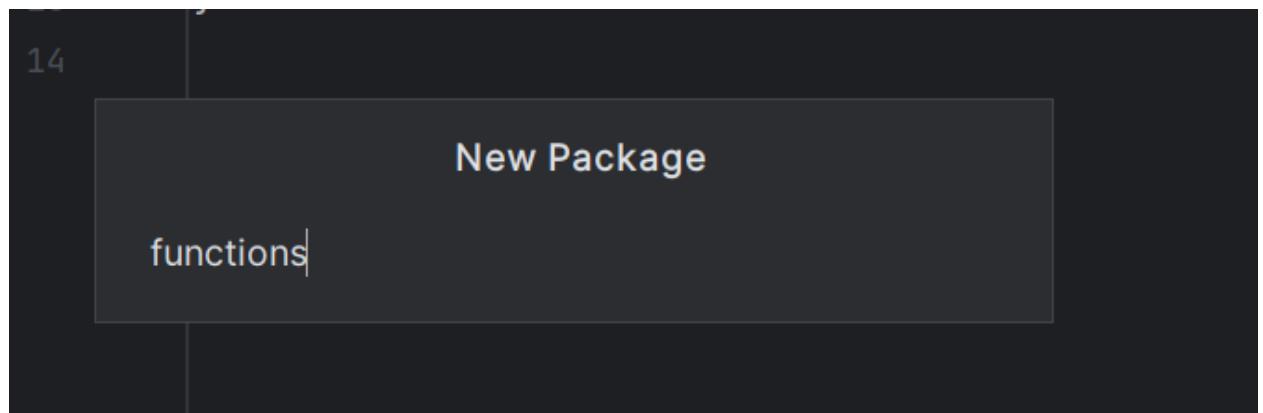
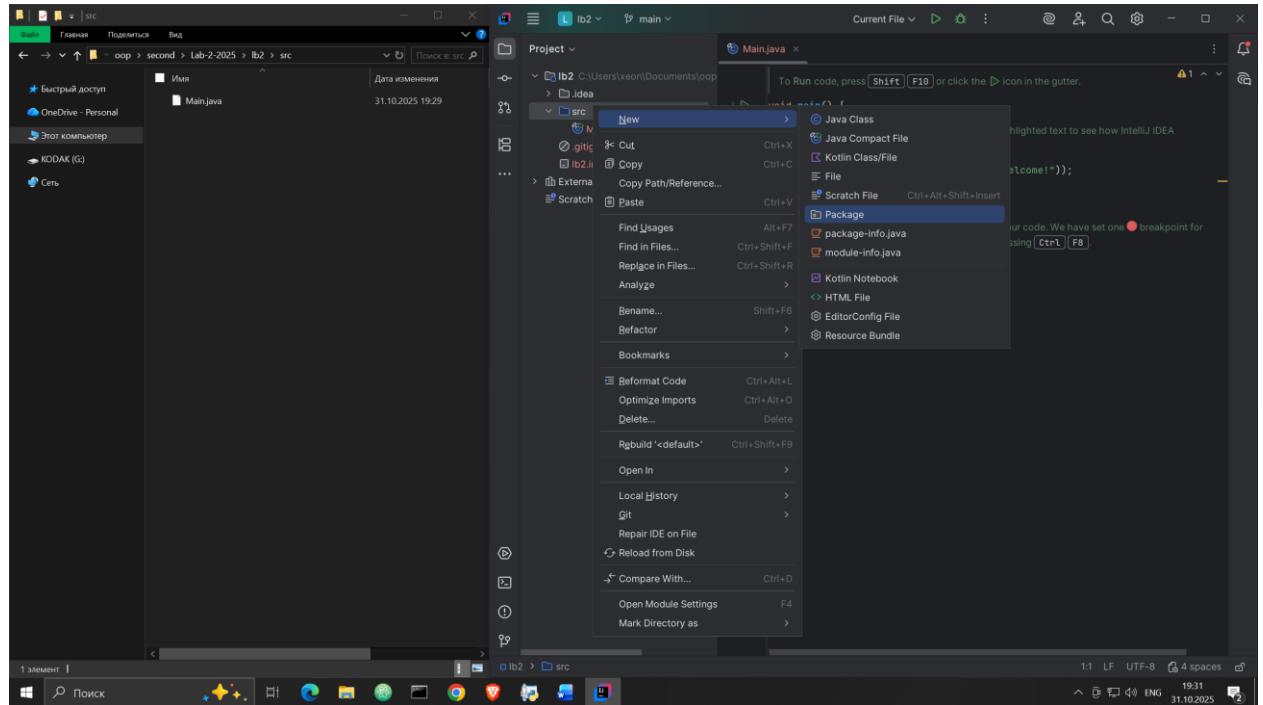


Отчёт по лабораторной работе №2

Тенигин Валерий 6204-010302D

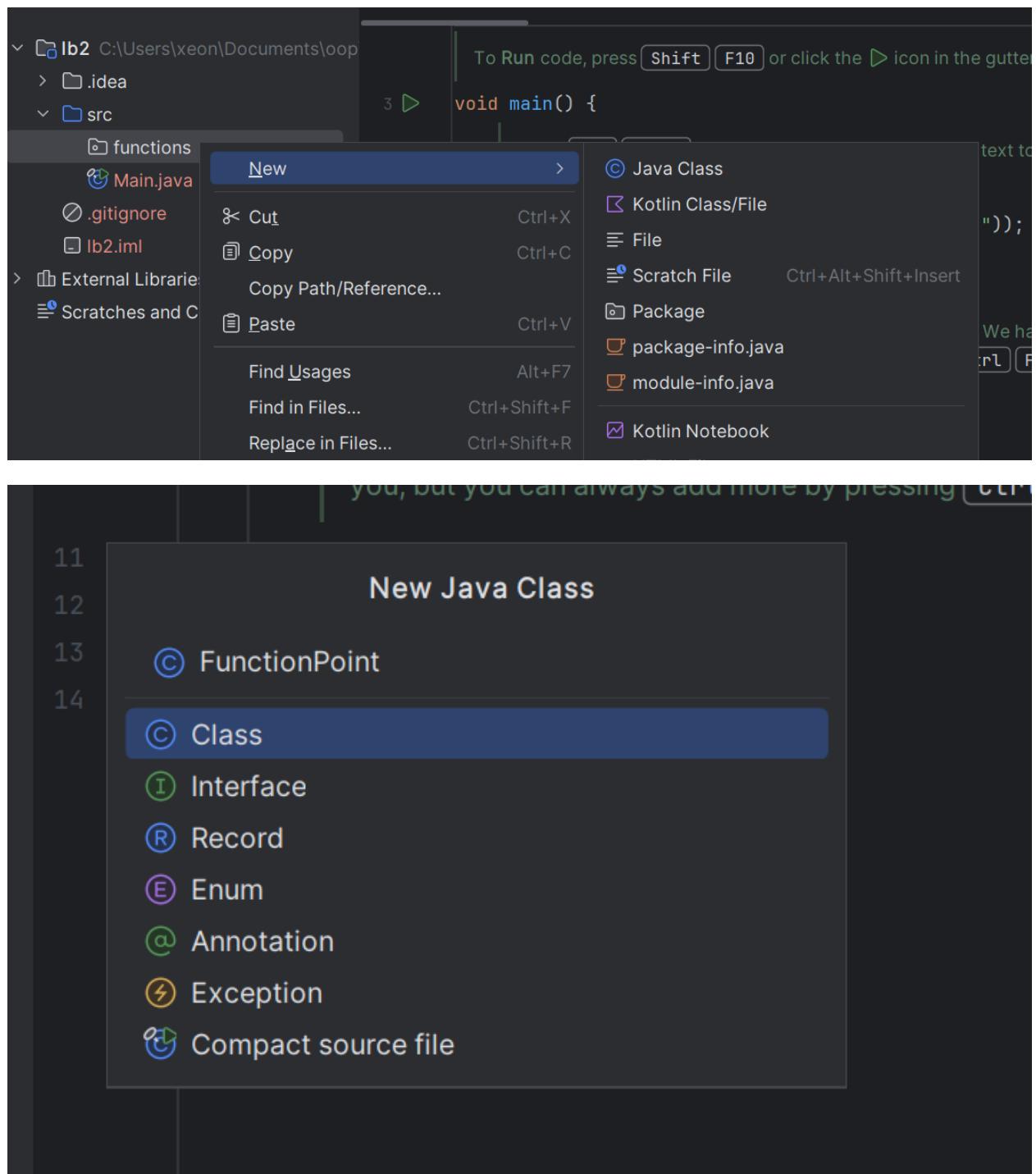
Задание 1

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).

```

package functions;

public class FunctionPoint {
    private double x; //координата точик x 3 usages
    private double y; //точки y 3 usages

    //создаёт объект точки с заданными координатами
    public FunctionPoint(double x, double y) {
        this.x = x;
        this.y = y;
    }

    //создаёт объект точки с теми же координатами, что у указанной точки
    public FunctionPoint(FunctionPoint point) {
        this.x = point.x;
        this.y = point.y;
    }

    //создаёт точку с координатами (0; 0)
    public FunctionPoint() {
        this(0, 0);
    }
}

```

Задание 3

Создаём ещё один класс TabulatedFunction

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The 'src' folder contains a 'functions' package which includes 'FunctionPoint.java' and 'Main.java'. Other files like '.gitignore' and 'Ib2.iml' are also present.
- Code Editor:** The 'FunctionPoint.java' file is open, showing its current implementation.
- New Java Class Dialog:** A modal dialog titled 'New Java Class' is displayed. It lists several options: 'TabulatedFunction' (selected), 'Class' (highlighted in blue), 'Interface', 'Record', 'Enum', 'Annotation', 'Exception', and '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.

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure under 'Ib2' with packages 'src' and 'functions' containing 'FunctionPoint' and 'TabulatedFunction'. The code editor on the right shows the content of 'TabulatedFunction.java'. The code defines two constructors for the 'TabulatedFunction' class. The first constructor takes 'leftX', 'rightX', and 'pointsCount' parameters, initializes a 'FunctionPoint[] points' array and an 'int pointsCount' variable, and then loops from 0 to 'pointsCount - 1' to create 'FunctionPoint' objects with increasing 'x' values. The second constructor takes 'leftX', 'rightX', and a 'double[] values' parameter, initializes 'pointsCount' to the length of 'values', creates a 'FunctionPoint[] points' array of the same length, and then loops from 0 to 'values.length - 1' to set each point's 'x' and 'y' values based on the input 'values' array.

```
package functions;

public class TabulatedFunction { no usages

    private FunctionPoint[] points; //массив типа FunctionPoint 4 usages
    private int pointsCount; 2 usages

    public TabulatedFunction(double leftX, double rightX, int pointsCount) { no usages
        this.pointsCount = pointsCount; //сохраняем колво точек
        this.points = new FunctionPoint[pointsCount]; //и создаём массив нужного размера

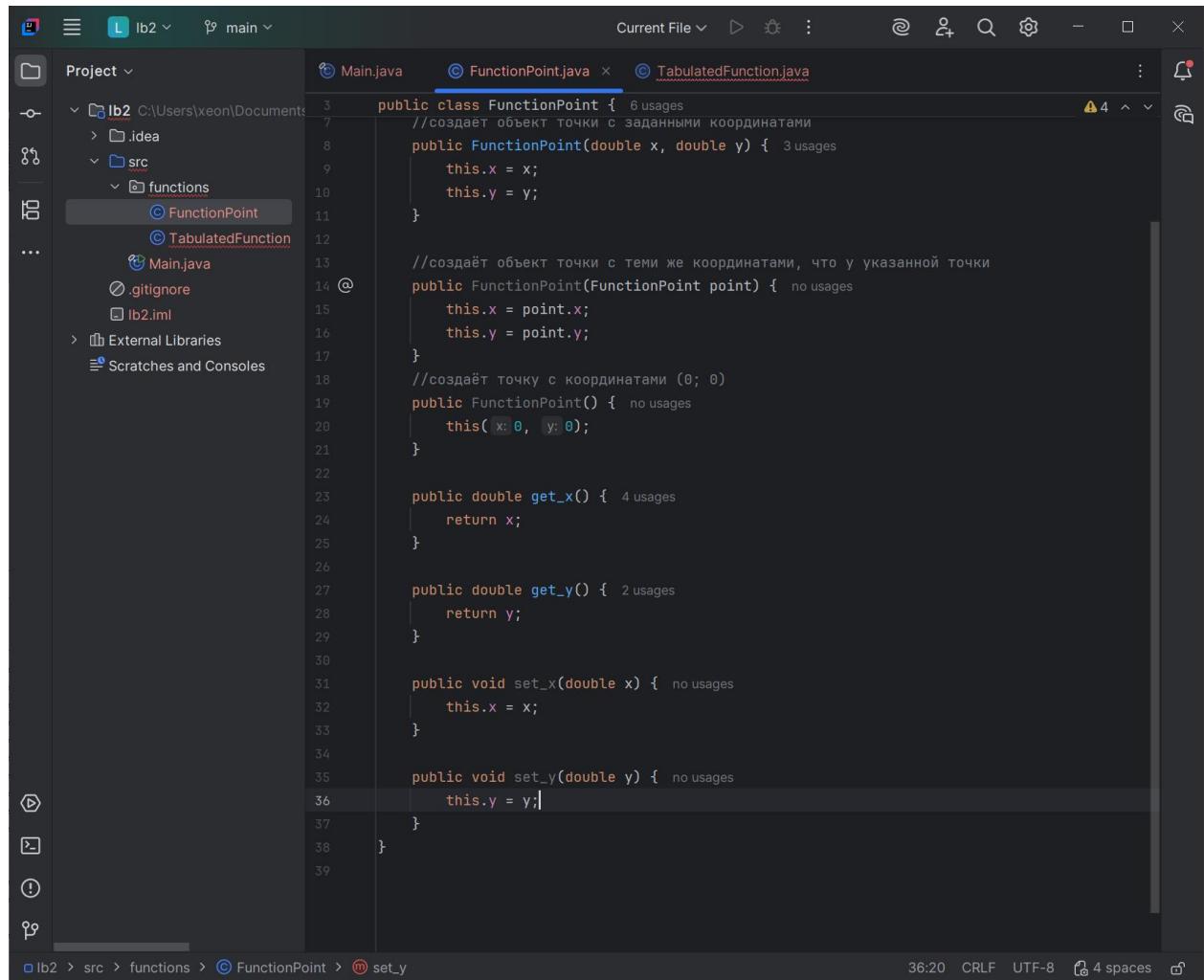
        double step = (rightX - leftX) / (pointsCount - 1); //расстояние между точками
        for (int i = 0; i < pointsCount; i++) { //проходимся по всем точкам
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, y: 0); //новая точка в массив
        }
    }

    public TabulatedFunction(double leftX, double rightX, double[] values) { no usages
        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]);
        }
    }
}
```

Задание 4

Добавил гетеры и сетеры в класс FunctionPoint



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "Ib2". The "functions" package contains three files: "FunctionPoint.java", "Main.java", and "TabulatedFunction.java". "FunctionPoint.java" is currently selected.
- Main.java:** A placeholder file with a single line of code: `import java.util.*;`.
- FunctionPoint.java:** Contains the following Java code:

```
public class FunctionPoint {    //создаёт объект точки с заданными координатами    public FunctionPoint(double x, double y) {        this.x = x;        this.y = y;    }    //создаёт объект точки с теми же координатами, что у указанной точки    public FunctionPoint(FunctionPoint point) {        this.x = point.x;        this.y = point.y;    }    //создаёт точку с координатами (0; 0)    public FunctionPoint() {        this(0, 0);    }    public double get_x() {        return x;    }    public double get_y() {        return y;    }    public void set_x(double x) {        this.x = x;    }    public void set_y(double y) {        this.y = y;    }}
```

The code editor shows line numbers from 3 to 39. There are 6 usages of the class and 3 usages of its constructor. The code is written in Java 8 style with annotations like `@` and `no usages`.

Добавлены 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 a Java code editor interface with the following details:

- Project View:** Shows the project structure under "Ib2". The "functions" package contains three files: `Main.java`, `FunctionPoint.java`, and `TabulatedFunction.java`. `FunctionPoint.java` is currently selected.
- Code Editor:** Displays the `TabulatedFunction.java` file. The code implements a tabulated function using linear interpolation between points. It includes methods for getting domain borders and calculating function values.
- Status Bar:** Shows the current file path: `Ib2 > src > functions > TabulatedFunction > getFunctionValue`. It also displays system information: 45:1 CRLF UTF-8 4 spaces.

```
public class TabulatedFunction { no usages
    public TabulatedFunction(double leftX, double rightX, double[] values) { no usages
        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() { 1 usage
        return points[0].get_x(); //x первой точки
    }

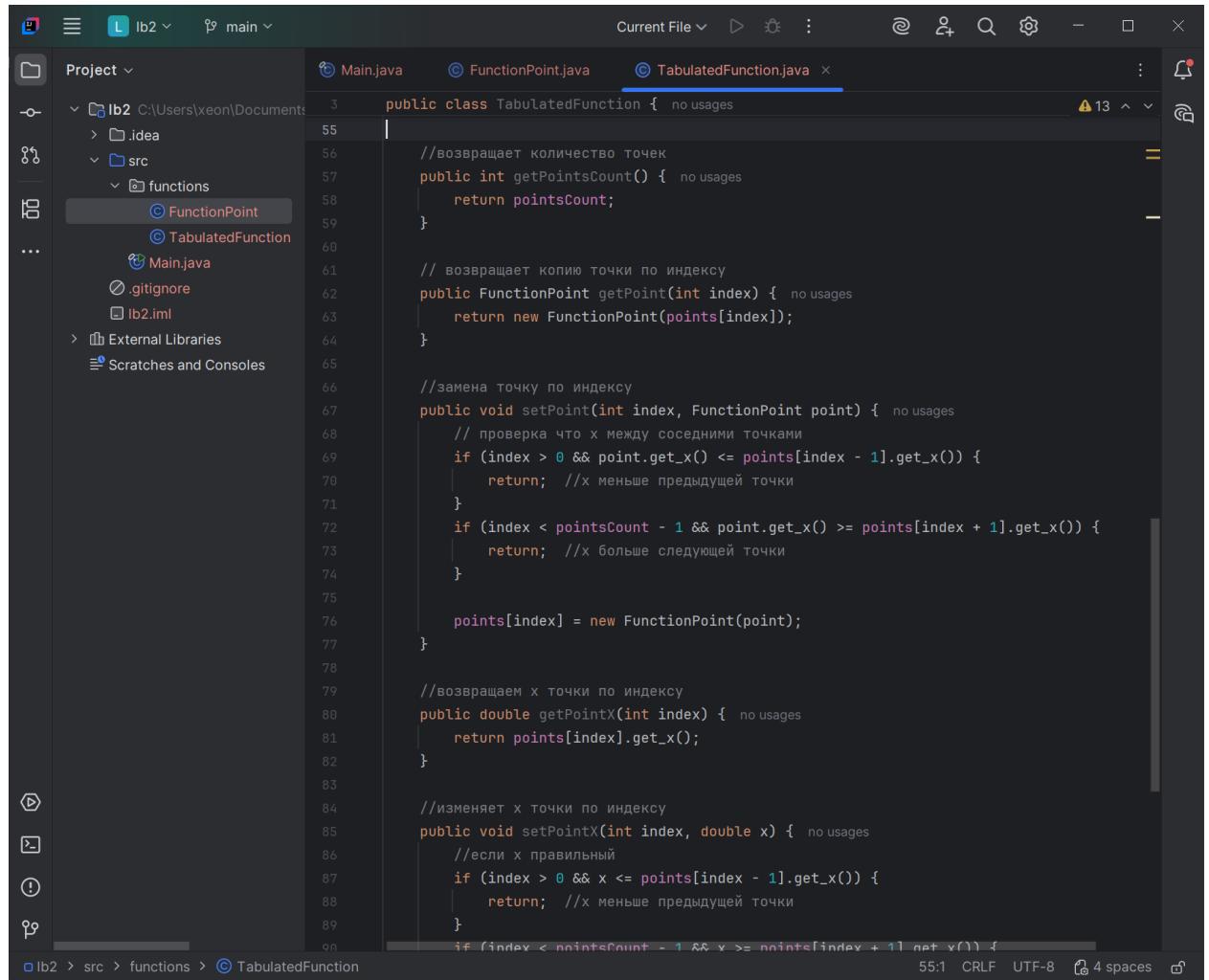
    public double getRightDomainBorder() { 1 usage
        return points[pointsCount - 1].get_x(); //x последней точки
    }

    public double getFunctionValue(double x) { no usages
        //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



The screenshot shows the IntelliJ IDEA interface. The left sidebar displays a project structure for a Java application named 'lb2'. The 'src' directory contains 'functions' which holds 'FunctionPoint.java' and 'TabulatedFunction.java'. The 'Main.java' file is also visible. The right side of the screen shows the code editor with three tabs: 'Main.java', 'FunctionPoint.java', and 'TabulatedFunction.java'. The 'TabulatedFunction.java' tab is active, showing the following code:

```
public class TabulatedFunction {    no usages
    //возвращает количество точек
    public int getPointsCount() {    no usages
        return pointsCount;
    }

    // возвращает копию точки по индексу
    public FunctionPoint getPoint(int index) {    no usages
        return new FunctionPoint(points[index]);
    }

    //замена точку по индексу
    public void setPoint(int index, FunctionPoint point) {    no usages
        // проверка что x между соседними точками
        if (index > 0 && point.get_x() <= points[index - 1].get_x()) {
            return; //x меньше предыдущей точки
        }

        if (index < pointsCount - 1 && point.get_x() >= points[index + 1].get_x()) {
            return; //x больше следующей точки
        }

        points[index] = new FunctionPoint(point);
    }

    //возвращаем x точки по индексу
    public double getPointX(int index) {    no usages
        return points[index].get_x();
    }

    //изменяет x точки по индексу
    public void setPointX(int index, double x) {    no usages
        //если x правильный
        if (index > 0 && x <= points[index - 1].get_x()) {
            return; //x меньше предыдущей точки
        }

        if (index < pointsCount - 1 && x >= points[index + 1].get_x()) {
```

The screenshot shows the IntelliJ IDEA interface with the code editor open to the `TabulatedFunction.java` file. The project navigation bar at the top indicates the current file is `TabulatedFunction.java`. The code editor displays the following Java class definition:

```
public class TabulatedFunction {    no usages
    public void setPoint(int index, FunctionPoint point) {    no usages
        }
    //возвращаем x точки по индексу
    public double getPointX(int index) {    no usages
        return points[index].get_x();
    }
    //изменяет x точки по индексу
    public void setPointX(int index, double x) {    no usages
        //если x правильный
        if (index > 0 && x <= points[index - 1].get_x()) {
            return; //x меньше предыдущей точки
        }
        if (index < pointsCount - 1 && x >= points[index + 1].get_x()) {
            return; //x больше следующей точки
        }

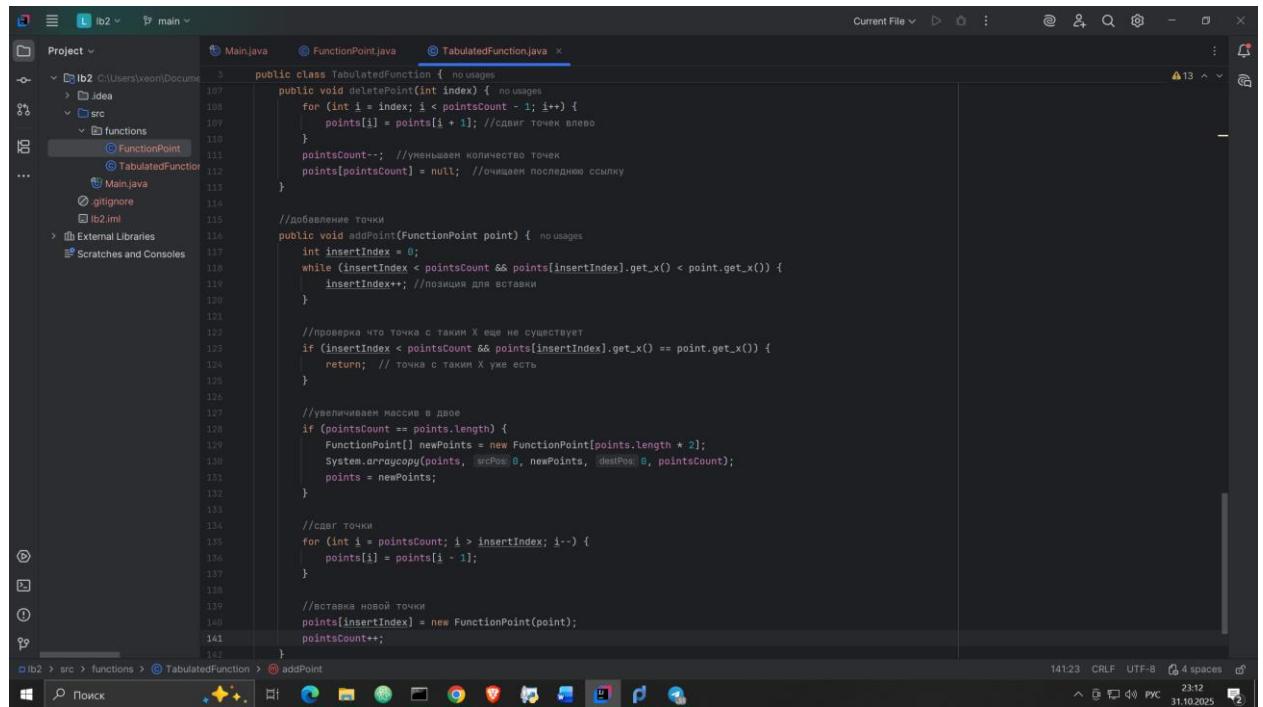
        points[index].set_x(x); // изменяем x
    }
    public double getPointY(int index) {    no usages
        return points[index].get_y(); //получает у точки по индексу
    }
    public void setPointY(int index, double y) {    no usages
        points[index].set_y(y); //изменяем у точки по индексу
    }
}
```

The code editor includes standard IntelliJ features like code folding, navigation bars, and a status bar at the bottom showing file statistics.

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.



The screenshot shows the IntelliJ IDEA interface with the code editor open to the `TabulatedFunction.java` file. The code implements a tabulated function class with methods for deleting and adding points.

```
public class TabulatedFunction {    public void deletePoint(int index) {        for (int i = index; i < pointsCount - 1; i++) {            points[i] = points[i + 1]; //сдвигает точки влево        }        pointsCount--; //уменьшаем количество точек        points[pointsCount] = null; //очищаем последнюю ссылку    }    //добавление точки    public void addPoint(FunctionPoint point) {        int insertIndex = 0;        while (insertIndex < pointsCount && points[insertIndex].get_x() < point.get_x()) {            insertIndex++; //позиция для вставки        }        if (insertIndex < pointsCount && points[insertIndex].get_x() == point.get_x()) {            return; //точка с таким X уже есть        }        if (pointsCount == points.length) {            FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];            System.arraycopy(points, srcPos, newPoints, destPos, pointsCount);            points = newPoints;        }        //сдвиг точки        for (int i = pointsCount; i > insertIndex; i--) {            points[i] = points[i - 1];        }        //вставка новой точки        points[insertIndex] = new FunctionPoint(point);        pointsCount++;    }}
```

Задание 7

Проверка работы

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure for "lb2" with files like "Main.java", "FunctionPoint.java", and "TabulatedFunction.java".
- Main.java Content:** The code implements a tabulated function. It defines a class "Main" with a static void main method. This method creates a "TabulatedFunction" object with points at (0, 0), (1, 1), (2, 2), (3, 3), and (4, 4). It then iterates through these points, printing them and their function values. It also adds a point at (2.5, 6.25) and removes the point at index 1 (1, 1). Finally, it prints the current number of points and the domain borders.
- Run Output:** The terminal window shows the execution results:
 - Test 1: Creation of functions.
 - Test 2: Calculation of values. It shows f(-1.0) is undefined, and values for f(0.0), f(1.5), f(4.0), and f(5.0).
 - Test 3: Addition and deletion of points. It shows the addition of a point at (2.5, 6.25), the removal of the point at index 1, and the final count of points and domain borders.
- Bottom Status Bar:** Shows the time as 18:35, file format as LF, encoding as UTF-8, and a space configuration of 4 spaces.

Доработанный код 1

Стал использовать эпсилон при сравнении

The image shows two side-by-side screenshots of a Java code editor, likely PyCharm, displaying the same file: TabulatedFunction.java. Both screenshots show the same code with minor differences in line numbers and some comments.

```
public class TabulatedFunction { // 3 usages & valera228*
    public FunctionPoint getPoint(int index) { return new FunctionPoint(points[index]); }

    //замена точки по индексу
    public void setPoint(int index, FunctionPoint point) { // no usages & valera228*
        // проверка что x между соседними точками
        if (index > 0 && point.get_x() <= points[index - 1].get_x() + 1e-10) {
            return; //x меньше предыдущей точки
        }
        if (index < pointsCount - 1 && point.get_x() >= points[index + 1].get_x() - 1e-10) {
            return; //x больше следующей точки
        }

        points[index] = new FunctionPoint(point);
    }

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

    //изменяет x точки по индексу
    public void setPointX(int index, double x) { // no usages & valera228*
        //если x правильный
        if (index > 0 && x <= points[index - 1].get_x() + 1e-10) {
            return; //x меньше предыдущей точки
        }
        if (index < pointsCount - 1 && x >= points[index + 1].get_x() - 1e-10) {
            return; //x больше следующей точки
        }

        points[index].set_x(x); // изменяю x
    }

    public double getY(int index) { return points[index].get_y(); } //получает у точки по индексу
}

Lab-2-2025 > functions > TabulatedFunction > setPointX
```

90.81 CRLF UTF-8 4 spaces


```
public void setPointY(int index, double y) { points[index].set_y(y); //изменяю у точки по индексу }

//удаление точки по индексу
public void deletePoint(int index) { 1 usage & valera228
    for (int i = index; i < pointsCount - 1; i++) {
        points[i] = points[i + 1]; //сдвиг точек влево
    }
    pointsCount--; //уменьшаем количество точек
    points[pointsCount] = null; //очищаем последнюю ссылку
}

//добавление точки
public void addPoint(FunctionPoint point) { 2 usages & valera228*
    int insertIndex = 0;
    while (insertIndex < pointsCount && Math.abs(points[insertIndex].get_x() - point.get_x()) < 1e-10) {
        insertIndex++; //позиция для вставки
    }

    //проверка что точка с таким X еще не существует
    if (insertIndex < pointsCount && Math.abs(points[insertIndex].get_x() - point.get_x()) < 1e-10) {
        return; //точка с таким X уже есть
    }

    //увеличиваем массив вдвое
    if (pointsCount == points.length) {
        FunctionPoint[] newPoints = new FunctionPoint[points.length * 2];
        System.arraycopy(points, srcPos: 0, newPoints, destPos: 0, pointsCount);
        points = newPoints;
    }

    //сдвиг точки
    for (int i = pointsCount; i > insertIndex; i--) {
        points[i] = points[i - 1];
    }

    //вставка новой точки
}
```

83.1 CRLF UTF-8 4 spaces

Изменил проверку работы

The screenshot shows an IDE window for 'Lab-2-2025' with the following details:

- Project Explorer:** Shows files: 'Main.java' and 'TabulatedFunction.java'.
- Code Editor:** Displays Java code for creating and manipulating functions. The code includes:
 - Creation of a function point and tabulated function.
 - Printing points after creation.
 - Calculation of function values for specific x-coordinates.
 - Addition and deletion of points from the function.
- Output Window:** Shows the execution results for three tests:
 - TEST 1:** Creation of a function. It prints points (0.0, 0.0), (1.0, 1.0), (2.0, 4.0), (3.0, 9.0), and (4.0, 16.0).
 - TEST 2:** Calculation of function values. It prints f(-1.0) as undefined, f(0.0) as 0.0, f(1.5) as 2.5, f(4.0) as 16.0, and f(5.0) as undefined.
 - TEST 3:** Addition and deletion of points. It adds a point at (2.5, 6.25) and then prints points again, showing they remain (0.0, 0.0), (1.0, 1.0), (2.0, 4.0), (2.5, 6.25), (3.0, 9.0), and (4.0, 16.0). It also prints the count of points as 6.
- Status Bar:** Shows the file path 'Lab-2-2025 > Main.java', line 1076, character 1076, encoding 'UTF-8', and a note '4 spaces'.

The screenshot shows the PyCharm IDE interface with the following details:

- Project Bar:** Shows "Lab-2-2025" as the current project.
- File Bar:** Shows "main" as the current file.
- Code Editor:** Displays Java code for a class named Main. The code performs various operations on a list of points (X and Y coordinates) and prints results to the console.
- Run Tab:** Shows the output of the code execution. It includes:
 - Print statements: "Количество точек: 5", "Точка 4: (4.0, 16.0)", "Количество точек: 5", "Изменение 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).
 - A note about adding a point at (2.0, 100.0): "Пытаемся добавить точку (2.0, 100.0) - должна быть отклонена".
 - A note about attempting to add a duplicate point at (2.0, 100.0): "Попытка 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)".
 - A test case for changing X coordinates: "*** ТЕСТ 4: ИЗМЕНЕНИЕ Х ТОЧЕК ***", "Изменение X точки с индексом 1 с 2.0 на 1.8", "Точки после изменения X точек 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)".
 - A note about changing X to -1.0: "Пытаемся изменить X точки 0 на -1.0 (должно быть отклонено)".
- Bottom Status Bar:** Shows "107 LF", "UTF-8", "4 spaces", "17.16", and "РУС".

The screenshot shows a Java development environment with a code editor and a terminal window. The code editor contains a Main.java file with several print statements and logic related to modifying points. The terminal window displays the output of the program's execution, showing test cases and their results.

```
public class Main {  
    public static void main(String[] args) {  
        // Пытаемся изменить X точки 0 на некорректное значение (должно быть отклонено)  
        System.out.println("\nПытаемся изменить X точки 0 на -1.0 (должно быть отклонено)");  
        func.setPoint(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) + ")");  
        }  
  
        // Замена точки целиком  
        System.out.println("\n*** ТЕСТ 5: ЗАМЕНА ТОЧЕК ***");  
  
        System.out.println("Заменен точку с индексом 2 на (2.2, 15.0)");  
        func.setPoint(index: 2, new FunctionPoint(x: 2.2, y: 15.0));  
        System.out.println("Точки после замены точки 2:");  
        for (int i = 0; i < func.getPointsCount(); i++) {  
            System.out.println(" Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");  
        }  
  
        // Попытка заменить точку на некорректную (должно быть отклонено)  
        System.out.println("\nПытаемся заменить точку 1 на (3.5, 5.0) - X выходит за границы (должно быть отклонено)");  
        func.setPoint(index: 1, new FunctionPoint(x: 3.5, y: 5.0));  
        System.out.println("Точки после НЕУДАЧНОЙ попытки замены:");  
        for (int i = 0; i < func.getPointsCount(); i++) {  
            System.out.println(" Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");  
        }  
  
        System.out.println("\n*** ТЕСТ 6: ГРАНИЦЫ И ФИНАЛЬНОЕ СОСТОЯНИЕ ***");  
        System.out.println("Левая граница: " + func.getLeftDomainBorder());  
        System.out.println("Правая граница: " + func.getRightDomainBorder());  
        System.out.println("Финальное количество точек: " + func.getPointsCount());  
        System.out.println("Финальные точки функции:");  
        for (int i = 0; i < func.getPointsCount(); i++) {  
            System.out.println(" Точка " + i + ": (" + func.getPointX(i) + ", " + func.getPointY(i) + ")");  
        }  
    }  
}
```

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

The screenshot shows a Java development environment with two tabs open: `Main.java` and `TabulatedFunction.java`. The `TabulatedFunction.java` tab is active, displaying the following code:

```
public class TabulatedFunction {  
    public void setPoint(int index, FunctionPoint point) {  
        points[index] = new FunctionPoint(point);  
    }  
  
    // возвращаем x точки по индексу  
    public double getPoint(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);  
    }  
}
```

The right side of the interface displays the output of several tests:

- Test 4: Изменение X точек с индексом 1 с 2.0 на 1.8. Result: Точки 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). Note: Пытаюсь изменить X точки 0 на -1.0 (должно быть отклонено).
- Test 5: Замена точки 2 на (2.2, 15.0). Result: Точки 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). Note: Заменена точку с индексом 2 на (2.2, 15.0). Точки 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).
- Test 6: Границы и финальное состояние. Result: Точки 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). Note: Пытаюсь заменить точку 1 на (3.5, 5.0) - X выходит за границы. Точки 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).

At the bottom, the status bar shows: 88:10 CRLF UTF-8 4 spaces 18:08 02.11.2025.