

## **Лабораторная работа №7**

Выполнил: Мурзяков Егор 6204 – 010302D

## Ход выполнения задания № 1

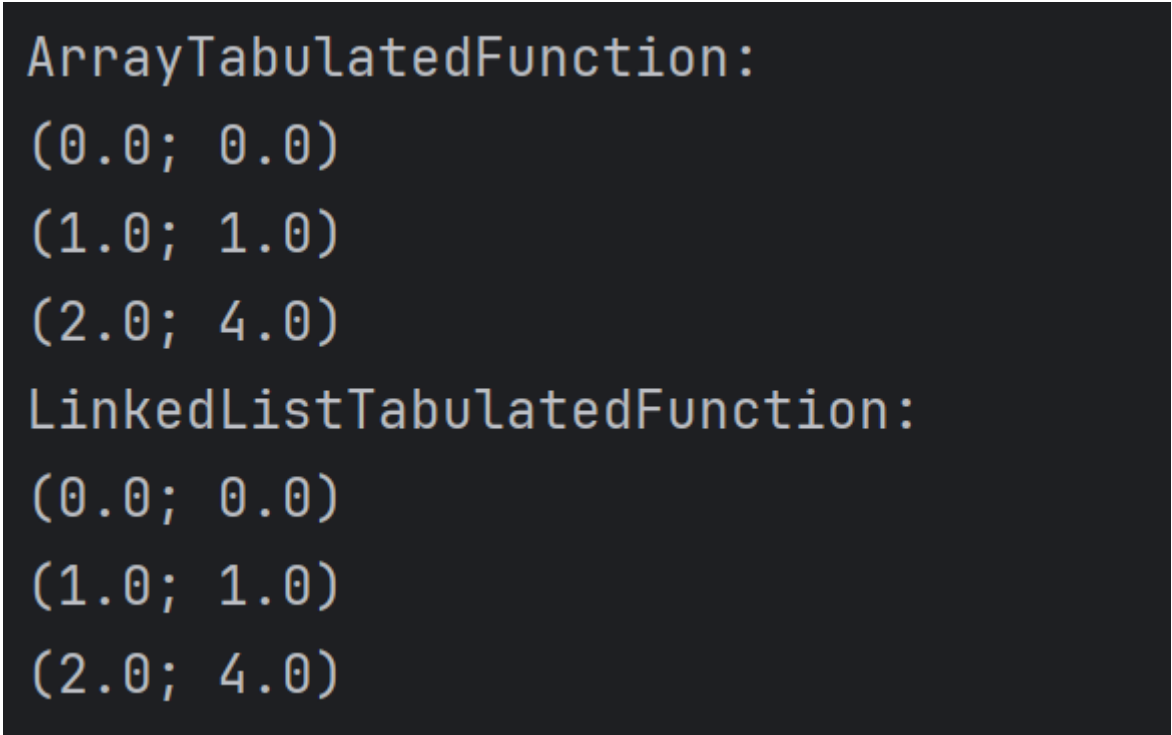
Следуя инструкциям в интерфейс `TabulatedFunction` был добавлен родительский тип `Iterable<FunctionPoint>`, что позволяет использовать объекты табулированных функций в улучшенном цикле `for-each`.

В классах `ArrayTabulatedFunction` и `LinkedListTabulatedFunction`, реализующих интерфейс `TabulatedFunction`, был реализован метод `iterator()`, возвращающий объект итератора.

Так как метод `remove()` в итераторах не реализован, он всегда выбрасывает исключение `UnsupportedOperationException`.

Метод `next()` выбрасывает исключение `NoSuchElementException`, если следующего элемента нет.

Результат проверки работы итераторов (рис. 1).



```
ArrayTabulatedFunction:
(0.0; 0.0)
(1.0; 1.0)
(2.0; 4.0)
LinkedListTabulatedFunction:
(0.0; 0.0)
(1.0; 1.0)
(2.0; 4.0)
```

Рис. 1

## Ход выполнения задания № 2

В пакете `functions` был создан интерфейс `TabulatedFunctionFactory`, который объявляет три перегруженных метода `createTabulatedFunction()`. Параметры этих методов соответствуют параметрам конструкторов классов табулированных функций.

Для удобства в классах `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` были созданы вложенные публичные классы фабрик. Эти классы реализуют интерфейс `TabulatedFunctionFactory` и порождают объекты соответствующих классов табулированных функций.

В классе `TabulatedFunctions` было объявлено приватное статическое поле `factory` типа `TabulatedFunctionFactory`, которое инициализируется объектом `ArrayTabulatedFunctionFactory` по умолчанию. Также был объявлен метод `setTabulatedFunctionFactory()`, позволяющий заменить объект фабрики во время выполнения программы.

В классе `TabulatedFunctions` были описаны три перегруженных метода `createTabulatedFunction()`, которые возвращают объекты табулированных функций, созданные с помощью текущей фабрики. Параметры этих методов соответствуют параметрам методов фабрики.

В методе `tabulate` класса `TabulatedFunctions`, было заменено явное создание объектов на вызов метода `createTabulatedFunction()`.

Для проверки работы фабрик был добавлен тестовый код, результат работы которого (рис. 2).

```
----- Задание 2 -----  
Фабрика по умолчанию: class functions.ArrayTabulatedFunction  
LinkedList фабрика: class functions.LinkedListTabulatedFunction  
Array фабрика: class functions.ArrayTabulatedFunction
```

Рис. 2

### Ход выполнения задания № 3

В классе `TabulatedFunctions` были добавлены три перегруженных версии метода `createTabulatedFunction`, использующих механизм рефлексии. Эти методы принимают в качестве параметра ссылку типа `Class` на описание класса, объект которого требуется создать, а также параметры, соответствующие конструкторам классов табулированных функций.

Были реализованы следующие методы:

- `createTabulatedFunction(Class functionClass, FunctionPoint[] points)`
- `createTabulatedFunction(Class functionClass, double leftX, double rightX, int pointsCount)`
- `createTabulatedFunction(Class functionClass, double leftX, double rightX, double[] values)`

Каждый из этих методов сначала проверяет, что переданный класс реализует интерфейс `TabulatedFunction`. Если проверка не проходит, выбрасывается исключение `IllegalArgumentException`.

Далее с помощью рефлексии находится конструктор с соответствующими типами параметров. Для этого используется метод `getConstructor` класса `Class`, в который передаются типы параметров конструктора. Если конструктор не найден, выбрасывается исключение `NoSuchMethodException`, которое перехватывается и оборачивается в `IllegalArgumentException`.

После успешного нахождения конструктора создается объект табулированной функции с помощью метода `newInstance`. Все исключения, возникающие в процессе рефлексии, перехватываются и преобразуются в `IllegalArgumentException` с сохранением информации о первоначальной ошибке.

Также был перегружен метод `tabulate` с использованием рефлексии и методы `inputTabulatedFunction` и `readTabulatedFunction`. Для проверки работы был использован код из задания, результат (рис. 3).

```

----- Задание 3 -----
class functions.ArrayTabulatedFunction
{(0.0; 0.0), (5.0; 0.0), (10.0; 0.0)}
class functions.ArrayTabulatedFunction
{(0.0; 0.0), (10.0; 10.0)}
class functions.LinkedListTabulatedFunction
{(0.0; 0.0), (10.0; 10.0)}
class functions.LinkedListTabulatedFunction
(0.0; 0.0)
(0.3141592653589793; 0.3090169943749474)
(0.6283185307179586; 0.5877852522924731)
(0.9424777960769379; 0.8090169943749475)
(1.2566370614359172; 0.9510565162951535)
(1.5707963267948966; 1.0)
(1.8849555921538759; 0.9510565162951536)
(2.199114857512855; 0.8090169943749475)
(2.5132741228718345; 0.5877852522924732)
(2.827433388230814; 0.3090169943749475)
(3.141592653589793; 1.2246467991473532E-16)

```

Рис 3