

# **Лабораторная работа №4**

Автор: Терин Ярослав

Группа: 6203-010302D

## Задание 1

В этом задании мы добавили в классы `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` конструкторы, получающие сразу все точки функции в виде массива объектов типа `FunctionPoint`. Если точек задано меньше двух, или если точки в массиве не упорядочены по значению абсциссы, конструкторы выбрасывают исключение `IllegalArgumentException`. Чтобы точки были упорядочены по значению абсциссы, мы каждый раз сравниваем значение текущей точки со значением предыдущей. В массив мы добавляем копию точки, чтобы не нарушить инкапсуляцию.

## Задание 2

В этом задании мы создали в пакете `functions` интерфейс `Function`, описывающий функции одной переменной и содержащий следующие методы: `getLeftDomainBorder`, `getRightDomainBorder`, `getFunctionValue`. Затем мы исключили соответствующие методы из интерфейса `TabulatedFunction` и с помощью ключевого слова `extends` расширили интерфейс `Function`.

## Задание 3

В этом задании мы создали пакет `functions.basic`. В этом пакете находится 5 публичных классов функций, заданных аналитически, и абстрактный класс `TrigonometricFunction`, реализующий интерфейс `Function` и описывающий методы получения границ области определения для тригонометрических функций. Классы `Sin`, `Cos` и `Tan` наследуют от этого класса и реализуют метод `getFunctionValue` каждый для своего класса. В пакете есть классы `Exp`, объекты которого вычисляют значение экспоненты, и `Log`, объекты которого вычисляют значение логарифма по заданному основанию. Так как метод `Math.log` вычисляет логарифм по натуральному основанию, в методе `getFunctionValue` мы воспользовались формулой для смены основания логарифма. Также все эти классы реализуют интерфейс `Function`.

## Задание 4

В этом задании мы создали пакет `functions.meta`. В этом пакете 6 публичных классов, реализующих интерфейс `Function` и позволяющий комбинировать функции. В классе `Sum` мы создали 2 приватных поля, хранящих объекты каждой функции, в конструкторе получаем ссылки типа `Function` на объекты суммируемых функций.левой границей области определения является наибольшая левая граница одной из функций, а правой границей – наименьшая. В методе `getFunctionValue` мы делаем проверку на то, принадлежит ли абсцисса области определения, если нет - возвращаем значение `Double.NaN`. Класс `Mult` реализован аналогично, только вместо суммы в методе `getFunctionValue` мы возвращаем произведение значений функций в этой точке. Класс `Power` имеет два приватных поля, хранящих функцию и степень, в которую нужно ее возвести. С помощью метода `Math.pow` мы возвращаем значение функции в точке, возведенное в степень. В конструктор класса `Scale` мы передаем функцию и коэффициенты для масштабирования по оси  $x$  и  $y$ . В области определения мы умножаем левую и правую границу на коэффициент масштабирования по оси абсцисс, если коэффициент отрицательный левая и правая граница меняются местами. Возвращаем новое значение функции. Аналогично класс `Shift`, только его конструктор в качестве параметров принимает значения для сдвига по оси координат, область определения сдвигается на значение сдвига по оси абсцисс, а затем возвращается новое значение функции. Далее класс `Composition` хранит два приватных поля с 2 функциями. Область определения берем у второй функции, то есть у той, которая внутренняя. Возвращаем значение первой функции от значения, которое принимает вторая функция в точке  $x$ .

## Задание 5

В пакете `functions` создан класс `Functions` с приватным конструктором, что предотвращает создание объектов этого класса. Класс содержит статические методы фабрики, которые далее по порядку описаны: `shift` возвращает функцию, полученную сдвигом исходной, `scale` возвращает функцию, полученную масштабированием, `power` возвращает функцию, являющуюся степенью исходной, `sum` возвращает сумму двух функций, `mult` возвращает произведение двух функций, `composition` возвращает композицию двух функций.

## Задание 6

В этом задании мы создали в пакете `functions` класс `TabulatedFunctions`, в котором есть приватный конструктор, чтобы нельзя было создать объекты этого класса, и метод `tabulate`. Получающий функцию и возвращающий её табулированный аналог на заданном отрезке с заданным количеством точек. В методе мы создаем массив для хранения значений функции в каждой точке разбиения, затем создаем объект, а потом в конструктор передаем значения, и возвращаем новую табулированную функцию.

## Задание 7

В этом задании мы добавили в класс `TabulatedFunctions` методы для работы с потоками ввода-вывода. Для байтовых потоков мы реализовали метод `outputTabulatedFunction()`, который записывает табулированную функцию в байтовый поток с использованием `DataOutputStream`. В этом методе мы сначала записываем количество точек функции, а затем последовательно записываем пары координат `X` и `Y` для каждой точки. Также для байтовых потоков мы создали метод `inputTabulatedFunction`, который читает табулированную функцию из байтового потока с помощью `DataInputStream`. В этом методе мы сначала читаем количество точек, затем читаем пары координат и создаем массив точек, на основе которого строим новую табулированную функцию. Для символьных потоков мы реализовали метод `writeTabulatedFunction`, который записывает табулированную функцию в символьный поток через `PrintWriter`. Мы записываем данные в текстовом формате: сначала количество точек, затем на каждой строке пару координат `X` и `Y`, разделенных пробелом. Также для символьных потоков мы создали метод `readTabulatedFunction`, который читает табулированную функцию из символьного потока с использованием `StreamTokenizer`. Мы читаем количество точек, затем последовательно читаем пары координат `X` и `Y`. Все методы объявлены с `throws IOException`, поэтому возникающие исключения перебрасываются на уровень выше для обработки.

# Задание 8

В этом задании мы проверили работу всех созданных классов. Мы создали объекты Sin и Cos и вывели их значения на отрезке 0, до пи. Затем создали табулированные аналоги этих функций с 10 точками и сравнили значения. С помощью класса Functions создали функцию суммы квадратов табулированных синуса и косинуса. Исследовали влияние количества точек табулирования на точность результирующей функции - при увеличении точек до 100 точность повысилась. Результат показан на фото.

```
Sin на отрезке (x = 0.0) = 0.0
Cos на отрезке (x = 0.0) = 1.0
Sin на отрезке (x = 0.1) = 0.09983341664682815
Cos на отрезке (x = 0.1) = 0.9950041652780258
Sin на отрезке (x = 0.2) = 0.19866933079506122
Cos на отрезке (x = 0.2) = 0.9800665778412416
Sin на отрезке (x = 0.30000000000000004) = 0.2955202066613396
Cos на отрезке (x = 0.30000000000000004) = 0.955336489125606
Sin на отрезке (x = 0.4) = 0.3894183423086505
Cos на отрезке (x = 0.4) = 0.9210609940028851
Sin на отрезке (x = 0.5) = 0.479425538604203
Cos на отрезке (x = 0.5) = 0.8775825618903728
Sin на отрезке (x = 0.6) = 0.5646424733950354
Cos на отрезке (x = 0.6) = 0.8253356149096783
Sin на отрезке (x = 0.7) = 0.644217687237691
Cos на отрезке (x = 0.7) = 0.7648421872844885
Sin на отрезке (x = 0.7999999999999999) = 0.7173560908995227
Cos на отрезке (x = 0.7999999999999999) = 0.6967067093471655
Sin на отрезке (x = 0.8999999999999999) = 0.7833269096274833
Cos на отрезке (x = 0.8999999999999999) = 0.621609682706645
Sin на отрезке (x = 0.9999999999999999) = 0.8414709848078964
Cos на отрезке (x = 0.9999999999999999) = 0.5403023058681398
Sin на отрезке (x = 1.0999999999999999) = 0.8912073600614353
Cos на отрезке (x = 1.0999999999999999) = 0.4535961214255775
Sin на отрезке (x = 1.2) = 0.9320390859672263
Cos на отрезке (x = 1.2) = 0.3623577544766736
```

Сумма квадратов синуса и косинуса(10 точек разбиения)

```
Сумма на отрезке (x = 0.0) = 1.0
Сумма на отрезке (x = 0.1) = 1.0
Сумма на отрезке (x = 0.2) = 1.0
Сумма на отрезке (x = 0.30000000000000004) = 1.0
Сумма на отрезке (x = 0.4) = 1.0
Сумма на отрезке (x = 0.5) = 1.0
Сумма на отрезке (x = 0.6) = 1.0
Сумма на отрезке (x = 0.7) = 0.9999999999999999
Сумма на отрезке (x = 0.7999999999999999) = 0.9999999999999999
Сумма на отрезке (x = 0.8999999999999999) = 0.9999999999999999
Сумма на отрезке (x = 0.9999999999999999) = 0.9999999999999999
Сумма на отрезке (x = 1.0999999999999999) = 1.0
Сумма на отрезке (x = 1.2) = 1.0
Сумма на отрезке (x = 1.3) = 1.0
Сумма на отрезке (x = 1.4000000000000001) = 0.9999999999999999
Сумма на отрезке (x = 1.5000000000000002) = 0.9999999999999999
Сумма на отрезке (x = 1.6000000000000003) = 0.9999999999999999
Сумма на отрезке (x = 1.7000000000000004) = 0.9999999999999999
Сумма на отрезке (x = 1.8000000000000005) = 0.9999999999999999
Сумма на отрезке (x = 1.9000000000000006) = 0.9999999999999999
Сумма на отрезке (x = 2.0000000000000004) = 0.9999999999999999
Сумма на отрезке (x = 2.1000000000000005) = 0.9999999999999999
Сумма на отрезке (x = 2.2000000000000006) = 0.9999999999999999
Сумма на отрезке (x = 2.3000000000000007) = 0.9999999999999999
Сумма на отрезке (x = 2.4000000000000001) = 0.9999999999999999
```

Табулированные функции

```
Sin на отрезке (x = 0.0) = 0.0
Cos на отрезке (x = 0.0) = 1.0
Sin на отрезке (x = 0.1) = 0.0
Cos на отрезке (x = 0.1) = 1.0
Sin на отрезке (x = 0.2) = 0.0
Cos на отрезке (x = 0.2) = 1.0
Sin на отрезке (x = 0.30000000000000004) = 0.0
Cos на отрезке (x = 0.30000000000000004) = 1.0
Sin на отрезке (x = 0.4) = 0.3420201433256687
Cos на отрезке (x = 0.4) = 0.9396926207859084
Sin на отрезке (x = 0.5) = 0.3420201433256687
Cos на отрезке (x = 0.5) = 0.9396926207859084
Sin на отрезке (x = 0.6) = 0.3420201433256687
Cos на отрезке (x = 0.6) = 0.9396926207859084
Sin на отрезке (x = 0.7) = 0.6427876096865393
Cos на отрезке (x = 0.7) = 0.766044443118978
Sin на отрезке (x = 0.7999999999999999) = 0.6427876096865393
Cos на отрезке (x = 0.7999999999999999) = 0.766044443118978
Sin на отрезке (x = 0.8999999999999999) = 0.6427876096865393
Cos на отрезке (x = 0.8999999999999999) = 0.766044443118978
Sin на отрезке (x = 0.9999999999999999) = 0.6427876096865393
Cos на отрезке (x = 0.9999999999999999) = 0.766044443118978
Sin на отрезке (x = 1.0999999999999999) = 0.8660254037844386
Cos на отрезке (x = 1.0999999999999999) = 0.5000000000000001
Sin на отрезке (x = 1.2) = 0.8660254037844386
Cos на отрезке (x = 1.2) = 0.5000000000000001
```

```
Сумма на отрезке (x = 0.0) = 1.0
Сумма на отрезке (x = 0.1) = 1.0
Сумма на отрезке (x = 0.2) = 1.0
Сумма на отрезке (x = 0.30000000000000004) = 0.9999999999999999
Сумма на отрезке (x = 0.4) = 1.0
Сумма на отрезке (x = 0.5) = 1.0
Сумма на отрезке (x = 0.6) = 1.0
Сумма на отрезке (x = 0.7) = 0.9999999999999999
Сумма на отрезке (x = 0.7999999999999999) = 1.0
Сумма на отрезке (x = 0.8999999999999999) = 1.0
Сумма на отрезке (x = 0.9999999999999999) = 1.0
Сумма на отрезке (x = 1.0999999999999999) = 1.0
Сумма на отрезке (x = 1.2) = 1.0
Сумма на отрезке (x = 1.3) = 1.0
Сумма на отрезке (x = 1.4000000000000001) = 0.9999999999999999
Сумма на отрезке (x = 1.5000000000000002) = 1.0
Сумма на отрезке (x = 1.6000000000000003) = 1.0
Сумма на отрезке (x = 1.7000000000000004) = 0.9999999999999999
Сумма на отрезке (x = 1.8000000000000005) = 1.0
Сумма на отрезке (x = 1.9000000000000006) = 1.0
Сумма на отрезке (x = 2.0000000000000004) = 1.0
Сумма на отрезке (x = 2.1000000000000005) = 0.9999999999999999
Сумма на отрезке (x = 2.2000000000000006) = 1.0
Сумма на отрезке (x = 2.3000000000000007) = 0.9999999999999999
Сумма на отрезке (x = 2.4000000000000001) = 1.0
```

#### Экспонента

Отрезок (x = 0) ex1 = 1.0 ex2 = 1.0  
Отрезок (x = 1) ex1 = 2.718281828459045 ex2 = 2.718281828459045  
Отрезок (x = 2) ex1 = 7.38905609893065 ex2 = 7.38905609893065  
Отрезок (x = 3) ex1 = 20.085536923187668 ex2 = 20.085536923187668  
Отрезок (x = 4) ex1 = 54.598150033144236 ex2 = 54.59815003314424  
Отрезок (x = 5) ex1 = 148.4131591025766 ex2 = 148.4131591025766  
Отрезок (x = 6) ex1 = 403.4287934927351 ex2 = 403.4287934927351  
Отрезок (x = 7) ex1 = 1096.6331584284585 ex2 = 1096.6331584284585  
Отрезок (x = 8) ex1 = 2980.9579870417283 ex2 = 2980.9579870417283  
Отрезок (x = 9) ex1 = 8103.083927575384 ex2 = 8103.083927575384  
Отрезок (x = 10) ex1 = 22026.465794806718 ex2 = 22026.46579480672

#### Логарифм

Отрезок (x = 0) log1 = NaN log2 = NaN  
Отрезок (x = 1) log1 = 0.0 log2 = 0.0  
Отрезок (x = 2) log1 = 0.6931471805599453 log2 = 0.6931471805599453  
Отрезок (x = 3) log1 = 1.0986122886681098 log2 = 1.0986122886681098  
Отрезок (x = 4) log1 = 1.3862943611198906 log2 = 1.3862943611198906  
Отрезок (x = 5) log1 = 1.6094379124341003 log2 = 1.6094379124341003  
Отрезок (x = 6) log1 = 1.791759469228055 log2 = 1.791759469228055  
Отрезок (x = 7) log1 = 1.9459101490553132 log2 = 1.9459101490553132  
Отрезок (x = 8) log1 = 2.0794415416798357 log2 = 2.0794415416798357  
Отрезок (x = 9) log1 = 2.1972245773362196 log2 = 2.1972245773362196  
Отрезок (x = 10) log1 = 2.302585092994046 log2 = 2.302585092994046

## Задание 9

В этом задании мы реализовали сериализацию табулированных функций. Создали табулированный аналог композиции  $\ln(\exp(x)) = x$  на отрезке  $[0, 10]$  с 11 точками. Сериализовали объект в файл `function.ser` и десериализовали его обратно. Сравнение показало полное совпадение значений исходной и восстановленной функций. Результат показан на фото.

### Сериализация

```
Отезок (x = 0) Fun = 0.0 Fun1 = 0.0
Отезок (x = 1) Fun = 1.0 Fun1 = 1.0
Отезок (x = 2) Fun = 2.0 Fun1 = 2.0
Отезок (x = 3) Fun = 3.0000000000000004 Fun1 = 3.0000000000000004
Отезок (x = 4) Fun = 4.0 Fun1 = 4.0
Отезок (x = 5) Fun = 4.999999999999999 Fun1 = 4.999999999999999
Отезок (x = 6) Fun = 6.0 Fun1 = 6.0
Отезок (x = 7) Fun = 6.999999999999999 Fun1 = 6.999999999999999
Отезок (x = 8) Fun = 7.999999999999998 Fun1 = 7.999999999999998
Отезок (x = 9) Fun = 9.000000000000002 Fun1 = 9.000000000000002
Отезок (x = 10) Fun = 10.000000000000002 Fun1 = 10.000000000000002
```

### Сериализация через Serializable

```
Отезок (x = 0) FunSerial = 0.0 Deserialized = 0.0
Отезок (x = 1) FunSerial = 1.0 Deserialized = 1.0
Отезок (x = 2) FunSerial = 2.0 Deserialized = 2.0
Отезок (x = 3) FunSerial = 3.0000000000000004 Deserialized = 3.0000000000000004
Отезок (x = 4) FunSerial = 4.0 Deserialized = 4.0
Отезок (x = 5) FunSerial = 4.999999999999999 Deserialized = 4.999999999999999
Отезок (x = 6) FunSerial = 6.0 Deserialized = 6.0
Отезок (x = 7) FunSerial = 6.999999999999999 Deserialized = 6.999999999999999
Отезок (x = 8) FunSerial = 7.999999999999998 Deserialized = 7.999999999999998
Отезок (x = 9) FunSerial = 9.000000000000002 Deserialized = 9.000000000000002
Отезок (x = 10) FunSerial = 10.000000000000002 Deserialized = 10.000000000000002
```

### Сериализация LinkedListTabulatedFunction

```
LinkedList (x = 0) = 0.0 Deserialized = 0.0
LinkedList (x = 1) = 1.0 Deserialized = 1.0
LinkedList (x = 2) = 2.0 Deserialized = 2.0
LinkedList (x = 3) = 3.0000000000000004 Deserialized = 3.0000000000000004
LinkedList (x = 4) = 4.0 Deserialized = 4.0
LinkedList (x = 5) = 4.999999999999999 Deserialized = 4.999999999999999
LinkedList (x = 6) = 6.0 Deserialized = 6.0
LinkedList (x = 7) = 6.999999999999999 Deserialized = 6.999999999999999
LinkedList (x = 8) = 7.999999999999998 Deserialized = 7.999999999999998
LinkedList (x = 9) = 9.000000000000002 Deserialized = 9.000000000000002
LinkedList (x = 10) = 10.000000000000002 Deserialized = 10.000000000000002
```