

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ ГОСУДАРСТВЕННОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени
академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №3

Студент Семенова В. Г.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Содержание

Задание 1	3
Задание 2	3
Задание 3	4
Задание 4	4
Задание 5	5
Задание 6	5
Задание 7	5

Задание 1

Ознакомьтесь 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 `function` создаем 2 класса исключений. Первый класс `FunctionPointIndexOutOfBoundsException` наследуется от `IndexOutOfBoundsException`, значит исключение не проверяемое и не требует обработки в `try-catch`. В нем создается конструктор, который вызывает конструктор родительского класса с сообщением об ошибке.

Второй класс `InappropriateFunctionPointException` наследуется от `Exception`. Это проверяемое исключение, которое требует обработки. В нем создается конструктор, который передает сообщение родительскому классу `Exception`. Сообщения создаются при выбрасывании исключений в разных методах.

Задание 3

Добавлены необходимые изменения. Их работа проверена через запуск в консоли.

```
Проверка исключений

1. Попытка получить точку с индексом 25:
FunctionPointIndexOutOfBoundsException

2. Попытка установить точку с нарушением порядка X:
InappropriateFunctionPointException - X должен быть больше чем у левой точки

3. Попытка добавить точку с существующим X (5.0):
InappropriateFunctionPointException - Точка с таким x уже существует

4. Попытка установить X точки[3] = 1.0 (нарушение порядка):
InappropriateFunctionPointException - X должен быть больше чем у левой точки

5. Попытка удалить точку при недостаточном количестве:
IllegalStateException - Невозможно удалить точку: в наборе менее 3 точек

6. Попытка создать функцию с leftX >= rightX:
IllegalArgumentException - Левая граница (10.0) должна быть меньше правой границы (5.0)

7. Конструктор с pointsCount < 2:
IllegalArgumentException - Количество точек (1) должно быть не менее 2

8. setPointY() с индексом 100:
FunctionPointIndexOutOfBoundsException - Выход за границы индекса указанной точки

9. getPointX() с индексом -5:
FunctionPointIndexOutOfBoundsException - Выход за границы индекса указанной точки

9. getPointX() с индексом 57:
FunctionPointIndexOutOfBoundsException - Выход за границы индекса указанной точки
```

Задание 4

Создали класс `LinkedListTabulatedFunction`, в нем описали приватный статический класс `FunctionNode`, поля которого объявлены как `public` для удобства доступа внутри внешнего класса. Он содержит конструктор, возвращающий копию точки. Основной класс содержит 4 поля типа `private`: `head`- голова списка, `pointsCount`- количество точек, `lastEl`, `nextEl`- переменные для хранения следующего и предыдущий элементы.

Метод `FunctionNode getNodeByIndex(int index)` оптимизировали, добавив проверки, на сравнение переданного индекса с проверенным ранее на точное совпадение или отличие на единицу. Значение проверенного индекса сохраняем в `lastIndex`. Этот метод типа `private`, так как он возвращает внутреннюю структуру (`FunctionNode`), которую пользователь не должен видеть.

Реализовали метод `FunctionNode addNodeToTail()`, добавляющий новый узел в конец. Настраиваем связи для него так, чтобы предыдущим элементом для него стал старый хвост, а следующим голова, так как список циклический. Аналогично обновляем хвост и голову. Метод создает пустой узел, поэтому он скрыт от пользователя модификатором `private`

Метод `FunctionNode addNodeByIndex(int index)` сначала проверяет, что индекс не выходит за границы, и что он не равен числу точек, если так, то элемент ставится в конец. Создаем новую нулевую точку и вспомогательную переменную, в которую поместим точку с переданным индексом(`targetNode`)

```
FunctionNode newNode= new FunctionNode(new FunctionPoint(0, 0));
```

```
FunctionNode targetNode= getNodeByIndex(index);
```

Настраиваем связи: предыдущий элемент для нового-предыдущий для старого. Следующий для нового-старый. Предыдущий элемент для старого-новый, следующий для старого остается таким же. Метод определен как `private`, так как вставляет узел, что должно быть не доступно пользователю.

В методе `deleteNodeByIndex()` предыдущий узел (`prevNode`) указывает на следующий узел (`nextNode`) вместо удаляемого, следующий узел (`nextNode`) теперь указывает на предыдущий узел (`prevNode`) вместо удаляемого. Метод скрыт от пользователя, так как тот должен удалять точки через `deletePoint()`.

Задание 5

Реализовали конструкторы аналогичные `TabulatedFunction`, добавили те же исключения. Все методы имеют те же сигнатуры и выбрасывают те же исключения.

Оптимизировали методы `getLeftDomainBorder()` и `getRightDomainBorder()` с помощью прямого доступа к узлам, использовали кеширование `lastNode` и `lastIndex` для ускорения последовательных операций.

Задание 6

Создали интерфейс `TabulatedFunction.java`, в котором объявили методы без реализации. Добавили `implements TabulatedFunction` в объявлении классов, который нужен для проверки, что мы реализуем все методы интерфейса.

Задание 7

Объявляем переменную как интерфейс:

TabulatedFunction parabola;

Создаем объект с возможностью выбора конкретного класса (`LinkedListTabulatedFunction` или `ArrayTabulatedFunction`).

```
if (args.length > 0 && args[0].equalsIgnoreCase("linked")) {  
    parabola = new LinkedListTabulatedFunction(0.0, 10.0, 11);  
    System.out.println("=== Используется LinkedListTabulatedFunction  
===");  
} else {  
    parabola = new ArrayTabulatedFunction(0.0, 10.0, 11);  
    System.out.println("=== Используется ArrayTabulatedFunction ===");  
}
```

Программа корректно работает при создании объектов обоих классов.

Используя LinkedListTabulatedFunction:

```
C:\Users\леново\OneDrive\Рабочий стол\oon\Lab-3-2025>java Main Linked
=== Используется LinkedListTabulatedFunction ===
Табулированная функция y = x^2:
Область определения: от 0.0 до 10.0
Количество точек: 11
Точки функции:
(0.0; 0.0)
(1.0; 1.0)
(2.0; 4.0)
(3.0; 9.0)
(4.0; 16.0)
(5.0; 25.0)
(6.0; 36.0)
(7.0; 49.0)
(8.0; 64.0)
(9.0; 81.0)
(10.0; 100.0)

getPoint()

Получение точки с индексом 5:
Точка[5] = (5.0; 25.0)

Проверка исключений

1. Попытка получить точку с индексом 25:
FunctionPointIndexOutOfBoundsException

2. Попытка установить точку с нарушением порядка X:
InappropriateFunctionPointException - X должен быть больше чем у левой точки

3. Попытка добавить точку с существующим X (5.0):
InappropriateFunctionPointException - Точка с таким x уже существует

4. Попытка установить X точки[3] = 1.0 (нарушение порядка);
```

Используя ArrayTabulatedFunction:

```
C:\Users\леново\OneDrive\Рабочий стол\oon\Lab-3-2025>java Main
=== Используется ArrayTabulatedFunction ===
Табулированная функция y = x^2:
Область определения: от 0.0 до 10.0
Количество точек: 11
Точки функции:
(0.0; 0.0)
(1.0; 1.0)
(2.0; 4.0)
(3.0; 9.0)
(4.0; 16.0)
(5.0; 25.0)
(6.0; 36.0)
(7.0; 49.0)
(8.0; 64.0)
(9.0; 81.0)
(10.0; 100.0)

getPoint()

Получение точки с индексом 5:
Точка[5] = (5.0; 25.0)

Проверка исключений

1. Попытка получить точку с индексом 25:
FunctionPointIndexOutOfBoundsException

2. Попытка установить точку с нарушением порядка X:
InappropriateFunctionPointException - X должен быть больше чем у левой точки

3. Попытка добавить точку с существующим X (5.0);
```

