

## Лабораторная работа №3.

Выполнил : Горынин Дмитрий, ст.гр. 6201-120303D.

### Задание 1.

Ознакомиться (изучить документацию) со следующими классами исключений, входящих в API Java:

java.lang.Exception, java.lang.IndexOutOfBoundsException,  
java.lang.ArrayIndexOutOfBoundsException, java.lang.IllegalArgumentException,  
java.lang.IllegalStateException

### Задание 2.

В пакете functions создать два класса исключений:

1. FunctionPointIndexOutOfBoundsException
2. InappropriateFunctionPointException

```
1 package functions;
2
3 public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException {
4     public FunctionPointIndexOutOfBoundsException() { no usages
5         super();
6     }
7
8     public FunctionPointIndexOutOfBoundsException(String message) { no usages
9         super(message);
10    }
11
12    public FunctionPointIndexOutOfBoundsException(int index) { 10 usages
13        super("Function point index out of range: " + index);
14    }
15
16    public FunctionPointIndexOutOfBoundsException(String message, Throwable cause) { no usages
17        super(message);
18        initCause(cause);
19    }
20 }
```

```
1 package functions;
2
3 public class InappropriateFunctionPointException extends Exception {
4     public InappropriateFunctionPointException() { no usages
5         super();
6     }
7
8     public InappropriateFunctionPointException(String message) { 4 usages
9         super(message);
10    }
11
12    public InappropriateFunctionPointException(String message, Throwable cause) { no usages
13        super(message, cause);
14    }
15
16 @
17     public InappropriateFunctionPointException(FunctionPoint point) { 6 usages
18         super("Inappropriate function point: (" + point.getX() + "; " + point.getY() + ")");
19     }
20 }
```

### Задание 3.

В разработанный ранее класс TabulatedFunction внести изменения, обеспечивающие выбрасывание исключений методами класса.

Для примера:

```
public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) { 5 usages
    if (leftX >= rightX || pointsCount < 2) {
        throw new IllegalArgumentException("Некорректные параметры функции");
    }
}
```

Задание 4.

В пакете functions создать класс LinkedListTabulatedFunction, объект которого также должен описывать табулированную функцию.

```
1  package functions;
2
3  public class LinkedListTabulatedFunction implements TabulatedFunction { 5 usages
4
5      private static class FunctionNode { 37 usages
6          private FunctionPoint point; 33 usages
7          private FunctionNode prev; 15 usages
8          private FunctionNode next; 23 usages
9
10         public FunctionNode(FunctionPoint point) { 1 usage
11             this.point = point;
12         }
13
14         public FunctionNode(FunctionPoint point, FunctionNode prev, FunctionNode next) { 2 usages
15             this.point = point;
16             this.prev = prev;
17             this.next = next;
18         }
19     }
20 }
```

Задание 5.

Для обеспечения второй функции класса LinkedListTabulatedFunction реализовать в классе конструкторы и методы, аналогичные конструкторам и методам класса TabulatedFunction.

Результат на скриншоте задания 4.

Задание 6.

Класс TabulatedFunction переименовать в класс ArrayTabulatedFunction.

Создать интерфейс TabulatedFunction, содержащий объявления общих методов классов ArrayTabulatedFunction и LinkedListTabulatedFunction.

```

1 package functions;
2
3 public interface TabulatedFunction {
4     // Методы для работы с функцией
5     double getLeftDomainBorder(); 5 usages 2 implementations
6     double getRightDomainBorder(); 5 usages 2 implementations
7     double getFunctionValue(double x); 6 usages 2 implementations
8
9     // Методы для работы с точками
10    int getPointsCount(); 3 usages 2 implementations
11    FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException; 1 usage 2 implementations
12    void setPoint(int index, FunctionPoint point) 2 usages 2 implementations
13        throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;
14    double getPointX(int index) throws FunctionPointIndexOutOfBoundsException; 1 usage 2 implementations
15    void setPointX(int index, double x) no usages 2 implementations
16        throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;
17    double getPointY(int index) throws FunctionPointIndexOutOfBoundsException; no usages 2 implementations
18    void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException; 4 usages 2 implementations
19
20    // Методы изменения количества точек
21    void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException; 4 usages 2 implementations
22    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException; 5 usages 2 implementations
23
24    // Вспомогательный метод для вывода
25    void printFunction(); 4 usages 2 implementations
26 }

```

### Задание 7.

Проверить работу написанных классов.

```

1 import functions.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         System.out.println("== КОМПЛЕКСНАЯ ПРОВЕРКА РАБОТЫ КЛАССОВ ==\n");
6
7         // Тестирование ArrayTabulatedFunction
8         System.out.println("1. ТЕСТИРОВАНИЕ ArrayTabulatedFunction:");
9         testImplementation(new ArrayTabulatedFunction( leftX: 0, rightX: 4, pointsCount: 5 ), implName: "ArrayTabulatedFunction");
10
11        // Тестирование LinkedListTabulatedFunction
12        System.out.println("\n\n2. ТЕСТИРОВАНИЕ LinkedListTabulatedFunction:");
13        testImplementation(new LinkedListTabulatedFunction( leftX: 0, rightX: 4, pointsCount: 5 ), implName: "LinkedListTabulatedFunction");
14
15        // Сравнительный тест
16        System.out.println("\n\n3. СРАВНИТЕЛЬНЫЙ ТЕСТ ОБЕИХ РЕАЛИЗАЦИЙ:");
17        comparativeTest();
18
19        System.out.println("\n== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ ==");
20    }
}

```

Вывод:

```
== ТЕСТИРОВАНИЕ ARRAY TABULATED FUNCTION ==
1. Тест конструкторов:
    ✓ IllegalArgumentException: leftX must be less than rightX
    ✓ IllegalArgumentException: pointsCount must be at least 2
    ✓ Корректная функция создана
2. Тест методов доступа:
    ✓ FunctionPointIndexOutOfBoundsException: Index out of bounds: -1
    ✓ FunctionPointIndexOutOfBoundsException: Index out of bounds: 10
3. Тест методов изменения:
    ОШИБКА: Не выброшено исключение для нарушения упорядоченности
    ОШИБКА: Не выброшено исключение для дублирования X
4. Тест удаления точек:
    ✓ IllegalStateException: Cannot delete point - function must have at least 2 points

== ТЕСТИРОВАНИЕ LINKED LIST TABULATED FUNCTION ==
1. Тест конструкторов:
    ✓ IllegalArgumentException: leftX must be less than rightX
    ✓ IllegalArgumentException: values array must have at least 2 elements
    ✓ Корректная функция создана
2. Тест методов доступа:
    ✓ FunctionPointIndexOutOfBoundsException: Index out of bounds: -1
    ✓ FunctionPointIndexOutOfBoundsException: Index out of bounds: 10
3. Тест методов изменения:
    ОШИБКА: Не выброшено исключение для нарушения упорядоченности
4. Тест удаления точек:
    ✓ IllegalStateException: Cannot delete point - function must have at least 2 points
5. Тест корректной работы:
    ✓ Точка добавлена корректно
    ✓ Y координата изменена корректно
    ✓ Значение функции в x=1.5: 1.7142857142857144
```

```
==== ТЕСТИРОВАНИЕ ЧЕРЕЗ ИНТЕРФЕЙС ===
```

```
Тестирование через интерфейс TabulatedFunction:
```

```
Тестирование ArrayTabulatedFunction:
```

```
Количество точек: 3
```

```
Левая граница: 0.0
```

```
Правая граница: 5.0
```

```
Точка 0: (0,0, 0,0)
```

```
Точка 1: (2,5, 0,0)
```

```
Точка 2: (5,0, 0,0)
```

```
f(2,5) = 0,00
```

```
✓ Все операции выполнены успешно
```

```
Тестирование LinkedListTabulatedFunction:
```

```
Количество точек: 3
```

```
Левая граница: 0.0
```

```
Правая граница: 5.0
```

```
Точка 0: (0,0, 1,0)
```

```
Точка 1: (2,5, 2,0)
```

```
Точка 2: (5,0, 3,0)
```

```
f(2,5) = 2,00
```

```
✓ Все операции выполнены успешно
```

```
Сравнение реализаций:
```

```
Созданы функции y = x^2 на [0, 10] с 6 точками
```

```
Проверка точек:
```

```
Точка 0: массив(0,0, 0,0) список(0,0, 0,0) совпадают=ДА
```

```
Точка 1: массив(2,0, 4,0) список(2,0, 4,0) совпадают=ДА
```

```
Точка 2: массив(4,0, 16,0) список(4,0, 16,0) совпадают=ДА
```

```
Точка 3: массив(6,0, 36,0) список(6,0, 36,0) совпадают=ДА
```

```
Точка 4: массив(8,0, 64,0) список(8,0, 64,0) совпадают=ДА
Точка 5: массив(10,0, 100,0) список(10,0, 100,0) совпадают=ДА
```

Сравнение значений функции:

```
f(0,0): массив=0,000000    список=0,000000    совпадают=ДА
f(1,0): массив=2,000000    список=2,000000    совпадают=ДА
f(2,5): массив=7,000000    список=7,000000    совпадают=ДА
f(5,0): массив=26,000000   список=26,000000   совпадают=ДА
f(7,5): массив=57,000000   список=57,000000   совпадают=ДА
f(10,0): массив=100,000000  список=100,000000  совпадают=ДА
f(12,0): массив=NaN       список=NaN        совпадают=ДА
```

Границы домена - массив: [0,0, 10,0]

Границы домена - список: [0,0, 10,0]

Границы совпадают: ДА

✓ Реализации работают идентично

```
Process finished with exit code 0
```