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 №3

Дисциплина: «ООП»

Выполнил: Илларионов Е.А.

Группа: 6201-120303D

Самара, 2025

Задание 1

Изучил классы исключений Java:

java.lang.Exception

java.lang.IndexOutOfBoundsException

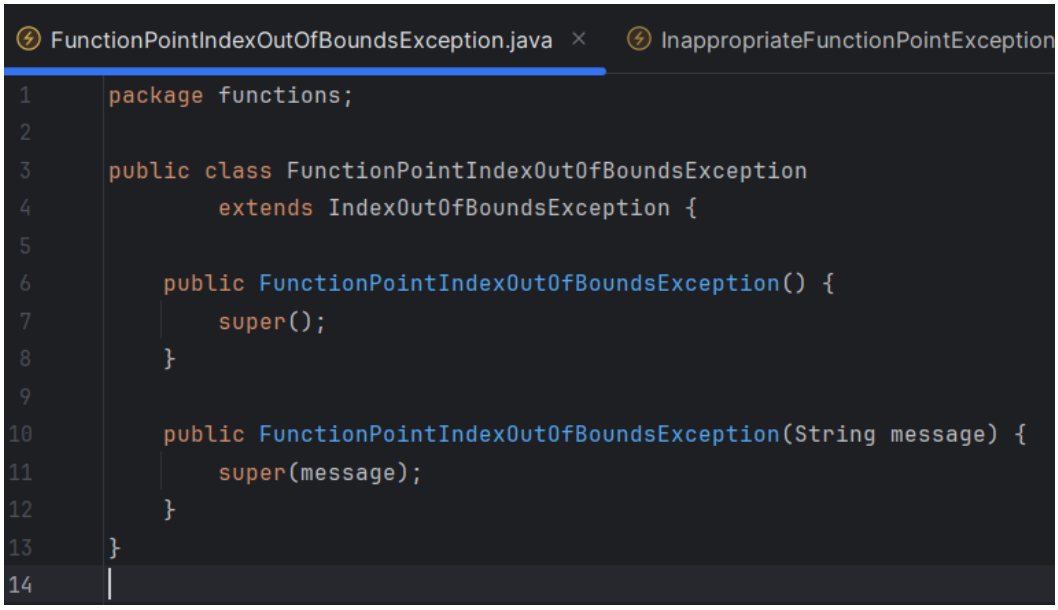
java.lang.ArrayIndexOutOfBoundsException

java.lang.IllegalArgumentException

java.lang.IllegalStateException

Задание 2

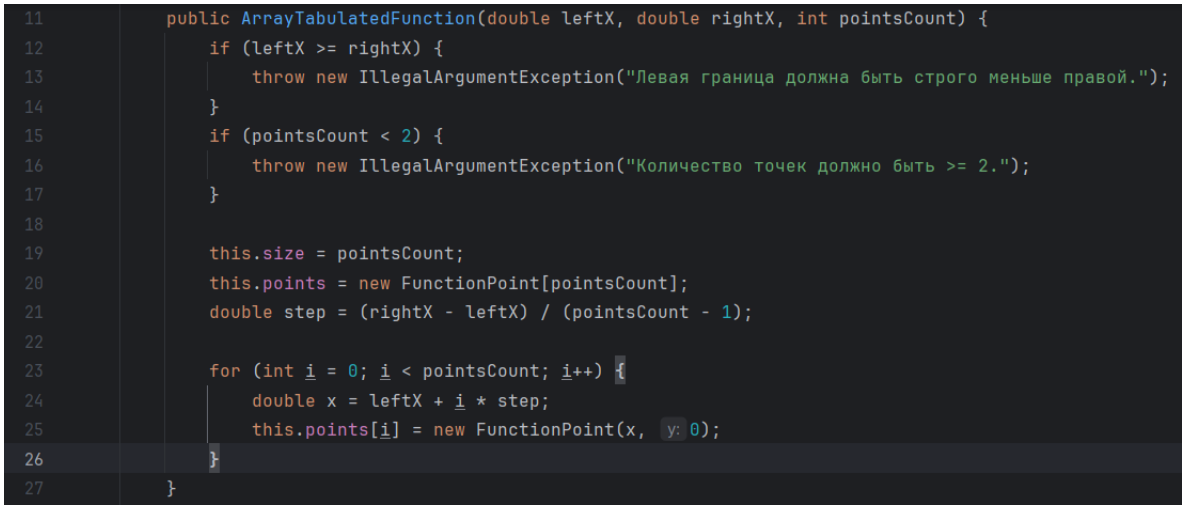
В пакете functions создал 2 класса исключений FunctionPointIndexOutOfBoundsException



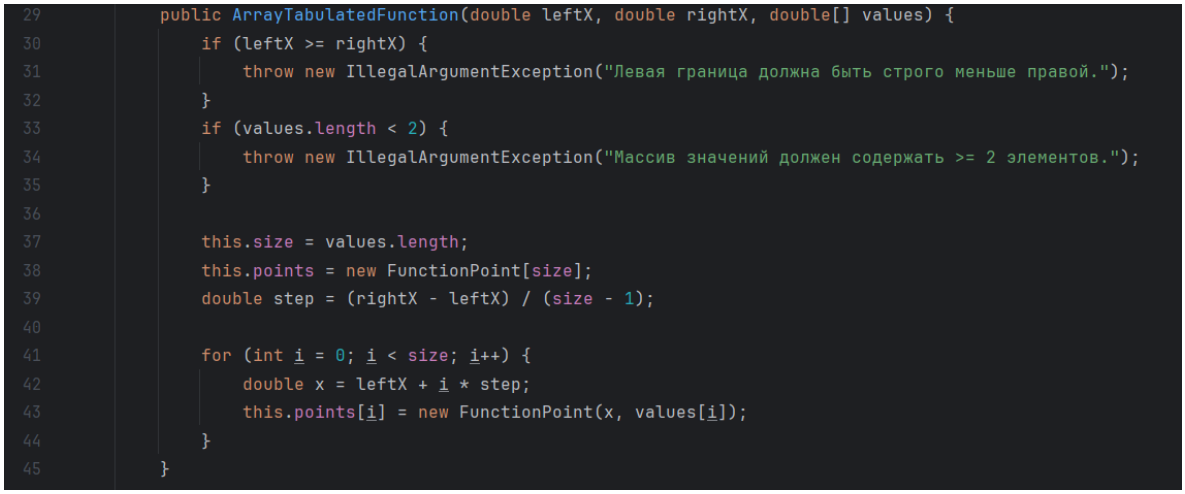
InappropiateFunctionPointExpection

Задание 3

В разработанный ранее класс TabulatedFunction внес изменения, обеспечивающие выбрасывание исключений методом класса



```
11      public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
12          if (leftX >= rightX) {
13              throw new IllegalArgumentException("Левая граница должна быть строго меньше правой.");
14          }
15          if (pointsCount < 2) {
16              throw new IllegalArgumentException("Количество точек должно быть >= 2.");
17          }
18
19          this.size = pointsCount;
20          this.points = new FunctionPoint[pointsCount];
21          double step = (rightX - leftX) / (pointsCount - 1);
22
23          for (int i = 0; i < pointsCount; i++) {
24              double x = leftX + i * step;
25              this.points[i] = new FunctionPoint(x,  y: 0);
26          }
27      }
```



```
29      public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {
30          if (leftX >= rightX) {
31              throw new IllegalArgumentException("Левая граница должна быть строго меньше правой.");
32          }
33          if (values.length < 2) {
34              throw new IllegalArgumentException("Массив значений должен содержать >= 2 элементов.");
35          }
36
37          this.size = values.length;
38          this.points = new FunctionPoint[size];
39          double step = (rightX - leftX) / (size - 1);
40
41          for (int i = 0; i < size; i++) {
42              double x = leftX + i * step;
43              this.points[i] = new FunctionPoint(x, values[i]);
44          }
45      }
```

Методы getPoint(), getPointX(), getPointY(), setPointY() выбрасывают исключение FunctionPointIndexOutOfBoundsException, если переданный в метод номер выходит за границы набора точек.

Методы setPoint(), setPointX(), выбрасывают исключение FunctionPointIndexOutOfBoundsException, если переданный в метод номер выходит за границы набора точек, и исключение InappropiateFunctionPointExpection в том случае, если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции.

Задание 4

В пакете functions создал класс LinkedListTabulatedFunction.

Класс FunctionNode реализован как приватный вложенный класс, обеспечивая инкапсуляцию структуры списка.

```
3   public class LinkedListTabulatedFunction implements TabulatedFunction {
4
5       // Вложенный приватный класс FunctionNode
6       private static class FunctionNode {
7           public FunctionPoint point;
8           public FunctionNode prev;
9           public FunctionNode next;
10
11          public FunctionNode(FunctionPoint point) {
12              this.point = point;
13              this.prev = this.next = this;
14          }
15      }
```

Реализован метод getNodeByIndex(int index), который использует кэширование последнего узла и выбирает направление поиска (от начала или от конца списка) для ускорения операций.

```
74      private FunctionNode getNodeByIndex(int index) {
75          FunctionNode current;
76
77          if (index == lastAccessedIndex) {
78              return lastAccessedNode;
79          }
80
81          if (index < size / 2) {
82              current = head.next;
83              for (int i = 0; i < index; i++) {
84                  current = current.next;
85              }
86          } else {
87              current = head.prev;
88              for (int i = size - 1; i > index; i--) {
89                  current = current.prev;
90              }
91          }
```

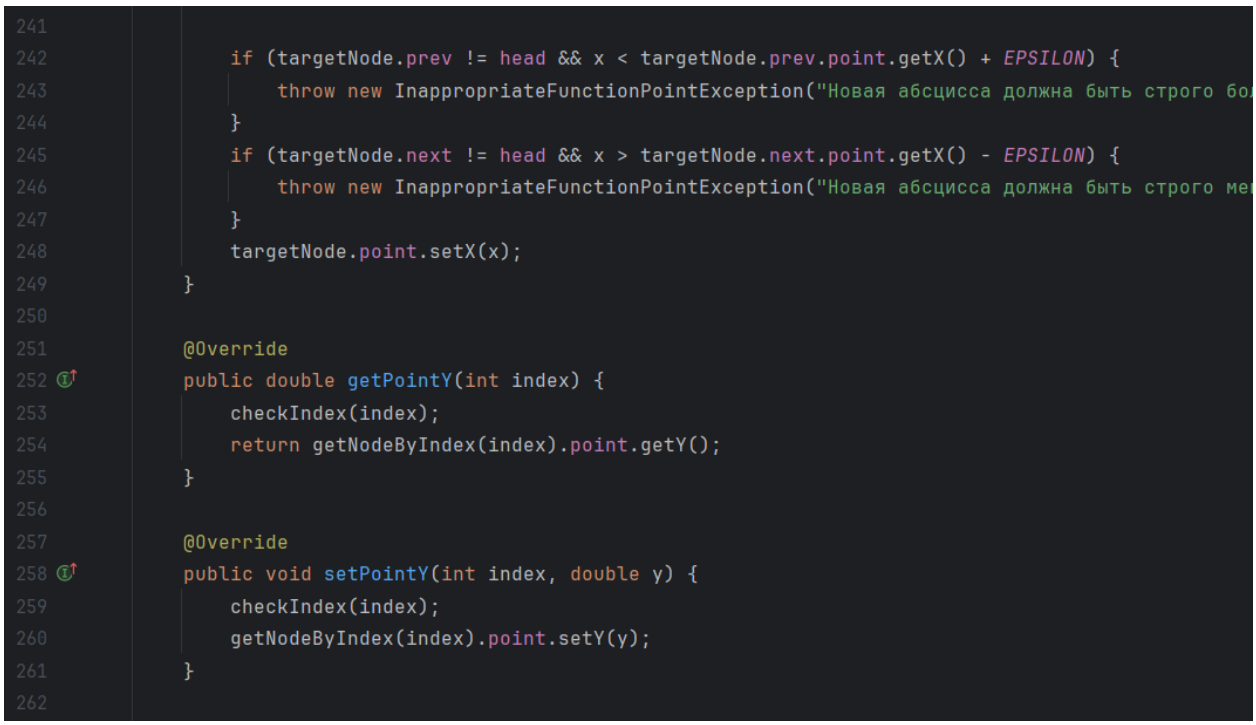
Задание 5

В классе LinkedListTabulatedFunction реализованы публичные конструкторы и методы, соответствующие сигнатурам ArrayTabulatedFunction

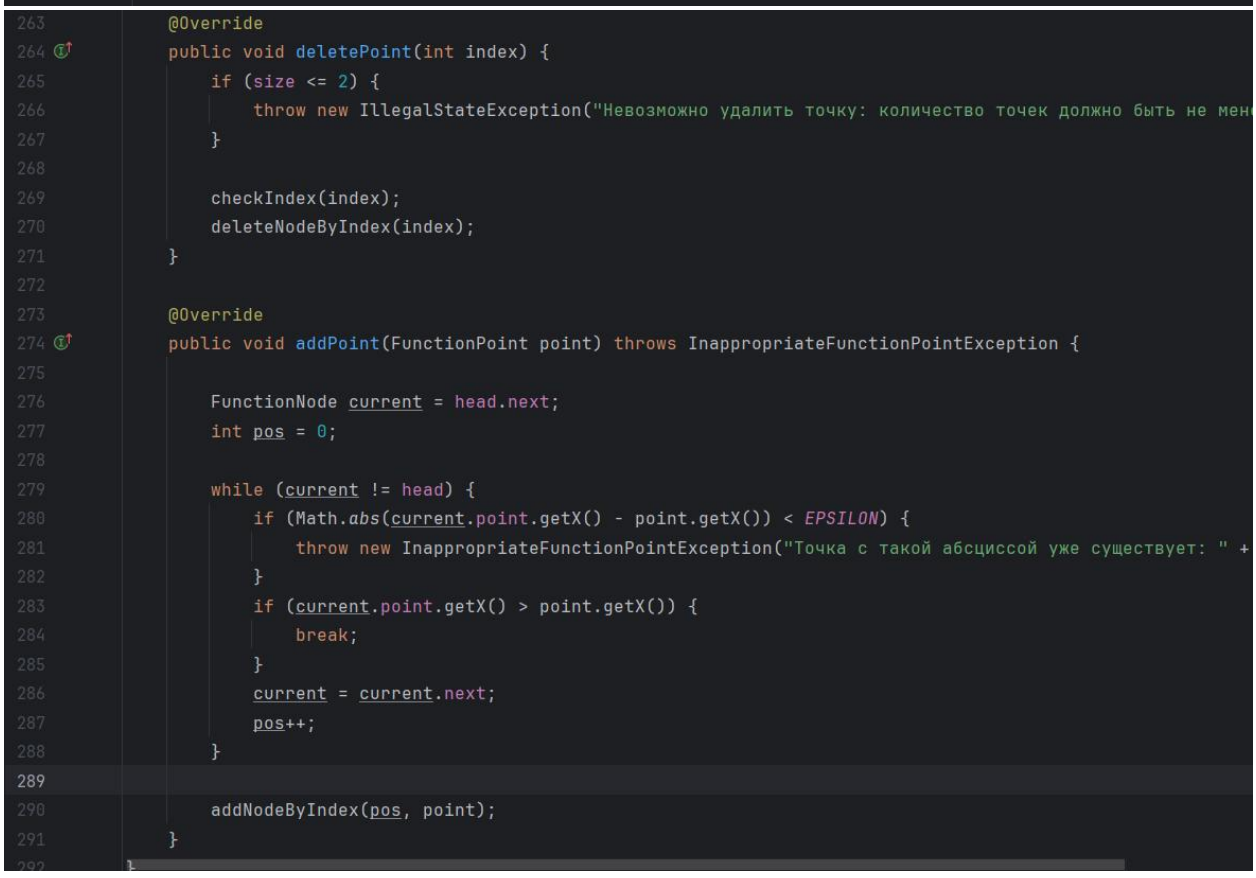
```java
public LinkedListTabulatedFunction(double leftX, double rightX, double[] values) {
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть строго меньше правой.");
    }
    if (values.length < 2) {
        throw new IllegalArgumentException("Массив значений должен содержать >= 2 элементов.");
    }

    initializeList();

    double step = (rightX - leftX) / (values.length - 1);

    for (int i = 0; i < values.length; i++) {
        double x = leftX + i * step;
        addNodeToTail(new FunctionPoint(x, values[i]));
    }
}
```

```java
public double getRightDomainBorder() {
    if (size == 0) return Double.NaN;
    return head.prev.point.getX();
}

@Override
public double getFunctionValue(double x) {
    if (x < getLeftDomainBorder() || x > getRightDomainBorder()) return Double.NaN;

    FunctionNode current = head.next;
    if (lastAccessedIndex != -1) {
        current = lastAccessedNode;
    }

    while (current.next != head) {
        FunctionNode nextNode = current.next;

        if (x >= current.point.getX() && x <= nextNode.point.getX() + EPSILON) {
            lastAccessedNode = current;
            lastAccessedIndex = -1;
            // Линейная интерполяция
            double x0 = current.point.getX();
            double y0 = current.point.getY();
            double x1 = nextNode.point.getX();
            double y1 = nextNode.point.getY();

            if (Math.abs(x1 - x0) < EPSILON) return y0;

            return y0 + (y1 - y0) * (x - x0) / (x1 - x0);
        }
```

```
            current = current.next;
        }
        return Double.NaN;
    }

    @Override
    public int getPointsCount() {
        return size;
    }

    @Override
    public FunctionPoint getPoint(int index) {
        checkIndex(index);
        return new FunctionPoint(getNodeByIndex(index).point);
    }

    @Override
    public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException {
        checkIndex(index);

        FunctionNode targetNode = getNodeByIndex(index);
        double newX = point.getX();

        if (targetNode.prev != head && newX < targetNode.prev.point.getX() + EPSILON) {
            throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго больше предыду
        }
        if (targetNode.next != head && newX > targetNode.next.point.getX() - EPSILON) {
            throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго меньше следующ
        }

        targetNode.point.setX(newX);
        targetNode.point.setY(point.getY());
    }

    @Override
    public double getPointX(int index) {
        checkIndex(index);
        return getNodeByIndex(index).point.getX();
    }

    @Override
    public void setPointX(int index, double x) throws InappropriateFunctionPointException {
        checkIndex(index);

        FunctionNode targetNode = getNodeByIndex(index);
```



```
241
242         if (targetNode.prev != head && x < targetNode.prev.point.getX() + EPSILON) {
243             throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго бо
244         }
245         if (targetNode.next != head && x > targetNode.next.point.getX() - EPSILON) {
246             throw new InappropriateFunctionPointException("Новая абсцисса должна быть строго ме
247         }
248         targetNode.point.setX(x);
249     }
250
251     @Override
252     public double getPointY(int index) {
253         checkIndex(index);
254         return getNodeByIndex(index).point.getY();
255     }
256
257     @Override
258     public void setPointY(int index, double y) {
259         checkIndex(index);
260         getNodeByIndex(index).point.setY(y);
261     }
262
```



```
263     @Override
264     public void deletePoint(int index) {
265         if (size <= 2) {
266             throw new IllegalStateException("Невозможно удалить точку: количество точек должно быть не мен
267         }
268
269         checkIndex(index);
270         deleteNodeByIndex(index);
271     }
272
273     @Override
274     public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {
275
276         FunctionNode current = head.next;
277         int pos = 0;
278
279         while (current != head) {
280             if (Math.abs(current.point.getX() - point.getX()) < EPSILON) {
281                 throw new InappropriateFunctionPointException("Точка с такой абсциссой уже существует: " +
282             }
283             if (current.point.getX() > point.getX()) {
284                 break;
285             }
286             current = current.next;
287             pos++;
288         }
289
290         addNodeByIndex(pos, point);
291     }
```

Задание 6

Класс TabulatedFunction переименован в ArrayTabulatedFunction.

Создан интерфейс TabulatedFunction.java, содержащий объявления всех общих методов.

Оба класса функций (ArrayTabulatedFunction и LinkedListTabulatedFunction) реализовали созданный интерфейс.

Задание 7

Класс main

```java
import functions.FunctionPoint;
import functions.TabulatedFunction;
import functions.ArrayTabulatedFunction;
import functions.LinkedListTabulatedFunction;
import functions.FunctionPointIndexOutOfBoundsException;
import functions.InappropriateFunctionPointException;

public class Main {

    // Вывод всех точек функции в консоль
    private static void printPoints(TabulatedFunction func) {
        System.out.println("Точки функции:");
        for (int i = 0; i < func.getPointsCount(); i++) {
            System.out.printf(
                    "  [%d] x = %.3f, y = %.3f%n",
                    i,
                    func.getPointX(i),
                    func.getPointY(i)
            );
        }
    }

    // Проверка исключений
    public static void testExceptions(TabulatedFunction func, String name) {
        System.out.println("\n--- Тестирование исключений для " + name + " ---");

        // 1. IllegalArgumentException — конструктор
        try {
            new ArrayTabulatedFunction( leftX: 10,  rightX: 0,  pointsCount: 5);
        } catch (IllegalArgumentException e) {
            System.out.println("Конструктор (OK): " + e.getMessage());
        }

        // 2. IllegalStateException — deletePoint при size <= 2
        try {
            double[] values = {1, 2};
            TabulatedFunction shortFunc = new ArrayTabulatedFunction( leftX: 0,  rightX: 1, values);
            shortFunc.deletePoint( index: 0);
        } catch (IllegalStateException e) {
            System.out.println("deletePoint (OK): " + e.getMessage());
        }

        // 3. FunctionPointIndexOutOfBoundsException
        try {
            func.getPoint( index: 100);
        } catch (FunctionPointIndexOutOfBoundsException e) {
            System.out.println("getPoint (OK): " + e.getMessage());
        }

        // 4. InappropriateFunctionPointException — addPoint
        try {
            func.addPoint(new FunctionPoint(func.getPointX( index: 1),  y: 50));
        } catch (InappropriateFunctionPointException e) {
            System.out.println("addPoint (OK): " + e.getMessage());
        }

    }
```

```
        // 5. InappropriateFunctionPointException — setPointX
        try {
            func.setPointX( index: 1,  x: func.getPointX( index: 0) - 1);
        } catch (InappropriateFunctionPointException e) {
            System.out.println("setPointX (OK): " + e.getMessage());
        }

        System.out.println("Тестирование исключений завершено.");
    }

    //Проверка добавления, удаления и замены нулевой точки
    public static void testZeroIndexOperations(TabulatedFunction func, String name) {
        System.out.println("\n--- Проверка операций с нулевой точкой: " + name + " ---");

        // Исходное состояние
        System.out.println("Исходные точки:");
        printPoints(func);

        // 1. Замена нулевой точки
        try {
            FunctionPoint oldPoint = func.getPoint( index: 0);
            FunctionPoint newPoint =
                    new FunctionPoint(oldPoint.getX(),  y: oldPoint.getY() + 100);

            func.setPoint( index: 0, newPoint);

            System.out.println("\nПосле setPoint(0):");
            printPoints(func);
        } catch (Exception e) {
            System.out.println("setPoint(0) ERROR: " + e.getMessage());
        }

        // 2. Удаление нулевой точки
        try {
            System.out.println("\nПеред deletePoint(0):");
            printPoints(func);

            func.deletePoint( index: 0);

            System.out.println("После deletePoint(0):");
            printPoints(func);
        } catch (Exception e) {
            System.out.println("deletePoint(0) ERROR: " + e.getMessage());
        }

        // 3. Добавление новой первой точки
        try {
            double newX = func.getLeftDomainBorder() - 1;
            FunctionPoint newPoint = new FunctionPoint(newX,  y: 999);

            System.out.println("\nПеред addPoint(new first):");
            printPoints(func);

            func.addPoint(newPoint);

            System.out.println("После addPoint(new first):");
            printPoints(func);
```

```
        } catch (Exception e) {
            System.out.println("addPoint(new first) ERROR: " + e.getMessage());
        }
    }

    public static void main(String[] args) {

        double[] values = {10, 20, 30, 40, 50};

        TabulatedFunction arrayFunc =
                new ArrayTabulatedFunction( leftX: 0,  rightX: 4, values);
        TabulatedFunction linkedFunc =
                new LinkedListTabulatedFunction( leftX: 0,  rightX: 4, values);

        System.out.println("--- Проверка работы ArrayTabulatedFunction ---");
        System.out.println("f(1.5) = " + arrayFunc.getFunctionValue( x: 1.5));

        System.out.println("\n--- Проверка работы LinkedListTabulatedFunction ---");
        System.out.println("f(2.5) = " + linkedFunc.getFunctionValue( x: 2.5));

        // Проверка исключений
        testExceptions(arrayFunc,  name: "ArrayTabulatedFunction");
        testExceptions(linkedFunc,  name: "LinkedListTabulatedFunction");

        // Проверка операций с нулевой точкой
        testZeroIndexOperations(
                new ArrayTabulatedFunction( leftX: 0,  rightX: 4, new double[]{10, 20, 30, 40}),
                 name: "ArrayTabulatedFunction"
        );

        testZeroIndexOperations(
                new LinkedListTabulatedFunction( leftX: 0,  rightX: 4, new double[]{10, 20, 30, 40}),
                 name: "LinkedListTabulatedFunction"
        );
    }
}
```

Результат

```
--- Проверка работы ArrayTabulatedFunction ---
f(1.5) = 25.0

--- Проверка работы LinkedListTabulatedFunction ---
f(2.5) = 35.0

--- Тестирование исключений для ArrayTabulatedFunction ---
Конструктор (ОК): Левая граница должна быть строго меньше правой.
deletePoint (ОК): Невозможно удалить точку: количество точек должно быть не менее трех.
getPoint (ОК): Индекс 100 выходит за границы [0, 4]
addPoint (ОК): Точка с такой абсциссой уже существует: 1.0
setPointX (ОК): Новая абсцисса должна быть строго больше предыдущей точки.
Тестирование исключений завершено.

--- Тестирование исключений для LinkedListTabulatedFunction ---
Конструктор (ОК): Левая граница должна быть строго меньше правой.
deletePoint (ОК): Невозможно удалить точку: количество точек должно быть не менее трех.
getPoint (ОК): Индекс 100 выходит за границы [0, 4]
addPoint (ОК): Точка с такой абсциссой уже существует: 1.0
setPointX (ОК): Новая абсцисса должна быть строго больше предыдущей точки.
Тестирование исключений завершено.

--- Проверка операций с нулевой точкой: ArrayTabulatedFunction ---
Исходные точки:
Точки функции:
  [0] x = 0,000, y = 10,000
  [1] x = 1,333, y = 20,000
```

```
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000

После setPoint(0):
Точки функции:
  [0] x = 0,000, y = 110,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000

Перед deletePoint(0):
Точки функции:
  [0] x = 0,000, y = 110,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000
После deletePoint(0):
Точки функции:
  [0] x = 1,333, y = 20,000
  [1] x = 2,667, y = 30,000
  [2] x = 4,000, y = 40,000

Перед addPoint(new first):
Точки функции:
  [0] x = 1,333, y = 20,000
  [1] x = 2,667, y = 30,000
  [2] x = 4,000, y = 40,000
После addPoint(new first):
```

```
Точки функции:
  [0] x = 0,333, y = 999,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000

--- Проверка операций с нулевой точкой: LinkedListTabulatedFunction ---
Исходные точки:
Точки функции:
  [0] x = 0,000, y = 10,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000

После setPoint(0):
Точки функции:
  [0] x = 0,000, y = 110,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000

Перед deletePoint(0):
Точки функции:
  [0] x = 0,000, y = 110,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000
```

```
После deletePoint(0):
Точки функции:
  [0] x = 1,333, y = 20,000
  [1] x = 2,667, y = 30,000
  [2] x = 4,000, y = 40,000

Перед addPoint(new first):
Точки функции:
  [0] x = 1,333, y = 20,000
  [1] x = 2,667, y = 30,000
  [2] x = 4,000, y = 40,000
После addPoint(new first):
Точки функции:
  [0] x = 0,333, y = 999,000
  [1] x = 1,333, y = 20,000
  [2] x = 2,667, y = 30,000
  [3] x = 4,000, y = 40,000

Process finished with exit code 0
```