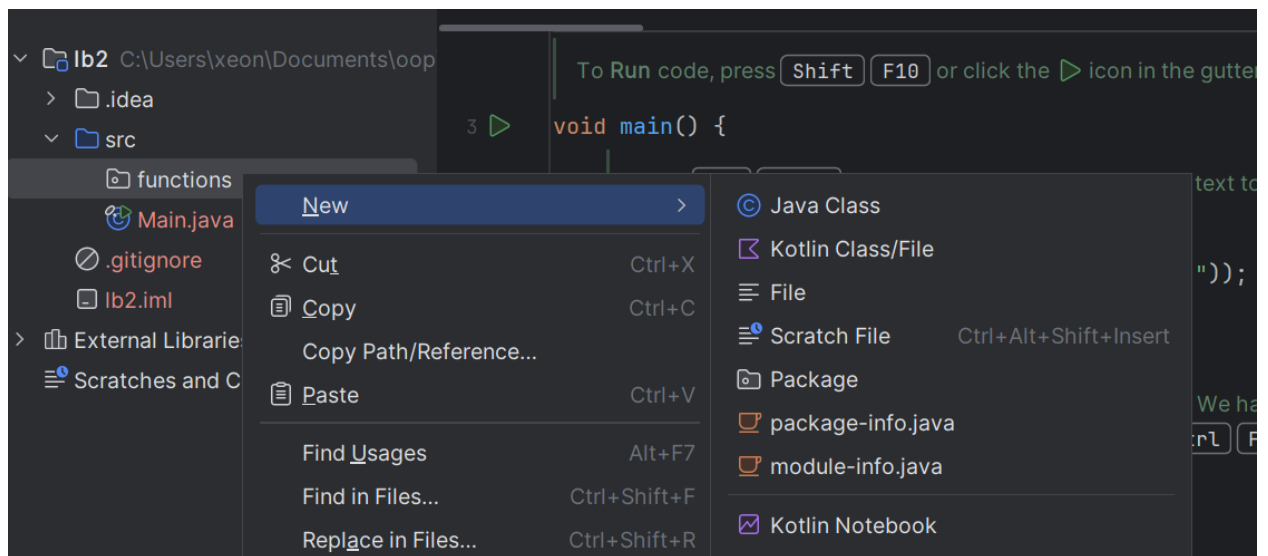
Задание 1

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).

```

1 package functions;
2
3 public class FunctionPoint { 1 usage
4     private double x; //координата точек x 3 usages
5     private double y; //точки y 3 usages
6
7     //создаёт объект точки с заданными координатами
8     public FunctionPoint(double x, double y) { 1 usage
9         this.x = x;
10        this.y = y;
11    }
12
13    //создаёт объект точки с теми же координатами, что у указанной точки
14    @ public FunctionPoint(FunctionPoint point) { no usages
15        this.x = point.x;
16        this.y = point.y;
17    }
18    //создаёт точку с координатами (0; 0)
19    public FunctionPoint() { no usages
20        this(0, 0);
21    }
22 }
  
```

Задание 3

Создаём ещё один класс TabulatedFunction

```

4     private double x; //координата точек x 3 usages
5     private double y; //точки y 3 usages
6
7     //создаёт объект точки с заданными координатами
8     public FunctionPoint(double x, double y) { 1u
9         this.x = x;
10        this.y = y;
11    }
12
13
14 @
15
16
17
18
19
20
21
22
23
  
```

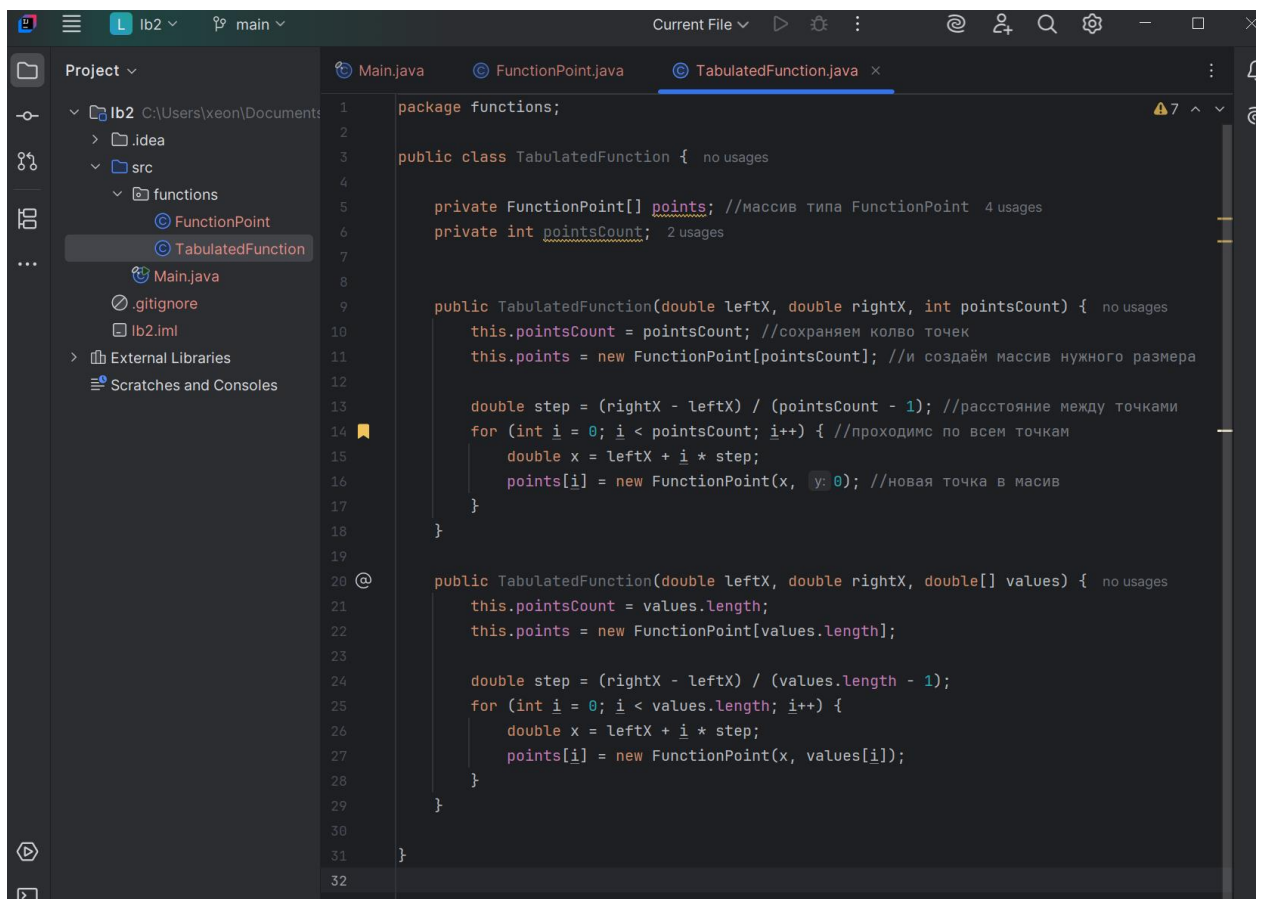
New Java Class

TabulatedFunction

- Class
- Interface
- Record
- Enum
- Annotation
- Exception
- Compact source file

Создаём следующие конструкторы:

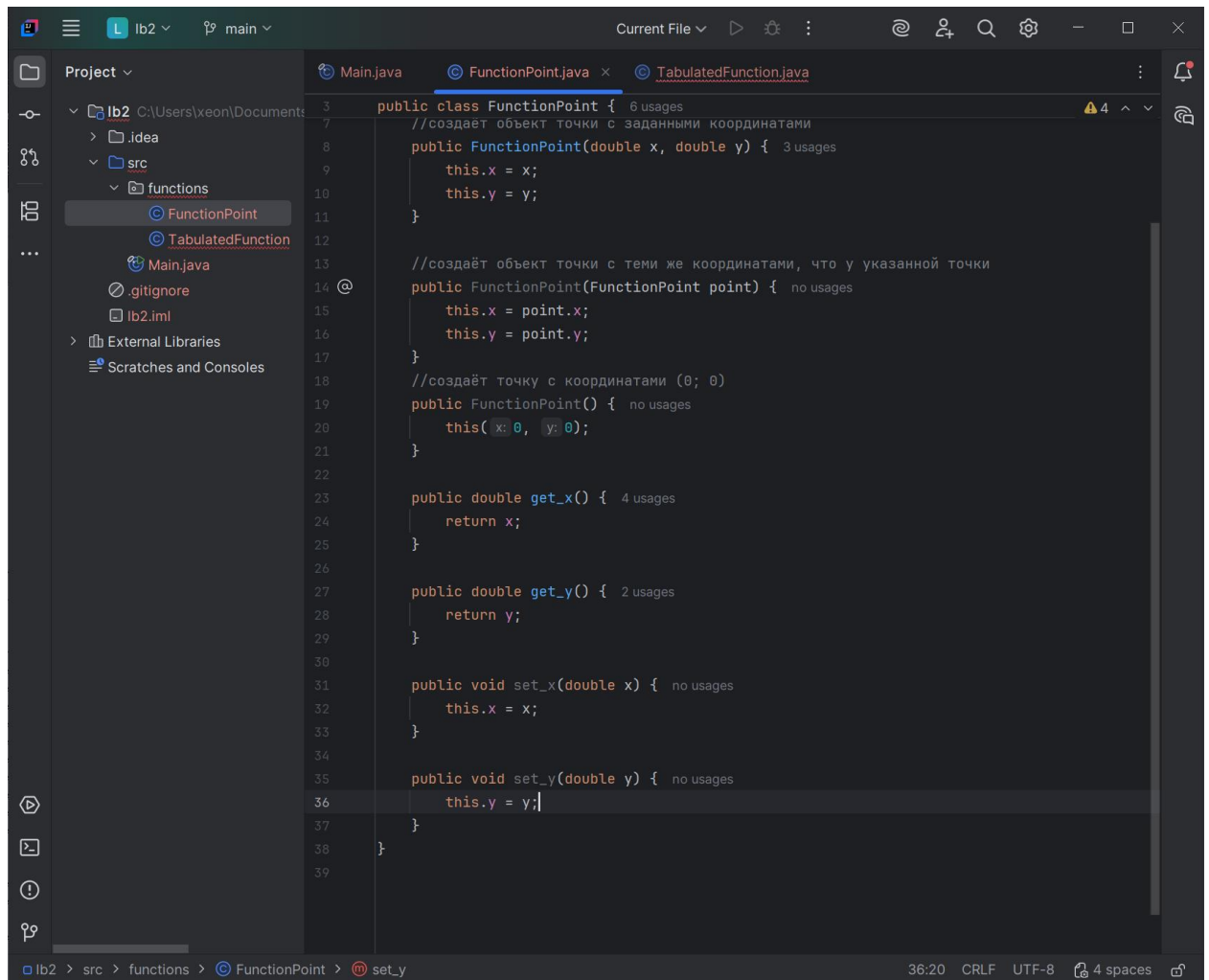
- `TabulatedFunction(double leftX, double rightX, int pointsCount)` – создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования (значения функции в точках при этом следует считать равными 0);
- `TabulatedFunction(double leftX, 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

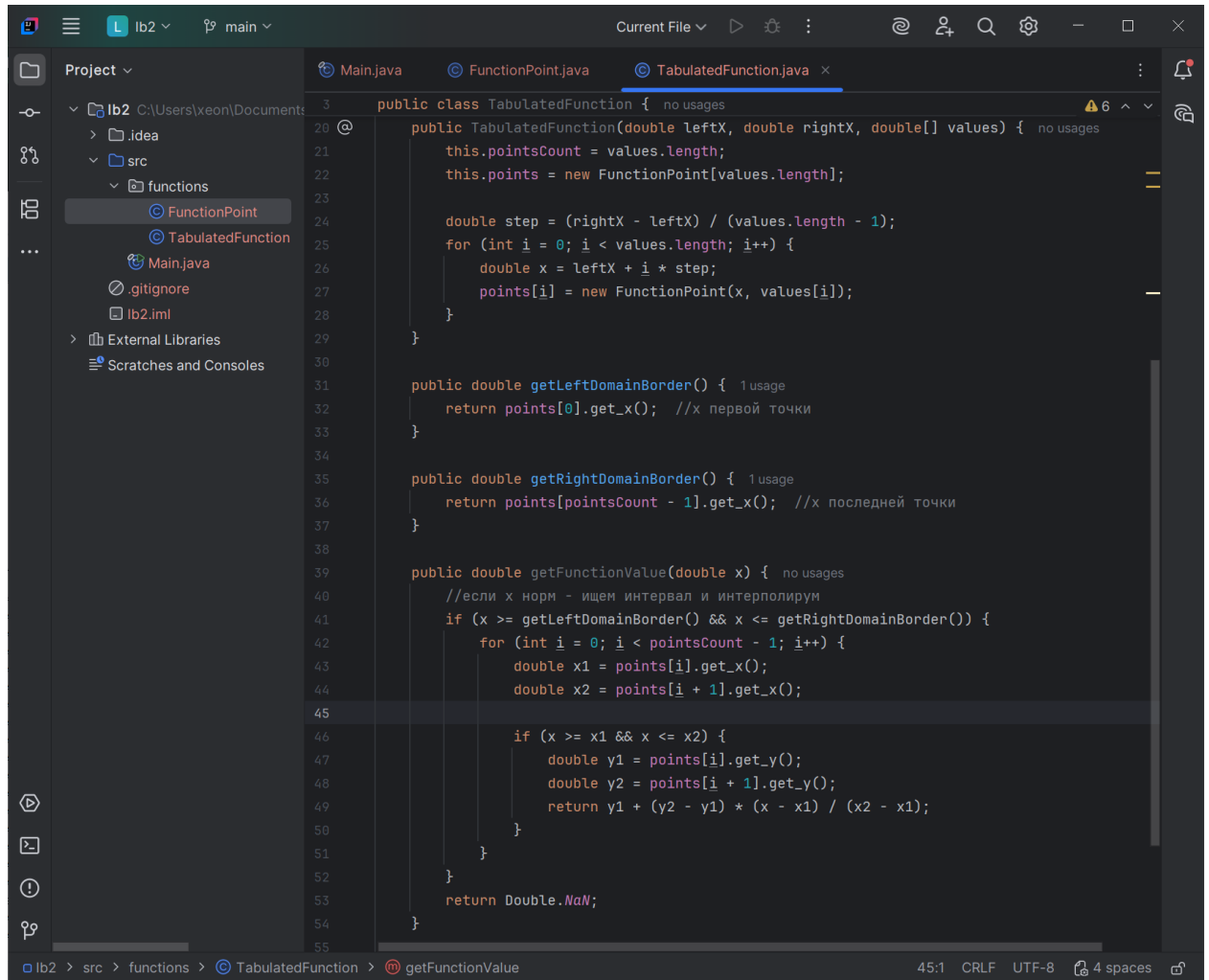
Добавил гетеры и сетеры в класс FunctionPoint



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

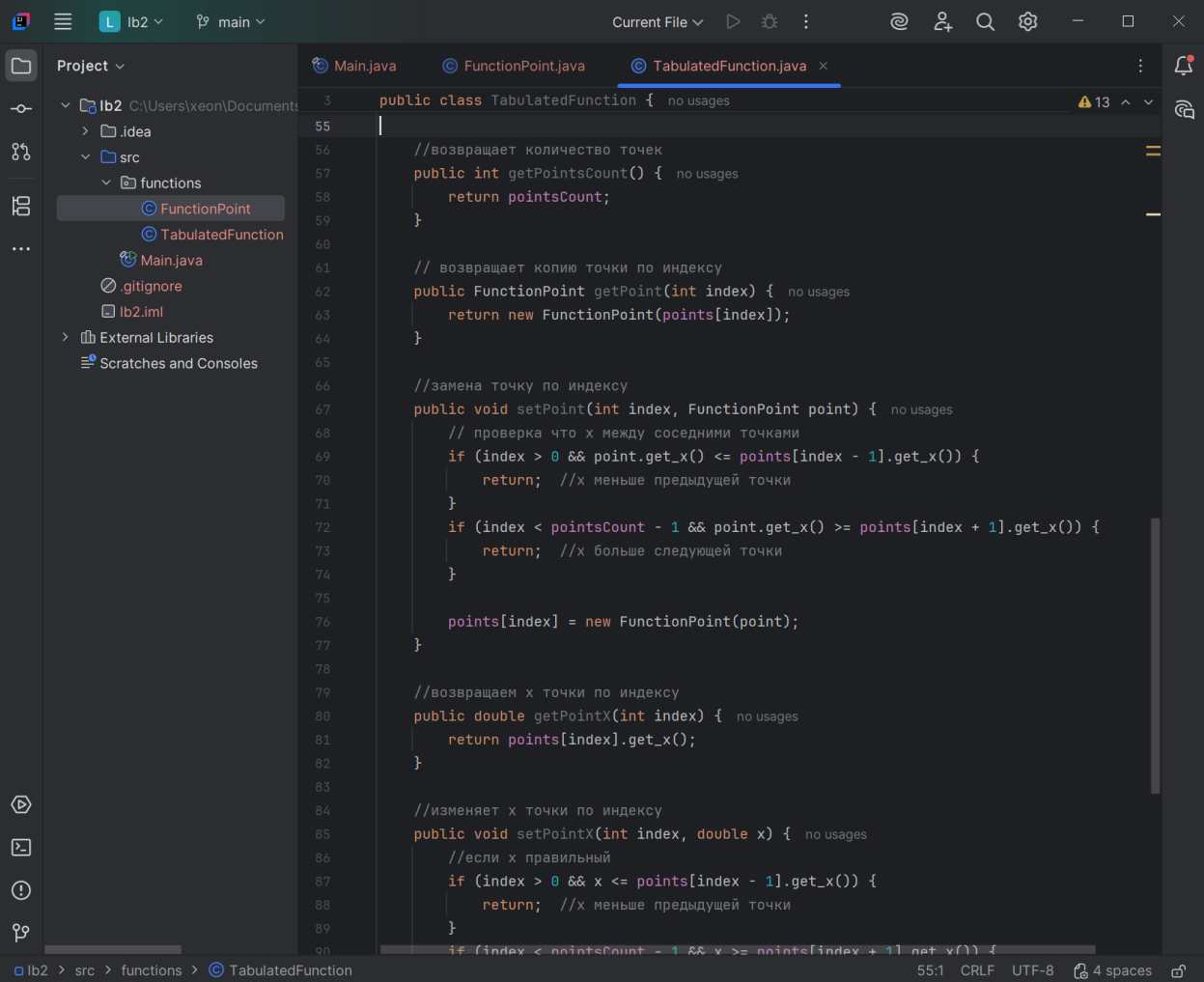


The screenshot shows an IDE with a project named 'lb2' and a source file 'TabulatedFunction.java'. The code defines a class 'TabulatedFunction' with a constructor and several methods for interpolating a function from a set of points.

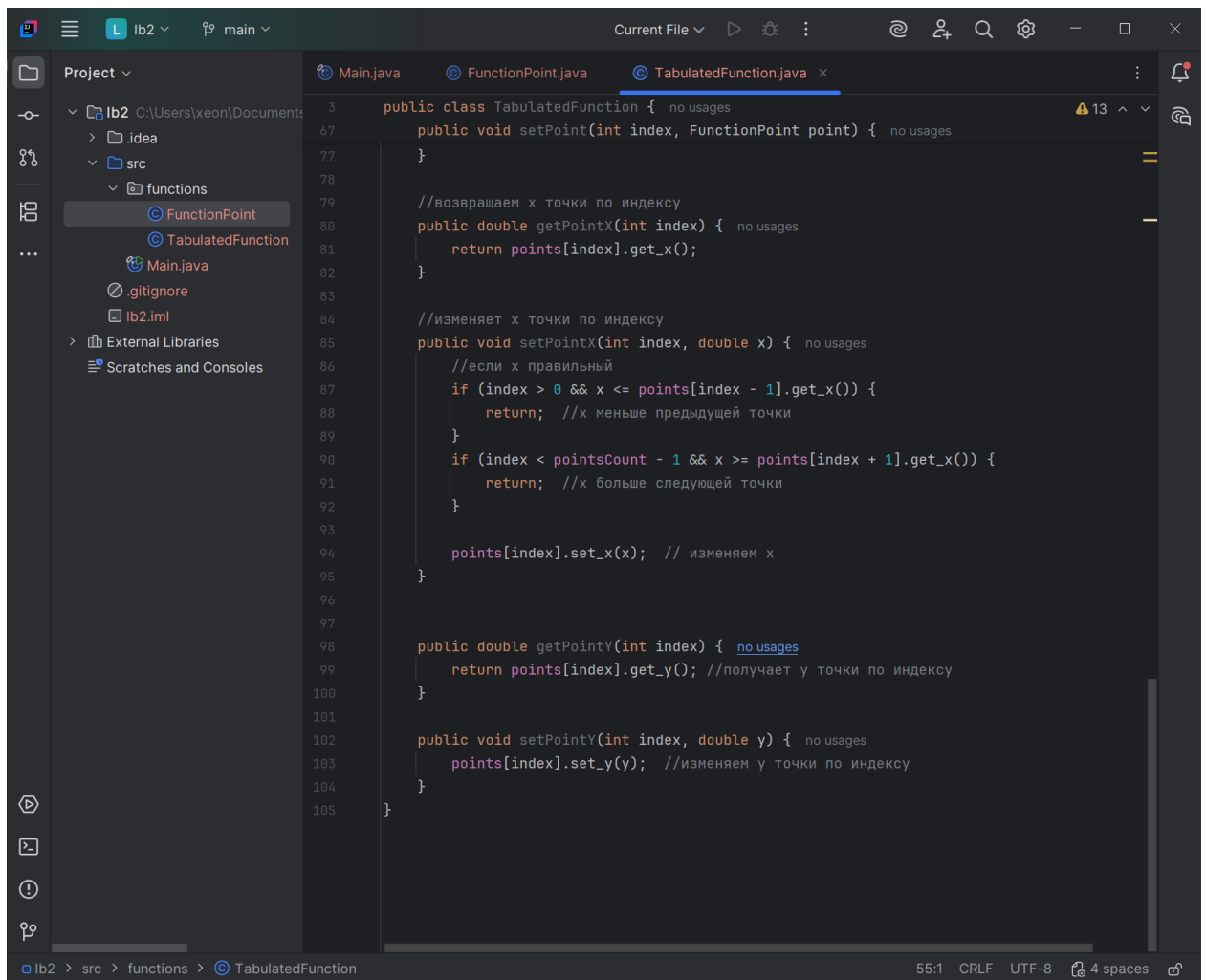
```
public class TabulatedFunction {  
    public TabulatedFunction(double leftX, double rightX, double[] values) {  
        this.pointsCount = values.length;  
        this.points = new FunctionPoint[values.length];  
  
        double step = (rightX - leftX) / (values.length - 1);  
        for (int i = 0; i < values.length; i++) {  
            double x = leftX + i * step;  
            points[i] = new FunctionPoint(x, values[i]);  
        }  
    }  
  
    public double getLeftDomainBorder() {  
        return points[0].get_x();  
    }  
  
    public double getRightDomainBorder() {  
        return points[pointsCount - 1].get_x();  
    }  
  
    public double getFunctionValue(double x) {  
        //если x норм - ищем интервал и интерполируем  
        if (x >= getLeftDomainBorder() && x <= getRightDomainBorder()) {  
            for (int i = 0; i < pointsCount - 1; i++) {  
                double x1 = points[i].get_x();  
                double x2 = points[i + 1].get_x();  
  
                if (x >= x1 && x <= x2) {  
                    double y1 = points[i].get_y();  
                    double y2 = points[i + 1].get_y();  
                    return y1 + (y2 - y1) * (x - x1) / (x2 - x1);  
                }  
            }  
        }  
        return Double.NaN;  
    }  
}
```

Задание 5

Добавлены методы для работы с точками табулированной функции согласно заданию



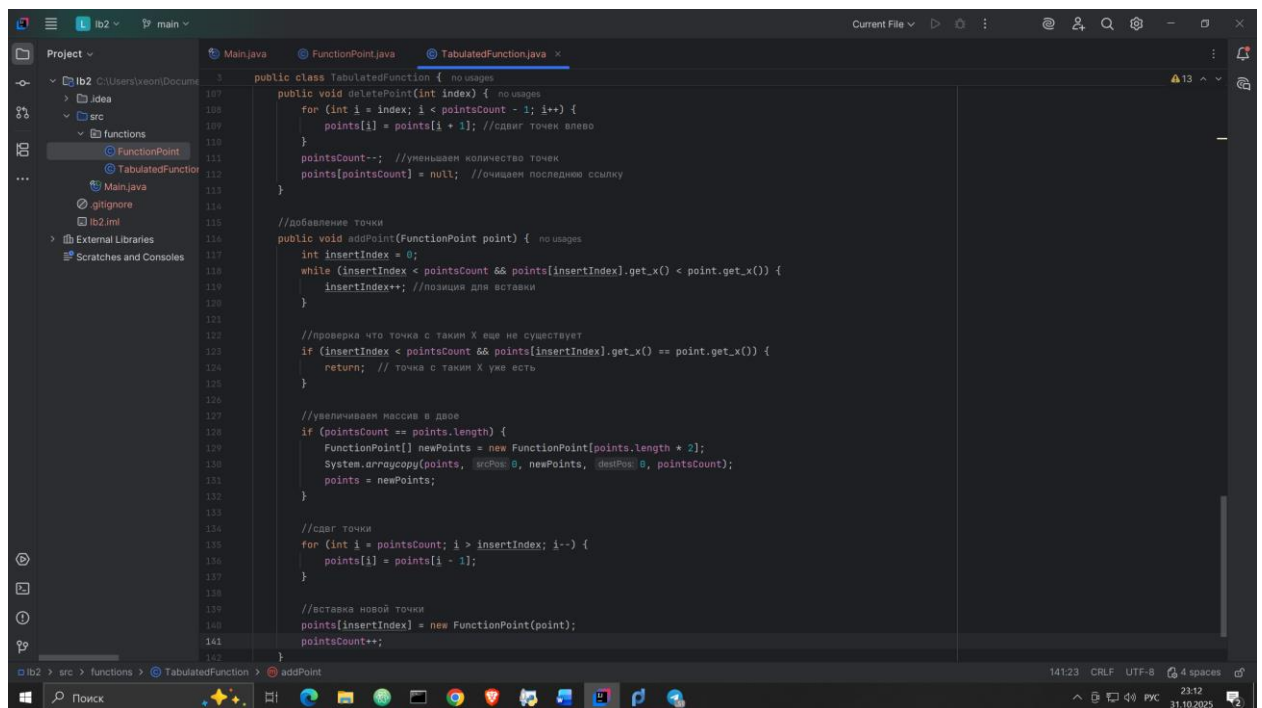
```
55 |  
56 |  
57 | //возвращает количество точек  
58 | public int getPointsCount() { no usages  
59 |     return pointsCount;  
60 | }  
61 |  
62 | // возвращает копию точки по индексу  
63 | public FunctionPoint getPoint(int index) { no usages  
64 |     return new FunctionPoint(points[index]);  
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:

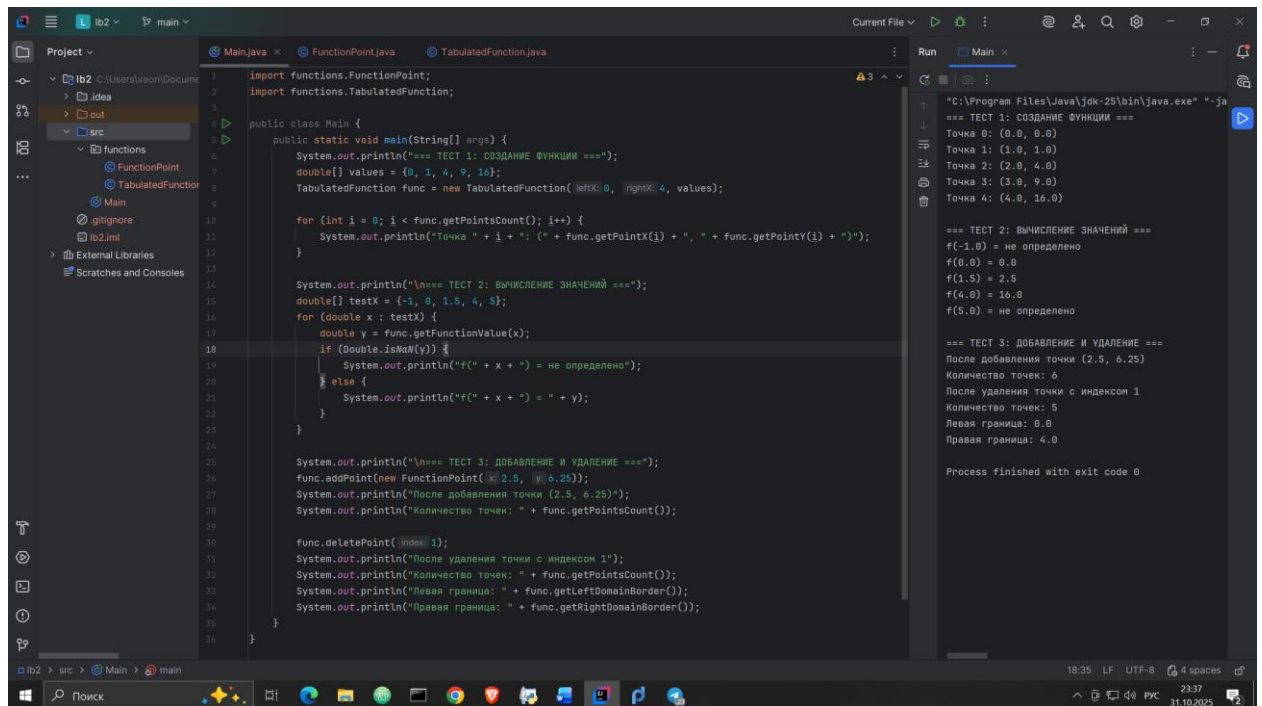
- Метод `void deletePoint(int index)` должен удалять заданную точку табулированной функции.
- Метод `void addPoint(FunctionPoint point)` должен добавлять новую точку табулированной функции.



```
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' folder contains three files: 'FunctionPoint.java', 'TabulatedFunction.java', and 'Main.java'. The 'Main.java' file is open and shows 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

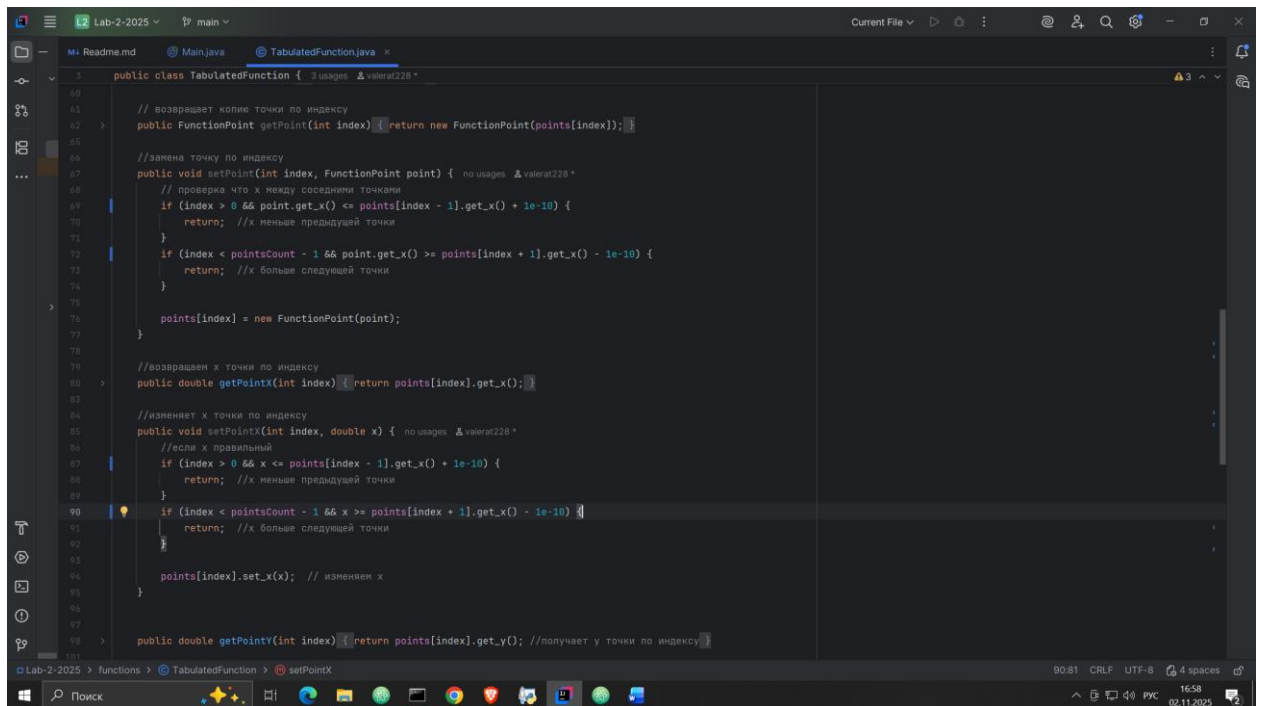
=== ТЕСТ 3: ДОБАВЛЕНИЕ И УДАЛЕНИЕ ===
После добавления точки (2.5, 6.25)
Количество точек: 6
После удаления точки с индексом 1
Количество точек: 5
Левая граница: 0.0
Правая граница: 4.0

Process finished with exit code 0
```

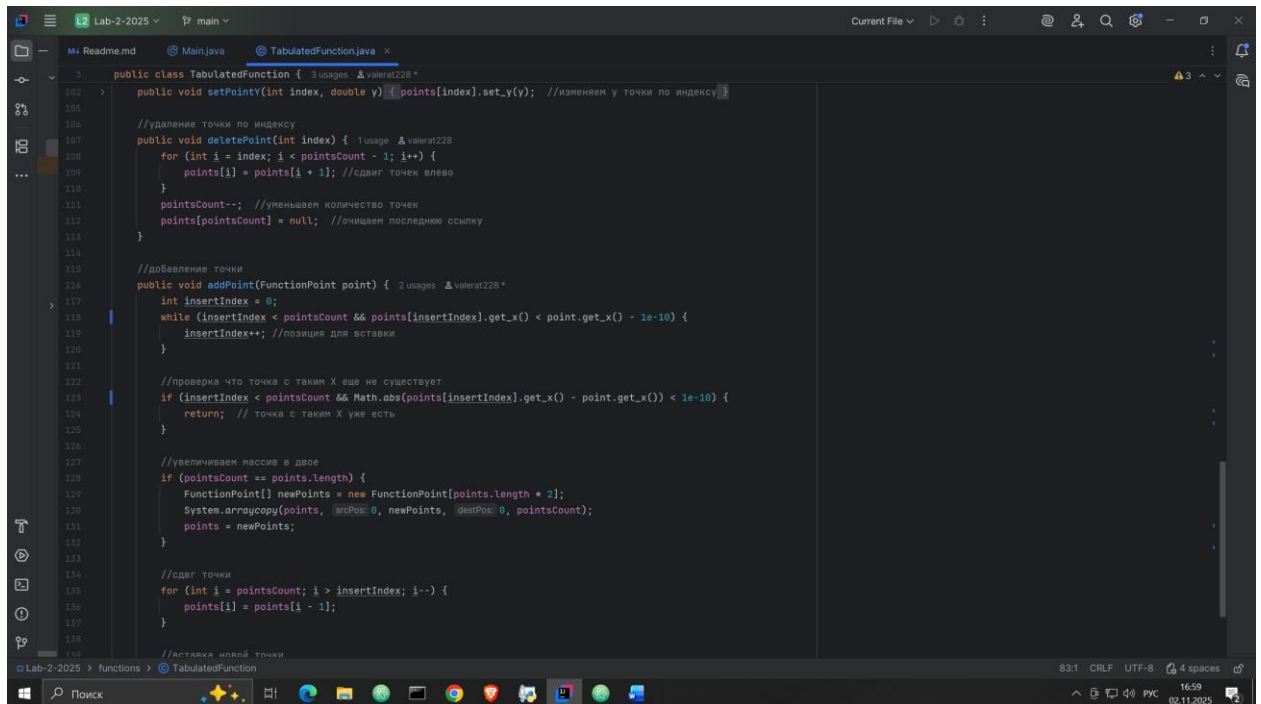
The status bar at the bottom indicates the file is 'Ib2', the editor is in 'src' and 'Main' files, and the main method is selected. The system clock shows 18:35 on 31.10.2025.

Доработанный код 1

Стал использовать эпсилон при сравнении



```
3 public class TabulatedFunction { 3 usages Δvalerat228 *
60
61 // возвращает копию точки по индексу
62 public FunctionPoint getPoint(int index) { return new FunctionPoint(points[index]); }
63
64 //замена точки по индексу
65 public void setPoint(int index, FunctionPoint point) { no usages Δvalerat228 *
66 // проверка что x между соседними точками
67 if (index > 0 && point.getX() <= points[index - 1].getX() + 1e-10) {
68 return; //x меньше предыдущей точки
69 }
70
71 if (index < pointsCount - 1 && point.getX() >= points[index + 1].getX() - 1e-10) {
72 return; //x больше следующей точки
73 }
74
75 points[index] = new FunctionPoint(point);
76
77 }
78
79 //возвращаем x точки по индексу
80 public double getPointX(int index) { return points[index].getX(); }
81
82 //изменяет x точки по индексу
83 public void setPointX(int index, double x) { no usages Δvalerat228 *
84 //если x правильный
85 if (index > 0 && x <= points[index - 1].getX() + 1e-10) {
86 return; //x меньше предыдущей точки
87 }
88
89 if (index < pointsCount - 1 && x >= points[index + 1].getX() - 1e-10) {
90 return; //x больше следующей точки
91 }
92
93 points[index].setX(x); // изменяем x
94
95 }
96
97 public double getPointY(int index) { return points[index].getY(); //получает y точки по индексу }
98
99 }
```



```
102 public void setPointY(int index, double y) { points[index].setY(y); //изменяем y точки по индексу }
103
104 //удаление точки по индексу
105 public void deletePoint(int index) { 1 usage Δvalerat228
106 for (int i = index; i < pointsCount - 1; i++) {
107 points[i] = points[i + 1]; //сдвиг точек влево
108 }
109 pointsCount--; //уменьшаем количество точек
110 points[pointsCount] = null; //очищаем последнюю ссылку
111 }
112
113 //добавление точки
114 public void addPoint(FunctionPoint point) { 2 usages Δvalerat228 *
115 int insertIndex = 0;
116 while (insertIndex < pointsCount && points[insertIndex].getX() < point.getX() - 1e-10) {
117 insertIndex++; //позиция для вставки
118 }
119
120 //проверка что точка с таким X еще не существует
121 if (insertIndex < pointsCount && Math.abs(points[insertIndex].getX() - point.getX()) < 1e-10) {
122 return; // точка с таким X уже есть
123 }
124
125 //увеличиваем массив в двое
126 if (pointsCount == points.length) {
127 FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
128 System.arraycopy(points, 0, newPoints, 0, pointsCount);
129 points = newPoints;
130 }
131
132 //сдвиг точки
133 for (int i = pointsCount; i > insertIndex; i--) {
134 points[i] = points[i - 1];
135 }
136
137 //вставка новой точки
138 points[insertIndex] = point;
139 pointsCount++;
140 }
```

Изменил проверку работы

```
import functions.FunctionPoint;
import functions.TabulatedFunction;

public class Main {
    public static void main(String[] args) {
        System.out.println("=== ТЕСТ 1: СОЗДАНИЕ ФУНКЦИИ ===");
        double[] values = {0, 1, 4, 9, 16};
        TabulatedFunction func = new TabulatedFunction(0, 4, values);

        // Выводим все точки после создания
        System.out.println("Точки после создания функции:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }

        System.out.println("\n=== ТЕСТ 2: ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ===");
        double[] testX = {-1, 0, 1.5, 4, 5};
        for (double x : testX) {
            double y = func.getFunctionValue(x);
            if (Double.isNaN(y)) {
                System.out.println("f(" + x + ") = не определено");
            } else {
                System.out.println("f(" + x + ") = " + y);
            }
        }

        System.out.println("\n=== ТЕСТ 3: ДОБАВЛЕНИЕ И УДАЛЕНИЕ ===");

        // Добавление точки
        System.out.println("Добавляем точку (2.5, 6.25)");
        func.addPoint(new FunctionPoint(x: 2.5, y: 6.25));
        System.out.println("Точки после добавления:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }

        System.out.println("\nКоличество точек: " + func.getPointsCount());

        // Удаление точки
        System.out.println("Удаляем точку с индексом 1");
        func.deletePoint(index: 1);
        System.out.println("Точки после удаления:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }

        System.out.println("\nКоличество точек: " + func.getPointsCount());

        // Изменение Y точки
        System.out.println("Изменяем Y точки с индексом 2 на 10.0");
        func.setPoint(index: 2, y: 10.0);
        System.out.println("Точки после изменения Y:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }

        // Попытка добавить точку с существующим X
        System.out.println("Пытаемся добавить точку (2.0, 100.0) - должна быть отклонена");
        func.addPoint(new FunctionPoint(x: 2.0, y: 100.0));
        System.out.println("Точки после попытки добавления дубликата:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }

        // Изменение X точки
        System.out.println("\n=== ТЕСТ 4: ИЗМЕНЕНИЕ X ТОЧЕК ===");
        System.out.println("Изменяем X точки с индексом 1 с " + func.getPointX(index: 1) + " на 1.8");
        func.setPoint(index: 1, x: 1.8);
        System.out.println("Точки после изменения X точки 1:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }

        System.out.println("\nПытаемся изменить X точки 0 на -1.0 (должно быть отклонено)");
        func.setPoint(index: 0, x: -1.0);
        System.out.println("Точки после НЕУДАЧНОЙ попытки изменения X:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.println("Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");
        }
    }
}
```

```
=== ТЕСТ 1: СОЗДАНИЕ ФУНКЦИИ ===
Точки после создания 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 ===
Добавляем точку (2.5, 6.25)
Точки после добавления:
Точка 0: (0.0, 0.0)
Точка 1: (1.0, 1.0)
Точка 2: (2.0, 4.0)
Точка 3: (2.5, 6.25)
Точка 4: (3.0, 9.0)
Точка 5: (4.0, 16.0)
Количество точек: 6

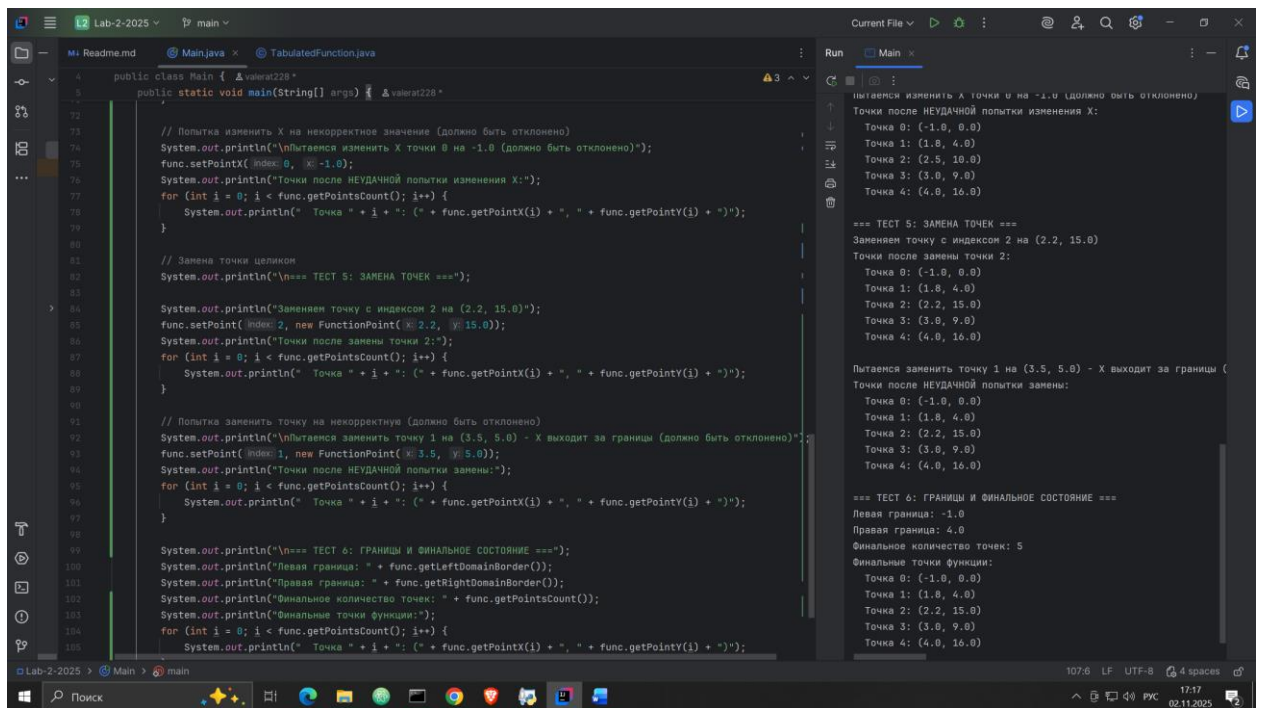
Удаляем точку с индексом 1
Точки после удаления:
Точка 0: (0.0, 0.0)
Точка 1: (2.0, 4.0)
Точка 2: (2.5, 6.25)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)
Количество точек: 5

Изменяем Y точки с индексом 2 на 10.0
Точки после изменения Y:
Точка 0: (0.0, 0.0)
Точка 1: (2.0, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

Пытаемся добавить точку (2.0, 100.0) - должна быть отклонена
Точки после попытки добавления дубликата:
Точка 0: (0.0, 0.0)
Точка 1: (2.0, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

=== ТЕСТ 4: ИЗМЕНЕНИЕ X ТОЧЕК ===
Изменяем X точки с индексом 1 с 2.0 на 1.8
Точки после изменения X точки 1:
Точка 0: (0.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

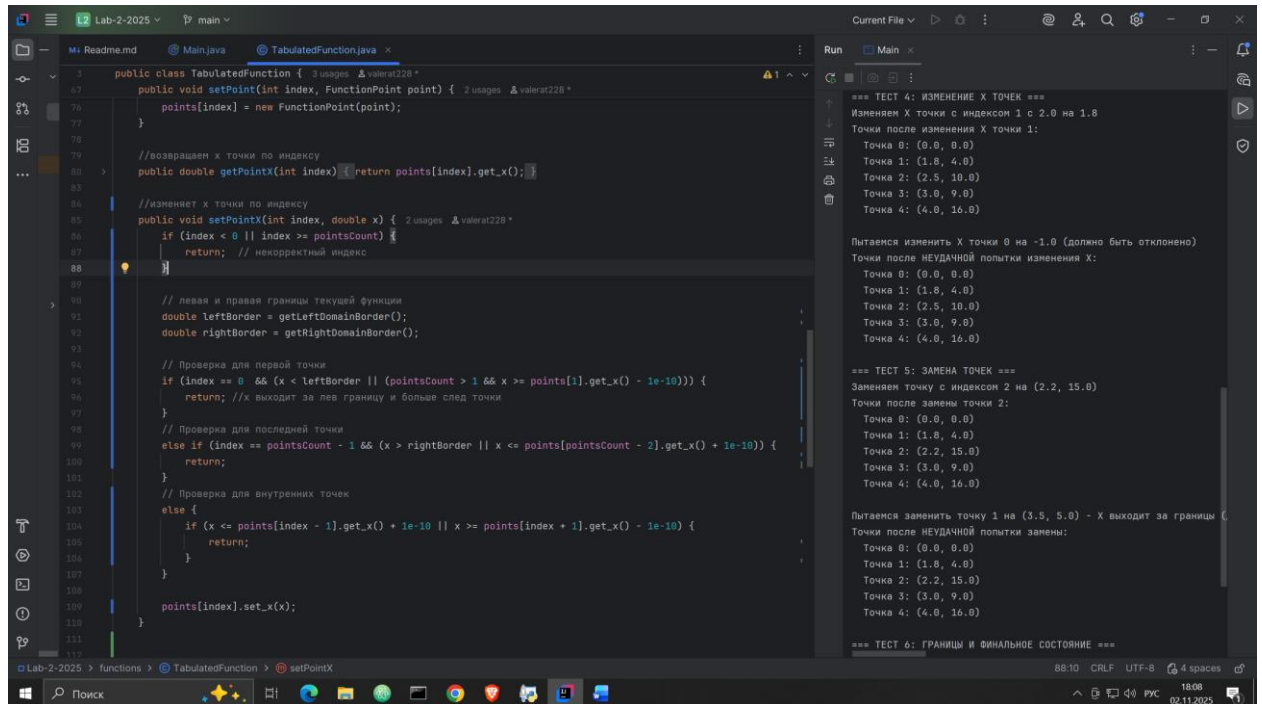
Пытаемся изменить X точки 0 на -1.0 (должно быть отклонено)
Точки после НЕУДАЧНОЙ попытки изменения X:
Точка 0: (-1.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)
```



Найдена ошибка

```
Пытаемся изменить X точки 0 на -1.0 (должно быть отклонено)
Точки после НЕУДАЧНОЙ попытки изменения X:
Точка 0: (-1.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)
```

Изменённый код для устранения ошибки



```
public class TabulatedFunction {
    public void setPoint(int index, FunctionPoint point) {
        points[index] = new FunctionPoint(point);
    }

    //возвращаем x точки по индексу
    public double getPointX(int index) { return points[index].get_x(); }

    //изменяет x точки по индексу
    public void setPointX(int index, double x) {
        if (index < 0 || index >= pointsCount) {
            return; // некорректный индекс
        }

        // левая и правая границы текущей функции
        double leftBorder = getLeftDomainBorder();
        double rightBorder = getRightDomainBorder();

        // Проверка для первой точки
        if (index == 0 && (x < leftBorder || (pointsCount > 1 && x >= points[1].get_x() - 1e-10))) {
            return; // x выходит за лев границу и больше след точки
        }

        // Проверка для последней точки
        else if (index == pointsCount - 1 && (x > rightBorder || x <= points[pointsCount - 2].get_x() + 1e-10)) {
            return;
        }

        // Проверка для внутренних точек
        else {
            if (x <= points[index - 1].get_x() + 1e-10 || x >= points[index + 1].get_x() - 1e-10) {
                return;
            }
        }

        points[index].set_x(x);
    }
}
```

=== TEST 4: ИЗМЕНЕНИЕ X ТОЧЕК ===
Изменяем X точки с индексом 1 с 2.0 на 1.8
Точки после изменения X точки 1:
Точка 0: (0.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

Пытаемся изменить X точки 0 на -1.0 (должно быть отклонено)
Точка после НЕУДАЧНОЙ попытки изменения X:
Точка 0: (0.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.5, 10.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

=== TEST 5: ЗАМЕНА ТОЧЕК ===
Заменяем точку с индексом 2 на (2.2, 15.0)
Точка после замены точки 2:
Точка 0: (0.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.2, 15.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

Пытаемся заменить точку 1 на (3.5, 5.0) - X выходит за границы
Точка после НЕУДАЧНОЙ попытки замены:
Точка 0: (0.0, 0.0)
Точка 1: (1.8, 4.0)
Точка 2: (2.2, 15.0)
Точка 3: (3.0, 9.0)
Точка 4: (4.0, 16.0)

=== TEST 6: ГРАНИЦЫ И ФИНАЛЬНОЕ СОСТОЯНИЕ ===