

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

Институт «Информатики и кибернетики»

Специальность «Фотоника и оптоинформатика 6201-120303D»

Отчет по лабораторной работе № 3

Выполнил: студент Султанова А.М.,

группа 6201-120303D

Проверил: преподаватель Борисов Д.С.

Самара 2025

## Задание 2

В пакете functions создаем два класса исключений:

- FunctionPointIndexOutOfBoundsException – исключение выхода за границы набора точек при обращении к ним по номеру, наследует от класса IndexOutOfBoundsException;
- InappropriateFunctionPointException – исключение, выбрасываемое при попытке добавления или изменения точки функции несоответствующим образом, наследует от класса Exception.

```
Project : lab2 C:\Users\admin\IdeaProjects\lab2
Main.java
FunctionPointIndexOutOfBoundsException.java
InappropriateFunctionPointException.java
Function

package functions;

public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException {
    public FunctionPointIndexOutOfBoundsException() { super("Индекс точки функции вне допустимого диапазона."); }
    public FunctionPointIndexOutOfBoundsException(String message) { super(message); }
}

Project : lab2 C:\Users\admin\IdeaProjects\lab2
Main.java
FunctionPointIndexOutOfBoundsException.java
InappropriateFunctionPointException.java
Function

package functions;

public class InappropriateFunctionPointException extends Exception {
    public InappropriateFunctionPointException() { super("Недопустимое добавление или изменение точки функции."); }
    public InappropriateFunctionPointException(String message) { super(message); }
}
```

## Задание 3

В каждом конструкторе TabulatedFunction (их два — с count и с values[]) добавим проверку на корректность аргументов.

```

public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) { no usages & SultanovaArina*
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть меньше правой");
    }
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Количество точек должно быть не меньше двух");
    }

    this.points = new FunctionPoint[pointsCount];
    this.size = pointsCount;
    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + i * step;
        points[i] = new FunctionPoint(x, y: 0.0);
    }
}

public ArrayTabulatedFunction(double leftX, double rightX, double[] values) { no usages & SultanovaArina*
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть меньше правой");
    }
    if (values.length < 2) {
        throw new IllegalArgumentException("Количество точек должно быть не меньше двух");
    }
    int n = values.length;
    this.points = new FunctionPoint[n];
    this.size = n;
    double step = (rightX - leftX) / (n - 1);
    for (int i = 0; i < n; i++) {
        double x = leftX + i * step;
        points[i] = new FunctionPoint(x, values[i]);
    }
}
private void checkIndex(int index) { 7 usages new *
}

```

Теперь нужно выбрасывать `FunctionPointIndexOutOfBoundsException`, если `index` выходит за границы массива.

Создаем вспомогательный метод:

```

}
private void checkIndex(int index) { 7 usages new *
    if (index < 0 || index >= size) {
        throw new FunctionPointIndexOutOfBoundsException("Индекс " + index + " вне диапазона точек");
    }
}

```

Теперь в методы, где используется индекс (`getPoint`, `setPoint`, `getPointX`, `setPointX`, `getPointY`, `setPointY`, `deletePoint`) — добавим вызов этого метода в начале.

В `setPoint()` и `setPointX()` добавим проверку, чтобы новая `x` находилась между соседними.

В `addPoint()` добавим проверку, что точка с таким же `x` уже не существует.

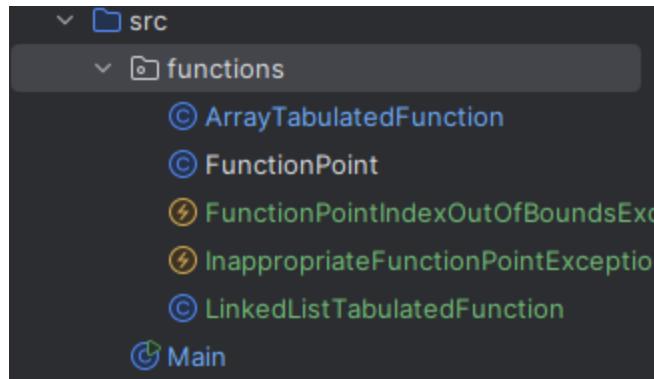
```

75 ⚡ public int getPointsCount() { 2 usages & SultanovaArina
76     return size;
77 }
78
79 ⚡ public FunctionPoint getPoint(int index) { 3 usages & SultanovaArina
80     checkIndex(index);
81     return new FunctionPoint(points[index]);
82 }
83
84 ⚡ @ public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException { 1 usage & SultanovaArina *
85     checkIndex(index);
86     double x = point.getX();
87
88
89     if ((index > 0 && (x < points[index - 1].getX() || Math.abs(x - points[index - 1].getX()) < EPS)) ||
90         (index < size - 1 && (x > points[index + 1].getX() || Math.abs(x - points[index + 1].getX()) < EPS))) {
91         throw new InappropriateFunctionPointException("Неверная координата X для данной позиции");
92     }
93     points[index] = new FunctionPoint(point);
94 }
95 ⚡ public double getPointX(int index) { 6 usages & SultanovaArina
96     checkIndex(index);
97     return points[index].getX();
98 }
99
100 ⚡ public void setPointX(int index, double x) throws InappropriateFunctionPointException { 1 usage & SultanovaArina *
101     checkIndex(index);
102
103     if ((index > 0 && (x < points[index - 1].getX() || Math.abs(x - points[index - 1].getX()) < EPS)) ||
104         (index < size - 1 && (x > points[index + 1].getX() || Math.abs(x - points[index + 1].getX()) < EPS))) {
105         throw new InappropriateFunctionPointException("Неверное значение X – нарушается порядок точек");
106     }
107
108     points[index].setX(x);
109 }
110 ⚡ public double getPointY(int index) { 2 usages & SultanovaArina
111     checkIndex(index);
112     return points[index].getY();
113 }
114 ⚡ public void setPointY(int index, double y) { 2 usages & SultanovaArina
115     checkIndex(index);
116     points[index].setY(y);
117 }
118
119 ⚡ public void deletePoint(int index) { 2 usages & SultanovaArina
120     checkIndex(index);
121     if (size < 3) {
122         throw new IllegalStateException("Нельзя удалить точку – останется меньше трёх");
123     }
124     for (int i = index; i < size - 1; i++) {
125         points[i] = points[i + 1];
126     }
127     size--;

```

## Задание 4

В пакете functions создаём новый файл:



Объявим класс и сделаем структуру головы списка

```
package functions;

public class LinkedListTabulatedFunction implements TabulatedFunction {
    private FunctionNode head; 21 usages
    private int size; 20 usages
    private static final double EPS = Math.ulp(1.0); 10 usages
```

Опишем внутренний класс FunctionNode, этот класс должен хранить: объект точки (FunctionPoint), ссылки на предыдущий и следующий элемент списка.

Он должен быть вложенным (private static class), чтобы не нарушать инкапсуляцию (внешний код не должен знать, что у нас внутри список).

```
private static class FunctionNode { 17 usages new *
    public FunctionPoint point; 1 usage
    public FunctionNode prev; 12 usages
    public FunctionNode next; 10 usages

    public FunctionNode(FunctionPoint point) { 3 usages new *
        this.point = point;
    }
}
```

Создаем голову списка: В конструкторе класса создаётся “пустая” голова — она не содержит данных, а только служит для связывания элементов. Она всегда ссылается сама на себя, если список пуст.

```
public LinkedListTabulatedFunction() { no usages new *
    head = new FunctionNode( point: null );
    head.next = head;
    head.prev = head;
    size = 0;
}
```

Метод getNodeByIndex(int index): Этот метод возвращает нужный узел списка по индексу. Здесь нужно сделать оптимизацию:

- если  $index < size / 2$ , идти с начала,
- иначе — с конца.

```

private FunctionNode getNodeByIndex(int index) { 2 usages new *
    if (index < 0 || index >= size) {
        throw new FunctionPointIndexOutOfBoundsException("Неверный индекс: " + index);
    }

    FunctionNode node;
    if (index < size / 2) {
        node = head.next;
        for (int i = 0; i < index; i++) {
            node = node.next;
        }
    } else {
        node = head.prev;
        for (int i = size - 1; i > index; i--) {
            node = node.prev;
        }
    }
    return node;
}

```

Метод addNodeToTail(): Добавляет новый элемент в конец списка и возвращает ссылку на добавленный узел.

```

private FunctionNode addNodeToTail() { no usages new *
    FunctionNode newNode = new FunctionNode(new FunctionPoint());
    FunctionNode last = head.prev;

    last.next = newNode;
    newNode.prev = last;
    newNode.next = head;
    head.prev = newNode;

    size++;
    return newNode;
}

```

Метод addNodeByIndex(int index): Добавляет новый элемент в указанную позицию.(например, для вставки между существующими точками).

Метод deleteNodeByIndex(int index): Удаляет узел по индексу.

```
private FunctionNode addNodeByIndex(int index) { no usages new *
    if (index < 0 || index > size) {
        throw new FunctionPointIndexOutOfBoundsException("Неверный индекс: " + index);
    }

    FunctionNode nextNode = (index == size) ? head : getNodeByIndex(index);
    FunctionNode prevNode = nextNode.prev;

    FunctionNode newNode = new FunctionNode(new FunctionPoint());
    newNode.prev = prevNode;
    newNode.next = nextNode;

    prevNode.next = newNode;
    nextNode.prev = newNode;

    size++;
    return newNode;
}

private FunctionNode deleteNodeByIndex(int index) { no usages new *
    if (size < 3) {
        throw new IllegalStateException("Нельзя удалить точку – останется меньше трёх");
    }

    FunctionNode node = getNodeByIndex(index);
    node.prev.next = node.next;
    node.next.prev = node.prev;

    size--;
    return node;
}
```

## Задание 5

Нужно, чтобы класс `LinkedListTabulatedFunction` имел два конструктора, аналогичных `ArrayTabulatedFunction`: Конструктор с границами и количеством точек, Конструктор с границами и массивом значений

```

public LinkedListTabulatedFunction(double leftX, double rightX, int pointsCount) { no usages new
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть меньше правой");
    }
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Количество точек должно быть не меньше двух");
    }

    this.head = new FunctionNode( point: null);
    head.next = head;
    head.prev = head;
    this.size = 0;

    double step = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; i++) {
        FunctionNode node = addNodeToTail();
        node.point.setX(leftX + i * step);
        node.point.setY(0.0);
    }
}

public LinkedListTabulatedFunction(double leftX, double rightX, double[] values) { no usages new
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть меньше правой");
    }
    if (values.length < 2) {
        throw new IllegalArgumentException("Количество точек должно быть не меньше двух");
    }

    this.head = new FunctionNode( point: null);
    head.next = head;
    head.prev = head;
    this.size = 0;

    double step = (rightX - leftX) / (values.length - 1);
    for (int i = 0; i < values.length; i++) {
        FunctionNode node = addNodeToTail();
        node.point.setX(leftX + i * step);
        node.point.setY(values[i]);
    }
}

```

Реализуем метод `getPoint(int index)`: Используем метод `getNodeByIndex(index)`.

Проверяем индекс и выбрасываем `FunctionPointIndexOutOfBoundsException`.

Возвращаем новый объект `FunctionPoint` (копию).

```

public FunctionPoint getPoint(int index) { no usages new *
    FunctionNode node = getNodeByIndex(index);
    return new FunctionPoint(node.point);
}

```

Реализуем методы `getPointX`, `getPointY`, `setPointY`:

getPointX / getPointY – просто возвращают значения из узла.

setPointY – проверка не нужна, можно просто изменить значение.

```
public double getPointX(int index) { 7 usages new *
|   return getNodeByIndex(index).point.getX();
}

public double getPointY(int index) { 2 usages new *
|   return getNodeByIndex(index).point.getY();
}

public void setPointY(int index, double y) { no usages new *
|   getNodeByIndex(index).point.setY(y);
}
```

Реализуем методы setPoint и setPointX с проверками: Проверяем правильность порядка X (не нарушает порядок соседних точек). Если нарушает – выбрасываем InappropriateFunctionPointException.

```
public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException { 1 usage & SultanovaArina *
|   FunctionNode node = getNodeByIndex(index);
|   double x = point.getX();

|   // Проверка соседей
|   if ((index > 0 && (x < getNodeByIndex(index - 1).point.getX() || Math.abs(x - getNodeByIndex(index - 1).point.getX()) < EPS)) ||
|       (index < size - 1 && (x > getNodeByIndex(index + 1).point.getX() || Math.abs(x - getNodeByIndex(index + 1).point.getX()) < EPS))) {
|     throw new InappropriateFunctionPointException("Неверная координата X");
|   }

|   node.point = new FunctionPoint(point);
}

public void setPointX(int index, double x) throws InappropriateFunctionPointException { 1 usage & SultanovaArina *
|   FunctionNode node = getNodeByIndex(index);

|   if ((index > 0 && (x < getNodeByIndex(index - 1).point.getX() || Math.abs(x - getNodeByIndex(index - 1).point.getX()) < EPS)) ||
|       (index < size - 1 && (x > getNodeByIndex(index + 1).point.getX() || Math.abs(x - getNodeByIndex(index + 1).point.getX()) < EPS))) {
|     throw new InappropriateFunctionPointException("Неверное значение X");
|   }

|   node.point.setX(x);
}
```

Реализуем addPoint: Проверим уникальность X (ни одна точка не должна совпадать). Находим позицию в списке и вставляем через addNodeByIndex.

```

public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException { 2 usages  ↗ SultanovaArina *
    double x = point.getX();

    for (int i = 0; i < size; i++) {
        if (Math.abs(getNodeByIndex(i).point.getX() - x) < EPS) {
            throw new InappropriateFunctionPointException("Точка с таким X уже существует");
        }
    }

    int insertIndex = 0;
    while (insertIndex < size && getPointX(insertIndex) < x && Math.abs(getPointX(insertIndex) - x) >= EPS) {
        insertIndex++;
    }

    FunctionNode node = addNodeByIndex(insertIndex);
    node.point = new FunctionPoint(point);
}

```

Реализуем deletePoint: Проверим, чтобы оставалось не меньше 3 точек.  
Используем deleteNodeByIndex.

```

public void deletePoint(int index) { no usages  new *
    if (size <= 2) {
        throw new IllegalStateException("Нельзя удалить точку – останется меньше трёх");
    }
    deleteNodeByIndex(index);
}

```

Реализуем getFunctionValue: Линейная интерполяция между точками.  
Используем getPointX и getPointY.

```

public double getFunctionValue(double x) { 1 usage  ↗ SultanovaArina *
    if (x < getLeftDomainBorder() - EPS || x > getRightDomainBorder() + EPS)
        return Double.NaN;

    for (int i = 0; i < size - 1; i++) {
        double x0 = getPointX(i);
        double x1 = getPointX(index: i + 1);
        double y0 = getPointY(i);
        double y1 = getPointY(index: i + 1);

        if (Math.abs(x - x0) < EPS) return y0;
        if (Math.abs(x - x1) < EPS) return y1;
        if (x > x0 && x < x1) {
            return y0 + (y1 - y0) * (x - x0) / (x1 - x0);
        }
    }
    return Double.NaN;
}

```

Реализуем методы границ и количества точек

```

public int getPointsCount() { no usages new *
    return size;
}

public double getLeftDomainBorder() { no usages new *
    return getPointX(index: 0);
}

public double getRightDomainBorder() { no usages new *
    return getPointX(index: size - 1);
}

```

## Задание 6

Создаём интерфейс TabulatedFunction

Что делаем:

Интерфейс содержит все методы, которые должны быть доступны для работы с табулированной функцией, но без реализации.

Методы включают:

Получение и изменение точек (getPoint, setPoint, getPointX, setPointX, getPointY, setPointY)

Добавление и удаление точек (addPoint, deletePoint)

Границы области определения (getLeftDomainBorder, getRightDomainBorder)

Количество точек (getPointsCount)

Значение функции (getFunctionValue)

```

package functions;

public interface TabulatedFunction { no usages 2 implementations & SultanovaArina *
    int getPointsCount(); no usages 2 implementations new *
    double getPointX(int index); no usages 2 implementations new *
    double getPointY(int index); no usages 2 implementations new *
    void setPointX(int index, double x) throws InappropriateFunctionPointException; no usages 2 implementations new *
    void setPointY(int index, double y); no usages 2 implementations new *
    FunctionPoint getPoint(int index); no usages 2 implementations new *
    void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException; no usages 2 implementations new *
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException; no usages 2 implementations new *
    void deletePoint(int index); no usages 2 implementations new *
    double getLeftDomainBorder(); no usages 2 implementations new *
    double getRightDomainBorder(); no usages 2 implementations new *
    double getFunctionValue(double x); no usages 2 implementations new *
}

```

Реализуем интерфейс в ArrayTabulatedFunction

## Реализуем интерфейс в LinkedListTabulatedFunction

```
public class ArrayTabulatedFunction implements TabulatedFunction { no usages new *
```

```
public class LinkedListTabulatedFunction implements TabulatedFunction { new *
```

## Задание 7

Проверяем работу написанных классов.

```
Main.java x FunctionPointIndexOutOfBoundsException.java InappropriateFunctionPointException.java FunctionPoint.java ArrayTabulatedFunction.java
1 import functions.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         double[] values = {0, 1, 8, 27, 64};
6
7         System.out.println("TECT ArrayTabulatedFunction");
8         TabulatedFunction arrayFunction = new ArrayTabulatedFunction(leftX: 0, rightX: 4, values);
9         testFunction(arrayFunction);
10        testExceptions(arrayFunction);
11
12        System.out.println("\nTECT LinkedListTabulatedFunction");
13        TabulatedFunction listFunction = new LinkedListTabulatedFunction(leftX: 0, rightX: 4, values);
14        testFunction(listFunction);
15        testExceptions(listFunction);
16    }
17
18    // Проверка основных операций: получение, изменение, удаление, добавление
19    public static void testFunction(TabulatedFunction func) {
20        System.out.println("\nТекущее состояние функции (" + func.getClass().getSimpleName() + ")");
21        printFunction(func);
22
23        System.out.println("\nПроверка вычисления значения функции:");
24        System.out.println("f(2.7) = " + func.getFunctionValue(x: 2.7));
25
26        System.out.println("\nИзменение значения Y:");
27        System.out.println("Меняем Y точки с индексом 3 на 50.0");
28        func.setPointY(index: 3, y: 50.0);
29        printFunction(func);
30        func.setPointY(index: 3, y: 27.0); // возвращаем исходное значение
31
32        System.out.println("\nУдаление точки:");
33        System.out.println("Удаляем точку с индексом 1");
34        func.deletePoint(index: 1);
35        printFunction(func);
36
37        System.out.println("\nДобавление точки:");
38        System.out.println("Добавляем точку (1.0; 1.0) обратно");
39        try {
40            func.addPoint(new FunctionPoint(x: 1.0, y: 1.0));
41        } catch (InappropriateFunctionPointException e) {
42            System.out.println("Не удалось добавить точку: " + e.getMessage());
43        }
44        printFunction(func);
45
46        System.out.println("\nПроверка setPoint() с некорректным X:");
47        try {
48            System.out.println("Попытка заменить точку с индексом 2 на (0.5; 0.125)");
49            func.setPoint(index: 2, new FunctionPoint(x: 0.5, y: 0.125));
50        } catch (InappropriateFunctionPointException e) {
51            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
52        }
53    }
54 }
```

```
54     }
55
56     // Проверка выбрасывания исключений
57     @Test
58     public static void testExceptions(TabulatedFunction func) { 2 usages ▾ SultanovaArina*
59         System.out.println("\n Тестирование исключений для " + func.getClass().getSimpleName() );
60
61         int totalPoints = func.getPointsCount();
62
63         // 1. Индекс за пределами
64         try {
65             System.out.println("Попытка получить точку с индексом -1");
66             func.getPoint( index: -1);
67         } catch (FunctionPointIndexOutOfBoundsException e) {
68             System.out.println("Перехвачено: " + e.getClass().getSimpleName());
69         }
70
71         try {
72             System.out.println("Попытка получить точку с индексом " + totalPoints);
73             func.getPoint(totalPoints);
74         } catch (FunctionPointIndexOutOfBoundsException e) {
75             System.out.println("Перехвачено: " + e.getClass().getSimpleName());
76         }
77
78         // 2. Некорректное изменение X
79         try {
80             System.out.println("Попытка изменить X точки 2 на 0.8 (нарушает порядок)");
81             func.setPointX( index: 2,  x: 0.8);
82         } catch (InappropriateFunctionPointException e) {
83             System.out.println("Перехвачено: " + e.getClass().getSimpleName());
84         }
85
86         // 3. Добавление точки с существующим X
87         try {
88             System.out.println("Попытка добавить точку с X=3.0 (уже существует)");
89             func.addPoint(new FunctionPoint( x: 3.0,  y: 27.0));
90         } catch (InappropriateFunctionPointException e) {
91             System.out.println("Перехвачено: " + e.getClass().getSimpleName());
92         }
93
94         // 4. Удаление из функции с 2 точками
95         try {
96             System.out.println("Попытка удалить точку из функции с двумя точками");
97             TabulatedFunction smallFunc;
98             if (func instanceof ArrayTabulatedFunction) {
99                 smallFunc = new ArrayTabulatedFunction( leftX: 0,  rightX: 1,  new double[]{0,  1});
100            } else {
101                smallFunc = new LinkedListTabulatedFunction( leftX: 0,  rightX: 1,  new double[]{0,  1});
102            }
103            smallFunc.deletePoint( index: 0);
104        } catch (IllegalStateException e) {
105            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
106        }
107    }
108 }
```

```
106
107     // 5. Некорректные параметры конструктора
108     try {
109         System.out.println("Попытка создать функцию с левым >= правого и меньше двух точек");
110         TabulatedFunction wrongFunc = new ArrayTabulatedFunction(leftX: 5, rightX: 1, pointsCount: 1);
111     } catch (IllegalArgumentException e) {
112         System.out.println("Перехвачено: " + e.getClass().getSimpleName());
113     }
114 }
115
116 // Вывод точек функции
117 @
118 public static void printFunction(TabulatedFunction func) { 4 usages & SultanovaArina*
119     int totalPoints = func.getPointsCount();
120     System.out.println("Функция состоит из " + totalPoints + " точек:");
121     for (int i = 0; i < totalPoints; i++) {
122         FunctionPoint p = func.getPoint(i);
123         System.out.println("Точка " + i + ": (" + p.getX() + "; " + p.getY() + ")");
124     }
125 }
126 }
```

```
Run Main ×
C | ☷ : 
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:D:\java\IntelliJ IDEA
TECT ArrayTabulatedFunction

Текущее состояние функции (ArrayTabulatedFunction)
Функция состоит из 5 точек:
Точка 0: (0.0; 0.0)
Точка 1: (1.0; 1.0)
Точка 2: (2.0; 8.0)
Точка 3: (3.0; 27.0)
Точка 4: (4.0; 64.0)

Проверка вычисления значения функции:
f(2.7) = 21.30000000000004

Изменение значения Y:
Меняем Y точки с индексом 3 на 50.0
Функция состоит из 5 точек:
Точка 0: (0.0; 0.0)
Точка 1: (1.0; 1.0)
Точка 2: (2.0; 8.0)
Точка 3: (3.0; 50.0)
Точка 4: (4.0; 64.0)

Удаление точки:
Удаляем точку с индексом 1
Функция состоит из 4 точек:
Точка 0: (0.0; 0.0)
Точка 1: (2.0; 8.0)
Точка 2: (3.0; 27.0)
Точка 3: (4.0; 64.0)

Добавление точки:
Добавляем точку (1.0; 1.0) обратно
Функция состоит из 5 точек:
Точка 0: (0.0; 0.0)
Точка 1: (1.0; 1.0)
Точка 2: (2.0; 8.0)
Точка 3: (3.0; 27.0)
Точка 4: (4.0; 64.0)

Проверка setPoint() с некорректным X:
Попытка заменить точку с индексом 2 на (0.5; 0.125)
Перехвачено: InappropriateFunctionPointException
```

```
Тестирование исключений для ArrayTabulatedFunction
Попытка получить точку с индексом -1
Перехвачено: FunctionPointIndexOutOfBoundsException
Попытка получить точку с индексом 5
Перехвачено: FunctionPointIndexOutOfBoundsException
Попытка изменить X точки 2 на 0.8 (нарушает порядок)
Перехвачено: InappropriateFunctionPointException
Попытка добавить точку с X=3.0 (уже существует)
Перехвачено: InappropriateFunctionPointException
Попытка удалить точку из функции с двумя точками
Перехвачено: IllegalStateException
Попытка создать функцию с левым >= правого и меньше двух точек
Перехвачено: IllegalArgumentException
```

#### TEST LinkedListTabulatedFunction

```
Текущее состояние функции (LinkedListTabulatedFunction)
Функция состоит из 5 точек:
Точка 0: (0.0; 0.0)
Точка 1: (1.0; 1.0)
Точка 2: (2.0; 8.0)
Точка 3: (3.0; 27.0)
Точка 4: (4.0; 64.0)
```

```
Проверка вычисления значения функции:
f(2.7) = 21.300000000000004
```

```
Изменение значения Y:
Меняем Y точки с индексом 3 на 50.0
Функция состоит из 5 точек:
Точка 0: (0.0; 0.0)
Точка 1: (1.0; 1.0)
Точка 2: (2.0; 8.0)
Точка 3: (3.0; 50.0)
Точка 4: (4.0; 64.0)
```

```
Удаление точки:
Удаляем точку с индексом 1
Функция состоит из 4 точек:
Точка 0: (0.0; 0.0)
Точка 1: (2.0; 8.0)
Точка 2: (3.0; 27.0)
Точка 3: (4.0; 64.0)
```

```
Добавление точки:
Добавляем точку (1.0; 1.0) обратно
Функция состоит из 5 точек:
Точка 0: (0.0; 0.0)
Точка 1: (1.0; 1.0)
Точка 2: (2.0; 8.0)
Точка 3: (3.0; 27.0)
Точка 4: (4.0; 64.0)
```

```
Проверка setPoint() с некорректным X:  
Попытка заменить точку с индексом 2 на (0.5; 0.125)  
Перехвачено: InappropriateFunctionPointException  
  
Тестирование исключений для LinkedListTabulatedFunction  
Попытка получить точку с индексом -1  
Перехвачено: FunctionPointIndexOutOfBoundsException  
Попытка получить точку с индексом 5  
Перехвачено: FunctionPointIndexOutOfBoundsException  
Попытка изменить X точки 2 на 0.8 (нарушает порядок)  
Перехвачено: InappropriateFunctionPointException  
Попытка добавить точку с X=3.0 (уже существует)  
Перехвачено: InappropriateFunctionPointException  
Попытка удалить точку из функции с двумя точками  
Перехвачено: IllegalStateException  
Попытка создать функцию с левым >= правого и меньше двух точек  
Перехвачено: IllegalArgumentException  
  
Process finished with exit code 0
```