

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 { 34 usages
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 { 21 usages
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 @ public InappropriateFunctionPointException(FunctionPoint point) { 6 usages
17     super("Inappropriate function point: (" + point.getX() + "; " + point.getY() + ")");
18 }
19 }
```

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
11     public FunctionNode(FunctionPoint point) { 1 usage
12         this.point = point;
13     }
14
15     public FunctionNode(FunctionPoint point, FunctionNode prev, FunctionNode next) { 2 usages
16         this.point = point;
17         this.prev = prev;
18         this.next = next;
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 { 17 usages 2 implementations
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("\n2. ТЕСТИРОВАНИЕ LinkedListTabulatedFunction:");
13         testImplementation(new LinkedListTabulatedFunction( leftX: 0, rightX: 4, pointsCount: 5), implName: "LinkedListTabulatedFunction");
14
15         // Сравнительный тест
16         System.out.println("\n3. СРАВНИТЕЛЬНЫЙ ТЕСТ ОБЕИХ РЕАЛИЗАЦИЙ:");
17         comparativeTest();
18
19         System.out.println("\n=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ ===");
20     }
21 }

```

Вывод:

```

1. ТЕСТИРОВАНИЕ ArrayTabulatedFunction:

--- ArrayTabulatedFunction ---
Инициализированная функция:
Табулированная функция (массив):
(0,000000; 0,000000) (1,000000; 1,000000) (2,000000; 4,000000) (3,000000; 9,000000) (4,000000; 16,000000)

А) КОРРЕКТНЫЕ ОПЕРАЦИИ:
✓ Добавлены точки (1.5, 2.25) и (3.5, 12.25)
✓ Изменена Y-координата точки с индексом 2 на 4.0
✓ Удалена точка с индексом 1
✓ Вычисления функции:
    f(0,5) = 1,3333
    f(1,0) = 2,6667
    f(2,0) = 4,0000
    f(2,5) = 6,5000
    f(3,0) = 9,0000
Табулированная функция (массив):
(0,000000; 0,000000) (1,500000; 4,000000) (2,000000; 4,000000) (3,000000; 9,000000) (3,500000; 12,250000) (4,000000; 16,000000)

```

Б) ТЕСТИРОВАНИЕ ИСКЛЮЧЕНИЙ:

1. `FunctionPointIndexOutOfBoundsException`:

- ✓ `FunctionPointIndexOutOfBoundsException` для индекса -1: Function point index out of range: -1
- ✓ `FunctionPointIndexOutOfBoundsException` для индекса 10: Function point index out of range: 10
- ✓ `FunctionPointIndexOutOfBoundsException` для индекса 100: Function point index out of range: 100
- ✓ `FunctionPointIndexOutOfBoundsException` для индекса -5: Function point index out of range: -5
- ✓ `FunctionPointIndexOutOfBoundsException` для `setPoint`: Function point index out of range: 10

2. `InappropriateFunctionPointException`:

- ✓ `InappropriateFunctionPointException` при добавлении дубликата: Inappropriate function point: (2.0; 100.0)
- ✓ `InappropriateFunctionPointException` при нарушении порядка: Inappropriate function point: (0.5; 0.25)

3. `IllegalStateException`:

- ✓ `IllegalStateException` при удалении из минимальной функции: Нельзя удалить точку - минимальное количество точек: 2

4. `IllegalArgumentException` в конструкторах:

- ✓ `IllegalArgumentException` для неправильных границ (массив): Некорректные параметры функции
- ✓ `IllegalArgumentException` для неправильных границ (список): Некорректные параметры функции
- ✓ `IllegalArgumentException` для недостаточного количества точек (массив): Некорректные параметры функции
- ✓ `IllegalArgumentException` для недостаточного количества точек (список): Некорректные параметры функции

В) ГРАНИЧНЫЕ СЛУЧАИ:

Точки вне области определения:

- `f(-1,0000) = NaN` (корректно)
- `f(10,0000) = NaN` (корректно)
- `f(-0,0001) = NaN` (корректно)
- `f(4,0001) = NaN` (корректно)

Граничные точки области определения:

- Левая граница: `f(0,0000) = 0,0000`
- Правая граница: `f(4,0000) = 16,0000`

А) КОРРЕКТНЫЕ ОПЕРАЦИИ:

- ✓ Добавлены точки (1.5, 2.25) и (3.5, 12.25)
- ✓ Изменена Y-координата точки с индексом 2 на 4.0
- ✓ Удалена точка с индексом 1

✓ Вычисления функции:

- `f(0,5) = 1,3333`
- `f(1,0) = 2,6667`
- `f(2,0) = 4,0000`
- `f(2,5) = 6,5000`
- `f(3,0) = 9,0000`

Табулированная функция (связный список):

(0,000000; 0,000000) (1,500000; 4,000000) (2,000000; 4,000000) (3,000000; 9,000000) (3,500000; 12,250000) (4,000000; 16,000000)

Б) ТЕСТИРОВАНИЕ ИСКЛЮЧЕНИЙ:

1. `FunctionPointIndexOutOfBoundsException`:

- ✓ `FunctionPointIndexOutOfBoundsException` для индекса -1: Function point index out of range: -1
- ✓ `FunctionPointIndexOutOfBoundsException` для индекса 10: Function point index out of range: 10
- ✓ `FunctionPointIndexOutOfBoundsException` для индекса 100: Function point index out of range: 100
- ✓ `FunctionPointIndexOutOfBoundsException` для индекса -5: Function point index out of range: -5
- ✓ `FunctionPointIndexOutOfBoundsException` для `setPoint`: Function point index out of range: 10

2. `InappropriateFunctionPointException`:

- ✓ `InappropriateFunctionPointException` при добавлении дубликата: Inappropriate function point: (2.0; 100.0)
- ✓ `InappropriateFunctionPointException` при нарушении порядка: Inappropriate function point: (0.5; 0.25)

3. `IllegalStateException`:

- ✓ `IllegalStateException` при удалении из минимальной функции: Нельзя удалить точку - минимальное количество точек: 2

4. `IllegalArgumentException` в конструкторах:

- ✓ `IllegalArgumentException` для неправильных границ (массив): Некорректные параметры функции
- ✓ `IllegalArgumentException` для неправильных границ (список): Некорректные параметры функции
- ✓ `IllegalArgumentException` для недостаточного количества точек (массив): Некорректные параметры функции
- ✓ `IllegalArgumentException` для недостаточного количества точек (список): Некорректные параметры функции

В) ГРАНИЧНЫЕ СЛУЧАИ:

Точки вне области определения:

$f(-1,0000) = \text{NaN}$ (корректно)

$f(10,0000) = \text{NaN}$ (корректно)

$f(-0,0001) = \text{NaN}$ (корректно)

$f(4,0001) = \text{NaN}$ (корректно)

Граничные точки области определения:

Левая граница: $f(0,0000) = 0,0000$

Правая граница: $f(4,0000) = 16,0000$

3. СРАВНИТЕЛЬНЫЙ ТЕСТ ОБЕИХ РЕАЛИЗАЦИЙ:

СРАВНЕНИЕ ArrayTabulatedFunction И LinkedListTabulatedFunction

Базовые характеристики:

Область определения: $[0,00, 4,00]$ vs $[0,00, 4,00]$

Количество точек: 5 vs 5

Сравнение вычислений:

$f(0,00)$: массив=0,000000	список=NaN	совпадают=НЕТ
$f(0,50)$: массив=0,500000	список=NaN	совпадают=НЕТ
$f(1,00)$: массив=1,000000	список=NaN	совпадают=НЕТ
$f(1,50)$: массив=2,500000	список=NaN	совпадают=НЕТ
$f(2,00)$: массив=4,000000	список=NaN	совпадают=НЕТ
$f(2,50)$: массив=6,500000	список=NaN	совпадают=НЕТ
$f(3,00)$: массив=9,000000	список=NaN	совпадают=НЕТ
$f(3,50)$: массив=12,500000	список=NaN	совпадают=НЕТ
$f(4,00)$: массив=16,000000	список=NaN	совпадают=НЕТ
$f(2,25)$: массив=5,250000	список=NaN	совпадают=НЕТ
$f(3,75)$: массив=14,250000	список=NaN	совпадают=НЕТ

РЕЗУЛЬТАТ СРАВНЕНИЯ: ЕСТЬ РАСХОЖДЕНИЯ ×

Проверка после модификаций:

Массив после модификаций:

Табулированная функция (массив):

(0,000000; 0,000000) (1,000000; 2,000000) (2,000000; 4,000000) (3,000000; 9,000000) (4,000000; 16,000000)

Список после модификаций:

Табулированная функция (связный список):

(0,000000; 0,000000) (1,000000; 2,000000) (2,000000; 4,000000) (3,000000; 9,000000) (4,000000; 16,000000)

=== ВСЕ ТЕСТЫ ЗАВЕРШЕНЫ ===

Process finished with exit code 0