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.

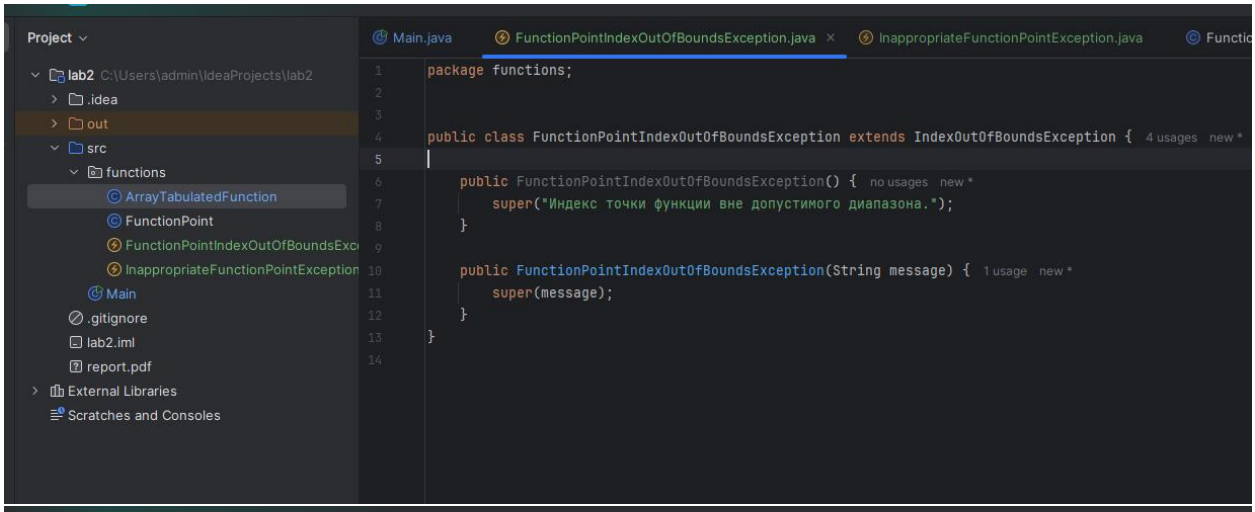
Самара 2025

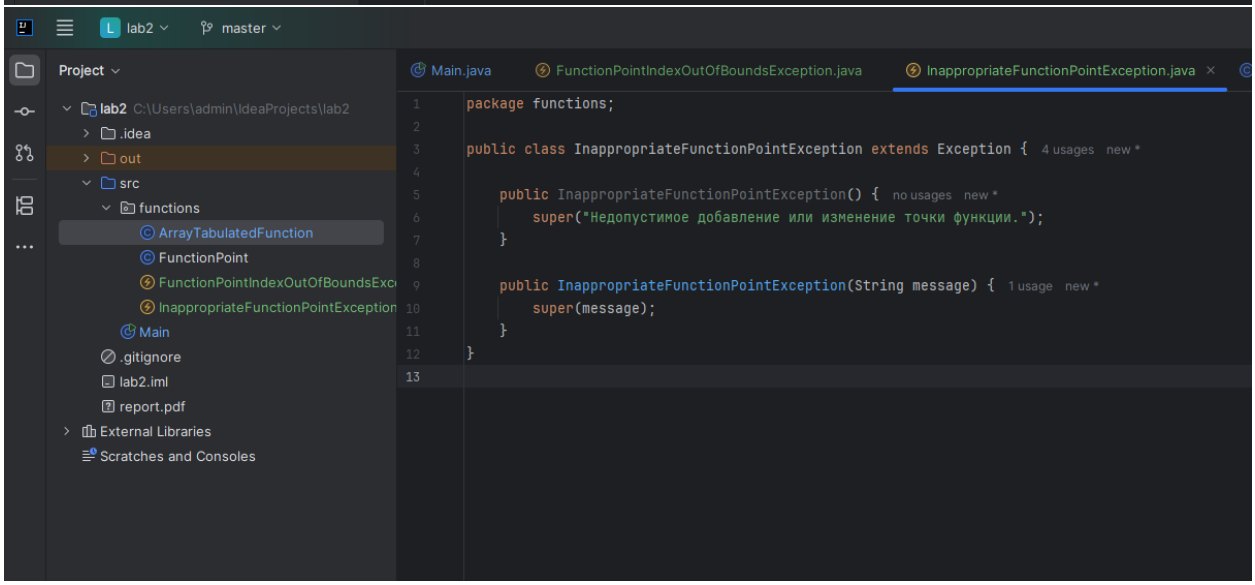
Задание 2

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.





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.

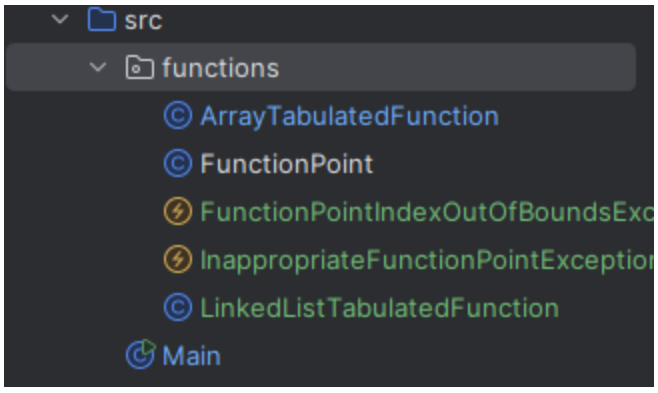
В setPoint() и setPointX() добавим проверку, чтобы новая x находилась между соседними.

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

```java
    public FunctionPoint getPoint(int index) {  no usages  SultanovaArina *
        checkIndex(index);
        return new FunctionPoint(points[index]);
    }

    public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException {  no usages  SultanovaArina *
        checkIndex(index);
        double x = point.getX();
        if ((index > 0 && x <= points[index - 1].getX()) ||
                (index < size - 1 && x >= points[index + 1].getX())) {
            throw new InappropriateFunctionPointException("Неверная координата X для данной позиции");
        }
        points[index] = new FunctionPoint(point);
    }


    public double getPointX(int index) {  no usages  SultanovaArina *
        checkIndex(index);
        return points[index].getX();
    }
    public void setPointX(int index, double x) throws InappropriateFunctionPointException {  no usages  SultanovaArina *
        checkIndex(index);

        if ((index > 0 && x <= points[index - 1].getX()) ||
                (index < size - 1 && x >= points[index + 1].getX())) {
            throw new InappropriateFunctionPointException("Неверное значение X — нарушается порядок точек");
        }

        points[index].setX(x);
    }


    public double getPointY(int index) {  no usages  SultanovaArina *
        checkIndex(index);
        return points[index].getY();
    }

    public void setPointY(int index, double y) {  no usages  SultanovaArina *
        checkIndex(index);
        points[index].setY(y);
    }

    public void deletePoint(int index) {  no usages  SultanovaArina *
        checkIndex(index);
        if (size < 3) {
            throw new IllegalStateException("Нельзя удалить точку — останется меньше трёх");
        }
        for (int i = index; i < size - 1; i++) {
            points[i] = points[i + 1];
        }
        size--;
    }
```

Задание 4

В пакете functions создаём новый файл:



Объявим класс и сделаем структуру головы списка

```
package functions;

public class LinkedListTabulatedFunction {  no usages  new *
    private FunctionNode head;  11 usages
    private int size;  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

```java
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).

```java
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 {  no usage
    FunctionNode node = getNodeByIndex(index);
    double x = point.getX();

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

    node.point = new FunctionPoint(point);
}

public void setPointX(int index, double x) throws InappropriateFunctionPointException {  no usages  new *
    FunctionNode node = getNodeByIndex(index);

    if ((index > 0 && x <= getNodeByIndex(index - 1).point.getX()) ||
            (index < size - 1 && x >= getNodeByIndex(index + 1).point.getX())) {
        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.

```java
public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {  no usages  new *
    double x = point.getX();

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

    int insertIndex = 0;
    while (insertIndex < size && getPointX(insertIndex) < x) {
        insertIndex++;
    }

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

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

```java
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.

```java
public double getFunctionValue(double x) {  no usages  new *
    if (x < getPointX( index: 0) || x > getPointX( index: size - 1)) {
        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 (x == x0) return y0;
        if (x == x1) 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

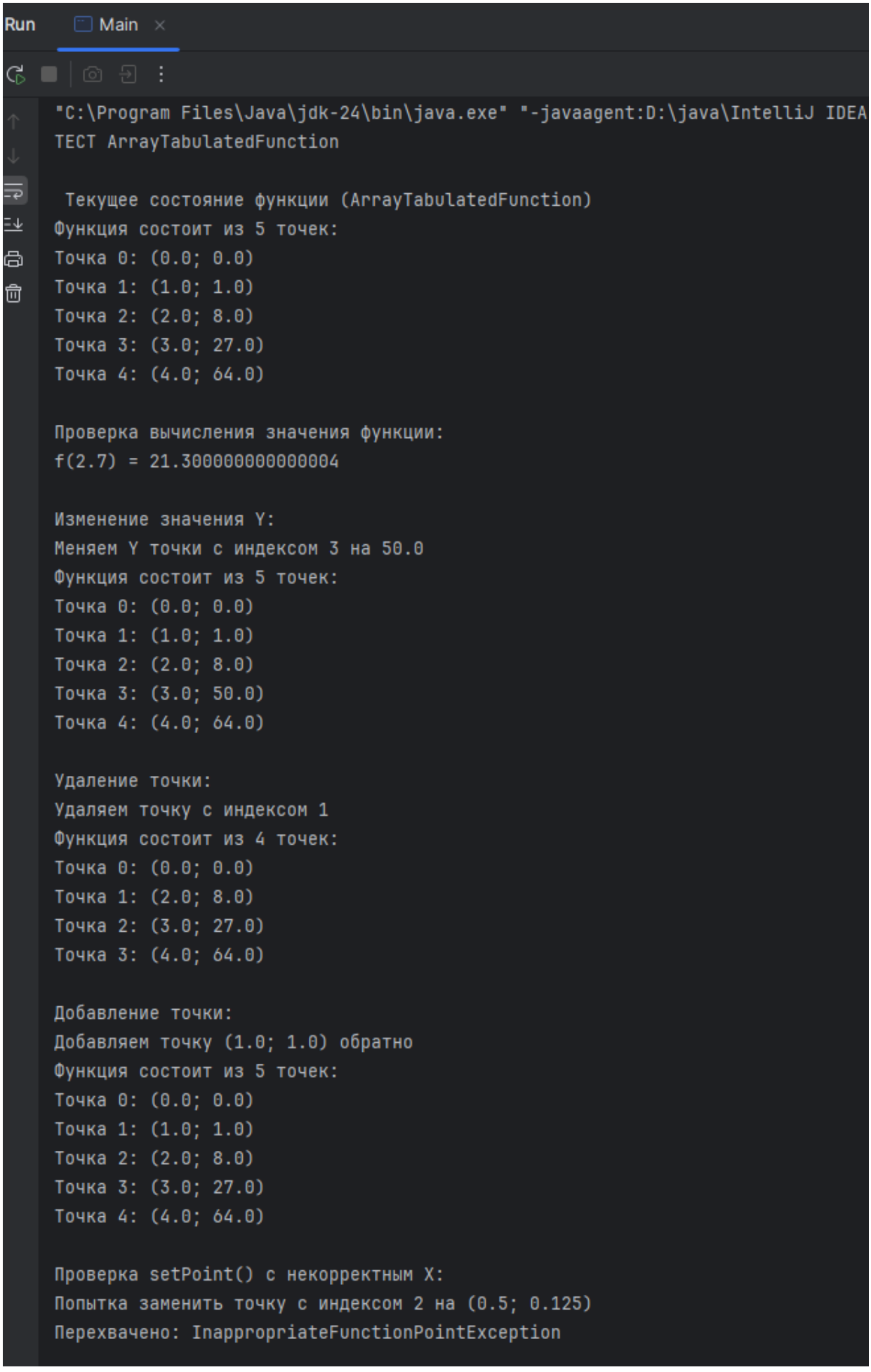
Проверяем работу написанных классов.

Main.java | FunctionPointIndexOutOfBoundsException.java | InappropriateFunctionPointException.java | FunctionPoint.java | ArrayTabu

```
import functions.*;

public class Main {  SultanovaArina *
    public static void main(String[] args) {  SultanovaArina *

        double[] values = {0, 1, 8, 27, 64};

        System.out.println("ТЕСТ ArrayTabulatedFunction ");
        TabulatedFunction arrayFunction = new ArrayTabulatedFunction( leftX: 0,  rightX: 4, values);
        testFunction(arrayFunction);
        testExceptions(arrayFunction);

        System.out.println("\nТЕСТ LinkedListTabulatedFunction");
        TabulatedFunction listFunction = new LinkedListTabulatedFunction( leftX: 0,  rightX: 4, values);
        testFunction(listFunction);
        testExceptions(listFunction);
    }

    // Проверка основных операций: получение, изменение, удаление, добавление
    public static void testFunction(TabulatedFunction func) {  2 usages  SultanovaArina *
        System.out.println("\n Текущее состояние функции (" + func.getClass().getSimpleName() + ") ");
        printFunction(func);

        System.out.println("\nПроверка вычисления значения функции:");
        System.out.println("f(2.7) = " + func.getFunctionValue( x: 2.7));

        System.out.println("\nИзменение значения Y:");
        System.out.println("Меняем Y точки с индексом 3 на 50.0");
        func.setPointY( index: 3,  y: 50.0);
        printFunction(func);
        func.setPointY( index: 3,  y: 27.0); // возвращаем исходное значение

        System.out.println("\nУдаление точки:");
        System.out.println("Удаляем точку с индексом 1");
        func.deletePoint( index: 1);
        printFunction(func);

        System.out.println("\nДобавление точки:");
        System.out.println("Добавляем точку (1.0; 1.0) обратно");
        try {
            func.addPoint(new FunctionPoint( x: 1.0,  y: 1.0));
        } catch (InappropriateFunctionPointException e) {
            System.out.println("Не удалось добавить точку: " + e.getMessage());
        }
        printFunction(func);

        System.out.println("\nПроверка setPoint() с некорректным X:");
        try {
            System.out.println("Попытка заменить точку с индексом 2 на (0.5; 0.125)");
            func.setPoint( index: 2, new FunctionPoint( x: 0.5,  y: 0.125));
        } catch (InappropriateFunctionPointException e) {
            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
        }
    }
```

```java
    }

    // Проверка выбрасывания исключений
    public static void testExceptions(TabulatedFunction func) {  2 usages  SultanovaArina *
        System.out.println("\n Тестирование исключений для " + func.getClass().getSimpleName() );

        int totalPoints = func.getPointsCount();

        // 1. Индекс за пределами
        try {
            System.out.println("Попытка получить точку с индексом -1");
            func.getPoint( index: -1);
        } catch (FunctionPointIndexOutOfBoundsException e) {
            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
        }

        try {
            System.out.println("Попытка получить точку с индексом " + totalPoints);
            func.getPoint(totalPoints);
        } catch (FunctionPointIndexOutOfBoundsException e) {
            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
        }

        // 2. Некорректное изменение X
        try {
            System.out.println("Попытка изменить X точки 2 на 0.8 (нарушает порядок)");
            func.setPointX( index: 2,  x: 0.8);
        } catch (InappropriateFunctionPointException e) {
            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
        }

        // 3. Добавление точки с существующим X
        try {
            System.out.println("Попытка добавить точку с X=3.0 (уже существует)");
            func.addPoint(new FunctionPoint( x: 3.0,  y: 27.0));
        } catch (InappropriateFunctionPointException e) {
            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
        }

        // 4. Удаление из функции с 2 точками
        try {
            System.out.println("Попытка удалить точку из функции с двумя точками");
            TabulatedFunction smallFunc;
            if (func instanceof ArrayTabulatedFunction) {
                smallFunc = new ArrayTabulatedFunction( leftX: 0,  rightX: 1, new double[]{0, 1});
            } else {
                smallFunc = new LinkedListTabulatedFunction( leftX: 0,  rightX: 1, new double[]{0, 1});
            }
            smallFunc.deletePoint( index: 0);
        } catch (IllegalStateException e) {
            System.out.println("Перехвачено: " + e.getClass().getSimpleName());
        }
```

```
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     public static void printFunction(TabulatedFunction func) {  4 usages  SultanovaArina *
118         int totalPoints = func.getPointsCount();
119         System.out.println("Функция состоит из " + totalPoints + " точек:");
120         for (int i = 0; i < totalPoints; i++) {
121             FunctionPoint p = func.getPoint(i);
122             System.out.println("Точка " + i + ": (" + p.getX() + "; " + p.getY() + ")");
123         }
124     }
125 }
126
```


Run
Main
"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) = 21.300000000000004
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 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

ТЕСТ 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
```