

Лабораторная работа №3

Выполнила: Аксенова Алина Владимировна
Номер группы: 6204-010302D
Дата выполнения: 2025г.

Оглавление

Задание1.....	3
Задание2.....	3
Задание3.....	4
Задание4.....	6
Задание5.....	6
Задание6.....	8
Задание7.....	9

Задание 1

Я изучила документацию по классам исключений Java. Я ознакомилась с иерархией исключений, особенностями проверяемых и непроверяемых исключений, а также с конкретными случаями использования каждого из указанных классов: Exception, IndexOutOfBoundsException, ArrayIndexOutOfBoundsException, IllegalArgumentException и IllegalStateException.

Задание 2

Для реализации класса FunctionPointIndexOutOfBoundsException я создала файл FunctionPointIndexOutOfBoundsException.java в пакете functions. Этот класс наследуется от IndexOutOfBoundsException, что делает его непроверяемым исключением. Я реализовала два конструктора: конструктор по умолчанию без параметров и конструктор с параметром message для передачи текста ошибки. Это исключение будет выбрасываться при попытке обращения к несуществующему индексу точки функции.

Для реализации класса InappropriateFunctionPointException я создала файл InappropriateFunctionPointException.java в пакете functions. Изначально этот класс наследовался от Exception, но в процессе тестирования я изменила наследование на RuntimeException для упрощения обработки исключений в коде. Класс содержит два конструктора аналогично предыдущему исключению и будет использоваться для случаев нарушения порядка точек или попытки добавления точки с существующей координатой X.

```
package functions;

public class FunctionPointIndexOutOfBoundsException extends
IndexOutOfBoundsException {
```

```
public FunctionPointIndexOutOfBoundsException() {
    super();
}

public FunctionPointIndexOutOfBoundsException(String message) {
    super(message);
}
}
```

```
package functions;

public class InappropriateFunctionPointException extends
RuntimeException {
    public InappropriateFunctionPointException() {
        super();
    }

    public InappropriateFunctionPointException(String message) {
        super(message);
    }
}
```

Задание 3

Для модификации класса TabulatedFunction я переименовала его в ArrayTabulatedFunction и добавила обработку исключений согласно заданию. В конструкторах я добавила проверки на корректность параметров: если левая граница больше или равна правой, выбрасывается IllegalArgumentException, также если количество точек меньше двух. Это гарантирует создание объекта только с корректными параметрами.

В методах работы с точками (getPoint, setPoint, getPointX, setPointX, getPointY, setPointY, deletePoint) я добавила проверки выхода за границы массива точек. При обнаружении недопустимого индекса выбрасывается FunctionPointIndexOutOfBoundsException. Для методов

setPoint и setPointX я реализовала проверку упорядоченности точек с использованием машинного эпсилона ($1e-10$) для сравнения вещественных чисел. Если новая координата X нарушает порядок точек, выбрасывается InappropriateFunctionPointException.

В методе addPoint я добавила проверку на дублирование координат X, также с использованием машинного эпсилона. В методе deletePoint добавила проверку на минимальное количество точек - если после удаления останется менее 3 точек, выбрасывается IllegalStateException.

```
public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
    if (leftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть меньше правой");
    }
    if (pointsCount < 2) {
        throw new IllegalArgumentException("Количество точек должно быть не менее 2");
    }

    this.pointsCount = pointsCount;
    this.points = new FunctionPoint[pointsCount + 5];

    double step = (rightX - leftX) / (pointsCount - 1);

    for (int i = 0; i < pointsCount; i++) {
        double x = leftX + i * step;
        points[i] = new FunctionPoint(x, 0);
    }
}
```

```
public void setPoint(int index, FunctionPoint point) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException("Индекс точки выходит за границы");
    }

    // Проверка что x находится между соседями (машинный эпсилон)
    if (index > 0 && point.getX() <= points[index - 1].getX() +
1e-10) {
```

```
        throw new InappropriateFunctionPointException("Координата X должна быть больше предыдущей точки");
    }
    if (index < pointsCount - 1 && point.getX() >= points[index + 1].getX() - 1e-10) {
        throw new InappropriateFunctionPointException("Координата X должна быть меньше следующей точки");
    }

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

Задание 4-5

Для реализации `LinkedListTabulatedFunction` я создала класс, который также реализует интерфейс `TabulatedFunction`, но использует связный список вместо массива. Внутри класса я объявила приватный внутренний класс `FunctionNode`, который представляет узел списка.

Каждый узел содержит поле `point` для хранения данных, а также ссылки `prev` и `next` на предыдущий и следующий узлы.

Я реализовала двусвязный циклический список с выделенной головой. Голова списка не хранит данные и всегда присутствует в списке. Для оптимизации доступа к элементам я добавила поля `lastAccessedNode` и `lastAccessedIndex`, которые кэшируют последний доступный узел. В методе `getNodeByIndex` я реализовала алгоритм, который начинает поиск с последнего доступного узла, если это эффективнее, чем начинать с головы списка.

Также я реализовала вспомогательные методы для работы со списком: `addNodeToTail` для добавления в конец, `addNodeByIndex` для вставки по индексу и `deleteNodeByIndex` для удаления узла. Все публичные методы класса реализуют тот же интерфейс, что и

ArrayTabulatedFunction, и выбрасывают те же исключения в аналогичных ситуациях.

```
private class FunctionNode {  
    FunctionPoint point;  
    FunctionNode prev;  
    FunctionNode next;  
  
    FunctionNode(FunctionPoint point) {  
        this.point = point;  
    }  
}
```

```
private FunctionNode getNodeByIndex(int index) {  
    if (index < 0 || index >= pointsCount) {  
        throw new FunctionPointIndexOutOfBoundsException("Индекс  
выходит за границы");  
    }  
  
    // Оптимизация: начинаем с последнего доступного узла  
    FunctionNode current;  
    int currentIndex;  
  
    if (lastAccessedIndex != -1 && Math.abs(index -  
lastAccessedIndex) < Math.abs(index)) {  
        current = lastAccessedNode;  
        currentIndex = lastAccessedIndex;  
    } else {  
        current = head.next;  
        currentIndex = 0;  
    }  
  
    // Двигаемся к нужному узлу  
    while (currentIndex < index) {  
        current = current.next;  
        currentIndex++;  
    }  
    while (currentIndex > index) {  
        current = current.prev;  
        currentIndex--;  
    }
```

```
    lastAccessedNode = current;
    lastAccessedIndex = currentIndex;
    return current;
}
```

```
private FunctionNode addNodeToTail() {
    FunctionNode newNode = new FunctionNode(null);
    newNode.next = head;
    newNode.prev = head.prev;
    head.prev.next = newNode;
    head.prev = newNode;
    pointsCount++;
    return newNode;
}
```

Задание 6

Для создания интерфейса TabulatedFunction я объявила все общие методы, которые должны быть реализованы в классах ArrayTabulatedFunction и LinkedListTabulatedFunction. Интерфейс включает методы для работы с функцией (getLeftDomainBorder, getRightDomainBorder, getFunctionValue), методы для работы с точками (getPointsCount, getPoint, setPoint, getPointX, setPointX, getPointY, setPointY) и методы изменения количества точек (deletePoint, addPoint).

После создания интерфейса я изменила оба класса функций, чтобы они реализовывали этот интерфейс. Это позволяет использовать полиморфизм - работать с функциями через ссылку типа TabulatedFunction, не завися от конкретной реализации.

```
package functions;

public interface TabulatedFunction {
```

```
// Методы для работы с функцией
double getLeftDomainBorder();
double getRightDomainBorder();
double getFunctionValue(double x);

// Методы для работы с точками
int getPointsCount();
FunctionPoint getPoint(int index);
void setPoint(int index, FunctionPoint point);
double getPointX(int index);
void setPointX(int index, double x);
double getPointY(int index);
void setPointY(int index, double y);

// Методы изменения количества точек
void deletePoint(int index);
void addPoint(FunctionPoint point);
}
```

```
public class ArrayTabulatedFunction implements TabulatedFunction {
    // реализация методов интерфейса
}

public class LinkedListTabulatedFunction implements
TabulatedFunction {
    // реализация методов интерфейса
}
```

Задание 7

Для тестирования созданных классов я разработала класс Main с точкой входа программы. В методе main я создала тесты для обеих реализаций табулированных функций. Я протестировала создание функций, получение значений, работу с точками, а также проверила корректность выбрасывания исключений при различных ошибочных сценариях.

Тестирование показало, что оба класса работают корректно, все исключения выбрасываются в предусмотренных случаях, а вычисления значений функций производятся правильно с использованием линейной интерполяции

```
private static void testExceptions() {
    System.out.println("\nТестирование корректных исключений:");

    try {
        // Некорректные границы
        new ArrayTabulatedFunction(5, 0, 3);
        System.out.println("ОШИБКА: Не выброшено исключение для некорректных границ");
    } catch (IllegalArgumentException e) {
        System.out.println("✓ Корректно: " + e.getMessage());
    }

    try {
        // Выход за границы индекса
        TabulatedFunction func = new ArrayTabulatedFunction(0, 2, 3);
        func.getPoint(10);
        System.out.println("ОШИБКА: Не выброшено исключение для выхода за границы");
    } catch (FunctionPointIndexOutOfBoundsException e) {
        System.out.println("✓ Корректно: " + e.getMessage());
    }
}
```

```
private static void testArrayFunction() {
    try {
        // Создание функции через массив
        double[] values = {1.0, 4.0, 9.0, 16.0};
        TabulatedFunction arrayFunc = new ArrayTabulatedFunction(0, 3,
values);

        System.out.println("ArrayFunction: точки от " +
arrayFunc.getLeftDomainBorder() +
                    " до " + arrayFunc.getRightDomainBorder());

        // Вывод всех точек
        for (int i = 0; i < arrayFunc.getPointsCount(); i++) {
            FunctionPoint point = arrayFunc.getPoint(i);
            System.out.printf("Точка %d: (%.1f, %.1f)%n", i,
point.getX(), point.getY());
        }
    }
}
```

```

    }

    // Тестирование вычисления значения
    System.out.printf("f(1.5) = %.2f%n",
arrayFunc.getFunctionValue(1.5));

} catch (Exception e) {
    System.out.println("Ошибка в ArrayFunction: " + e.getMessage());
}
}

```

Вывод Main:

```

p:\vscode\ws\best\jdt_ws\jdt.ls-java-project\bin\main
== Лабораторная работа #3 ==

--- Тестирование ArrayTabulatedFunction ---
ArrayTabulatedFunction: точки от 0.0 до 4.0, количество точек: 5
Все точки функции:
    Точка 0: (0,0, 1,0)
    Точка 1: (1,0, 4,0)
    Точка 2: (2,0, 9,0)
    Точка 3: (3,0, 16,0)
    Точка 4: (4,0, 25,0)

Вычисление значений функции:
    f(0.0) = 1,00
    f(1.5) = 6,50
    f(2.0) = 9,00
    f(3.7) = 22,30

Модификация точек:
    После изменения: точка 2 = (2,0, 10,0)
    После добавления точки (2.5, 6.25):
    Количество точек: 6
    После удаления точки 1:
    Количество точек: 5

--- Тестирование LinkedListTabulatedFunction ---
LinkedListTabulatedFunction: точки от 0.0 до 4.0, количество точек: 5
Все точки функции:
    Точка 0: (0,0, 0,0)
    Точка 1: (1,0, 0,0)
    Точка 2: (2,0, 0,0)
    Точка 3: (3,0, 0,0)
    Точка 4: (4,0, 0,0)

```

Вычисление значений функции:

f(0.0) = 0,00
f(1.2) = 0,00
f(2.5) = 0,00
f(3.8) = 0,00

Модификация точек:

После изменения: точка 3 = (3,0, 15,0)

После добавления точки (1.5, 3.0):

Количество точек: 6

Все точки после модификаций:

Точка 0: (0,0, 0,0)
Точка 1: (1,0, 0,0)
Точка 2: (1,5, 3,0)
Точка 3: (2,0, 0,0)
Точка 4: (3,0, 15,0)
Точка 5: (4,0, 0,0)

--- Тестирование исключений ---

Тестирование корректных исключений:

Корректно: Левая граница должна быть меньше правой

Корректно: Количество точек должно быть не менее 2

Корректно: Индекс точки выходит за границы

Корректно: Индекс выходит за границы

Корректно: Координата X должна быть больше предыдущей точки

Корректно: Точка с такой координатой X уже существует

Корректно: Нельзя удалить точку: должно остаться минимум 2 точки

Корректно: Индекс точки выходит за границы

Все тесты исключений пройдены успешно!

PS C:\Users\Alina> []

Обновленный вывод Main:

```
==== Лабораторная работа #3 ===
```

```
--- Тестирование ArrayTabulatedFunction ---
```

```
Создана ArrayTabulatedFunction:
```

```
Текущий набор точек (5 точек):
```

```
    Точка 0: (0,00, 1,00)
```

```
    Точка 1: (1,00, 4,00)
```

```
    Точка 2: (2,00, 9,00)
```

```
    Точка 3: (3,00, 16,00)
```

```
    Точка 4: (4,00, 25,00)
```

```
Тестирование вычисления значений (с оптимизацией):
```

```
f(0.0) = 1,00 (точное совпадение - использует существующий Y)
```

```
f(1.0) = 4,00 (точное совпадение - использует существующий Y)
```

```
f(1.5) = 6,50 (интерполяция)
```

```
f(2.0) = 9,00 (точное совпадение - использует существующий Y)
```

```
f(3.7) = 22,30 (интерполяция)
```

```
Модификация точек:
```

```
После изменения точки 2 на (2.0, 10.0):
```

```
Текущий набор точек (5 точек):
```

```
    Точка 0: (0,00, 1,00)
```

```
    Точка 1: (1,00, 4,00)
```

```
    Точка 2: (2,00, 10,00)
```

```
    Точка 3: (3,00, 16,00)
```

```
    Точка 4: (4,00, 25,00)
```

```
После добавления точки (2.5, 6.25):
```

```
Текущий набор точек (6 точек):
```

```
    Точка 0: (0,00, 1,00)
```

```
    Точка 1: (1,00, 4,00)
```

```
    Точка 2: (2,00, 10,00)
```

```
    Точка 3: (2,50, 6,25)
```

```
    Точка 4: (3,00, 16,00)
```

```
    Точка 5: (4,00, 25,00)
```

После удаления точки 1:

Текущий набор точек (5 точек):

Точка 0: (0,00, 1,00)
Точка 1: (2,00, 10,00)
Точка 2: (2,50, 6,25)
Точка 3: (3,00, 16,00)
Точка 4: (4,00, 25,00)

--- Тестирование LinkedListTabulatedFunction ---

Создана LinkedListTabulatedFunction:

Текущий набор точек (5 точек):

Точка 0: (0,00, 0,00)
Точка 1: (1,00, 0,00)
Точка 2: (2,00, 0,00)
Точка 3: (3,00, 0,00)
Точка 4: (4,00, 0,00)

После установки значений Y = x?:

Текущий набор точек (5 точек):

Точка 0: (0,00, 0,00)
Точка 1: (1,00, 1,00)
Точка 2: (2,00, 4,00)
Точка 3: (3,00, 9,00)
Точка 4: (4,00, 16,00)

Тестирование вычисления значений (с оптимизацией):

f(0.0) = 0,00 (точное совпадение - использует существующий Y)
f(1.0) = 1,00 (точное совпадение - использует существующий Y)
f(1.2) = 1,60 (интерполяция)
f(2.0) = 4,00 (точное совпадение - использует существующий Y)
f(2.5) = 6,50 (интерполяция)
f(3.8) = 14,60 (интерполяция)

Модификация точек:

После изменения точки 3 на (3.0, 15.0):

Текущий набор точек (5 точек):

Точка 0: (0,00, 0,00)

Точка 1: (1,00, 1,00)

Точка 2: (2,00, 4,00)

Точка 3: (3,00, 15,00)

Точка 4: (4,00, 16,00)

После добавления точки (1.5, 3.0):

Текущий набор точек (6 точек):

Точка 0: (0,00, 0,00)

Точка 1: (1,00, 1,00)

Точка 2: (1,50, 3,00)

Точка 3: (2,00, 4,00)

Точка 4: (3,00, 15,00)

Точка 5: (4,00, 16,00)

--- Тестирование исключений ---

Тестирование корректных исключений:

Корректно: Левая граница должна быть меньше правой

Корректно: Количество точек должно быть не менее 2

Создана функция для теста выхода за границы:

Текущий набор точек (3 точек):

Точка 0: (0,00, 0,00)

Точка 1: (1,00, 0,00)

Точка 2: (2,00, 0,00)

Корректно: Индекс точки выходит за границы

Создана функция для теста отрицательного индекса:

Текущий набор точек (4 точек):

Точка 0: (0,00, 0,00)

Точка 1: (1,00, 0,00)

Точка 2: (2,00, 0,00)

Точка 3: (3,00, 0,00)

Корректно: Индекс выходит за границы
Создана функция для теста нарушения порядка:
Текущий набор точек (4 точек):
 Точка 0: (0,00, 0,00)
 Точка 1: (1,00, 0,00)
 Точка 2: (2,00, 0,00)
 Точка 3: (3,00, 0,00)

Корректно: Координата X должна быть больше предыдущей точки
Создана функция для теста дублирования X:
Текущий набор точек (3 точек):
 Точка 0: (0,00, 0,00)
 Точка 1: (1,00, 0,00)
 Точка 2: (2,00, 0,00)

Корректно: Точка с такой координатой X уже существует
Создана функция для теста удаления:
Текущий набор точек (3 точек):
 Точка 0: (0,00, 0,00)
 Точка 1: (1,00, 0,00)
 Точка 2: (2,00, 0,00)

После первого удаления (2 точки осталось):
Текущий набор точек (2 точек):
 Точка 0: (1,00, 0,00)
 Точка 1: (2,00, 0,00)

Корректно: Нельзя удалить точку: должно остаться минимум 2 точки
Создана функция для теста некорректного индекса:
Текущий набор точек (3 точек):
 Точка 0: (0,00, 0,00)
 Точка 1: (1,00, 0,00)
 Точка 2: (2,00, 0,00)

Корректно: Индекс точки выходит за границы

Все тесты исключений пройдены успешно!
PS C:\Users\Alina> []