Отчёт по лабораторной работе №4

Студент: Прокопенко А. П.

Группа: 6201-120303D

Задание 1. Конструкторы для табулированных функций

- В классах ArrayTabulatedFunction и LinkedListTabulatedFunction были добавлены конструкторы, принимающие массив объектов FunctionPoint[]. В конструкторах реализована проверка корректности входных данных. В случае несоответствия данных этим требованиям конструктор выбрасывает исключение IllegalArgumentException. Для обеспечения инкапсуляции внутреннее представление функции (массив или список) инициализируется копией данных из переданного массива.

Задание 2. Интерфейс Function

- В пакете functions был создан новый интерфейс Function, который описывает общее поведение для всех функций одной переменной. Он содержит три метода:

1) double getLeftDomainBorder() – возвращает левую границу области определения.

2) double getRightDomainBorder() – возвращает правую границу области определения.

3) double getFunctionValue(double x) – вычисляет значение функции в точке x.

Интерфейс TabulatedFunction был изменён: из него удалены методы, дублирующие Function, и добавлено наследование (public interface TabulatedFunction extends Function).

Задание 3. Базовые аналитические функции

- В пакете functions.basic были созданы классы, реализующие интерфейс Function:

1) Exp: вычисляет экспоненту с помощью Math.exp().

2) Log: вычисляет логарифм по основанию, заданному в конструкторе, с помощью Math.log().

3) TrigonometricFunction: абстрактный базовый класс для тригонометрических функций.

4) Sin, Cos, Tan: наследуются от TrigonometricFunction и реализуют вычисление значения через Math.sin(), Math.cos() и Math.tan() соответственно.

Задание 4. Метаклассы для комбинирования функций

- В пакете functions.meta созданы классы-обёртки, реализующие интерфейс Function и представляющие собой комбинацию других функций:

1) Sum(Function f1, Function f2): сумма двух функций.

2) Mult(Function f1, Function f2): произведение двух функций.

3) Power(Function f, double power): возведение функции в степень.

4) Scale(Function f, double scaleX, double scaleY): масштабирование функции вдоль осей.

5) Shift(Function f, double shiftX, double shiftY): сдвиг функции вдоль осей.

6) Composition(Function f1, Function f2): композиция функций f1(f2(x)).

Задание 5. Вспомогательный класс Functions

- В пакете functions создан утилитарный класс Functions с приватным конструктором, чтобы запретить создание его экземпляров. Класс содержит статические методы-фабрики для удобного создания объектов из functions.meta:

1) shift(Function f, double shiftX, double shiftY)

2) scale(Function f, double scaleX, double scaleY)

3) power(Function f, double power)

4) sum(Function f1, Function f2)

5) mult(Function f1, Function f2)

6) composition(Function f1, Function f2)

Задания 6–7. Класс TabulatedFunctions и потоки ввода-вывода

- В пакете functions создан утилитарный класс TabulatedFunctions:

1) Метод tabulate: создаёт табулированную функцию на основе аналитической, проверяя, что заданный отрезок [leftX, rightX] не выходит за пределы области определения исходной функции.

2) Методы для бинарного ввода-вывода: outputTabulatedFunction и inputTabulatedFunction используют DataOutputStream и DataInputStream для записи и чтения данных в компактном бинарном формате.

3) Методы для символьного ввода-вывода: writeTabulatedFunction и readTabulatedFunction используют PrintWriter и StreamTokenizer для работы с текстовым представлением данных, удобным для чтения человеком.

- Обоснование по потокам и исключениям:

1) Закрытие потоков: Потоки не закрываются внутри методов, так как они передаются извне. Это стандартная практика, позволяющая вызывающему коду управлять жизненным циклом потока (например, использовать try-with-resources).

2) Обработка IOException: Исключение IOException не обрабатывается внутри методов, а пробрасывается выше (throws IOException). Это перекладывает ответственность за обработку ошибок ввода-вывода на вызывающий код, что является правильным проектным решением.

Задание 8. Тестирование классов

- Созданы объекты Sin и Cos, их значения выведены на консоль для

отрезка [0, π].

- С помощью TabulatedFunctions.tabulate созданы их табулированные версии. Значения сравнились с аналитическими, что показало наличие погрешности интерполяции.

- С помощью методов класса Functions была создана функция $f(x) = \sin^2(x) + \cos^2(x)$. Её значения на отрезке [0, π] были выведены на консоль и оказались близки к 1.0, что подтвердило корректность работы.

- Проведено сохранение табулированной экспоненты в бинарный файл и логарифма в текстовый файл. Затем функции были успешно восстановлены из этих файлов, и их значения совпали с исходными.

```
=== ТЕСТИРОВАНИЕ SIN И COS ===
Сравнение Sin и Cos на отрезке [0, π]:
x       Sin(x)      Cos(x)
----------------------------------------
0,000       0,000000    1,000000
0,100       0,099833    0,995004
0,200       0,198669    0,980067
0,300       0,295520    0,955336
0,400       0,389418    0,921061
0,500       0,479426    0,877583
0,600       0,564642    0,825336
0,700       0,644218    0,764842
0,800       0,717356    0,696707
0,900       0,783327    0,621610
1,000       0,841471    0,540302
1,100       0,891207    0,453596
1,200       0,932039    0,362358
1,300       0,963558    0,267499
1,400       0,985450    0,169967
1,500       0,997495    0,070737
1,600       0,999574    -0,029200
1,700       0,991665    -0,128844
1,800       0,973848    -0,227202
1,900       0,946300    -0,323290
2,000       0,909297    -0,416147
2,100       0,863209    -0,504846
2,200       0,808496    -0,588501
2,300       0,745705    -0,666276
2,400       0,675463    -0,737394
2,500       0,598472    -0,801144
2,600       0,515501    -0,856889
2,700       0,427380    -0,904072
2,800       0,334988    -0,942222
2,900       0,239249    -0,970958
3,000       0,141120    -0,989992
3,100       0,041581    -0,999135
```

```
=== ТЕСТИРОВАНИЕ ТАБУЛИРОВАНИЯ И СУММЫ КВАДРАТОВ ===
Сравнение оригинального Sin и табулированного:
x       Sin(x)      TabSin(x)   Разница
-----------------------------------------------
0,000       0,000000    0,000000    0,000000
0,100       0,099833    0,097982    0,001852
0,200       0,198669    0,195963    0,002706
0,300       0,295520    0,293945    0,001576
0,400       0,389418    0,385907    0,003512
0,500       0,479426    0,472070    0,007355
0,600       0,564642    0,558234    0,006409
0,700       0,644218    0,643982    0,000235
0,800       0,717356    0,707935    0,009421
0,900       0,783327    0,771888    0,011439
1,000       0,841471    0,835841    0,005630
1,100       0,891207    0,883993    0,007214
1,200       0,932039    0,918022    0,014017
1,300       0,963558    0,952051    0,011508
1,400       0,985450    0,984808    0,000642
1,500       0,997495    0,984808    0,012687
1,600       0,999574    0,984808    0,014766
1,700       0,991665    0,984808    0,006857
1,800       0,973848    0,966204    0,007644
1,900       0,946300    0,932175    0,014125
2,000       0,909297    0,898147    0,011151
2,100       0,863209    0,862441    0,000768
2,200       0,808496    0,798488    0,010008
2,300       0,745705    0,734535    0,011170
2,400       0,675463    0,670582    0,004881
2,500       0,598472    0,594072    0,004401
2,600       0,515501    0,507908    0,007593
2,700       0,427380    0,421745    0,005635
2,800       0,334988    0,334698    0,000290
2,900       0,239249    0,236716    0,002533
3,000       0,141120    0,138735    0,002385
3,100       0,041581    0,040753    0,000828
```

```
Сумма квадратов табулированных Sin и Cos:
x       Sin²+Cos²
----------------------
0,000       1,000000
0,100       0,975345
0,200       0,970488
0,300       0,985429
0,400       0,984968
0,500       0,970398
0,600       0,975624
0,700       0,999358
0,800       0,975073
0,900       0,970586
1,000       0,985897
1,100       0,984515
1,200       0,970314
1,300       0,975910
1,400       0,998723
1,500       0,974808
1,600       0,970691
1,700       0,986371
1,800       0,984068
1,900       0,970237
2,000       0,976203
2,100       0,998094
2,200       0,974549
2,300       0,970802
2,400       0,986852
2,500       0,983628
2,600       0,970167
2,700       0,976503
2,800       0,997473
2,900       0,974298
3,000       0,970920
3,100       0,987341
```

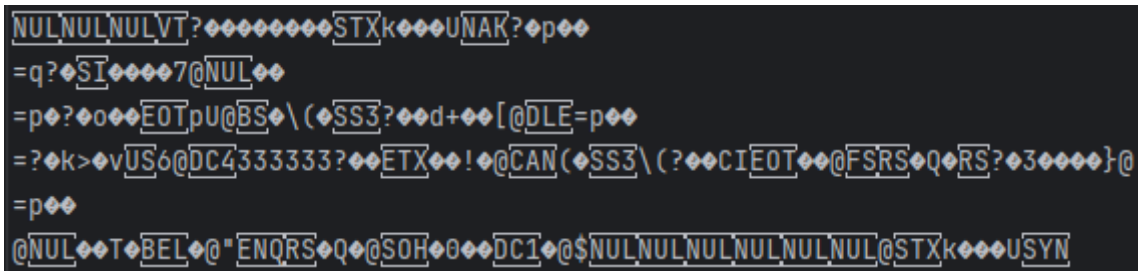
```
Исследование влияния количества точек на точность:
Точек: 5, Максимальная ошибка: 0,14639599
Точек: 10, Максимальная ошибка: 0,02983318
Точек: 20, Максимальная ошибка: 0,00681713
Точек: 50, Максимальная ошибка: 0,00102635

=== ТЕСТИРОВАНИЕ ТЕКСТОВОЙ СЕРИАЛИЗАЦИИ (ЭКСПОНЕНТА) ===
Сравнение оригинальной и восстановленной экспоненты:
x           Original            Read                Разница
------------------------------------------------------------
0,0         1,000000            1,000000            0,000000
1,0         2,718282            2,718282            0,000000
2,0         7,389056            7,389056            0,000000
3,0         20,085537           20,085537           0,000000
4,0         54,598150           54,598150           0,000000
5,0         148,413159          148,413159          0,000000
6,0         403,428793          403,428793          0,000000
7,0         1096,633158         1096,633158         0,000000
8,0         2980,957987         2980,957987         0,000000
9,0         8103,083928         8103,083928         0,000000
10,0        22026,465795        22026,465795        0,000000
```

Содержимое текстового файла exponential.txt:

11 0.0 1.0 1.0 2.718281828459045 2.0 7.38905609893065 3.0 20.085536923187668 4.0 54.598150033144236 5.0 148.4131591025766 6.0 403.4287934927351 7.0 1096.6331584284585 8.0 2980.9579870417283 9.0 8103.083927575384 10.0 22026.465794806718

```
=== ТЕСТИРОВАНИЕ БИНАРНОЙ СЕРИАЛИЗАЦИИ (ЛОГАРИФМ) ===
Сравнение оригинального и восстановленного логарифма:
x       Original    Read        Разница
-----------------------------------------------
0,1     -2,302585   -2,302585   0,000000
1,1     0,092705    0,092705    0,000000
2,1     0,740233    0,740233    0,000000
3,1     1,130147    1,130147    0,000000
4,1     1,409999    1,409999    0,000000
5,1     1,628429    1,628429    0,000000
6,1     1,807603    1,807603    0,000000
7,1     1,959502    1,959502    0,000000
8,1     2,091344    2,091344    0,000000
9,1     2,207812    2,207812    0,000000

Размер бинарного файла: 180 байт
```

Содержимое бинарного файла logatirthm.bin:



Задание 9. Сериализация табулированных функций

- Для всех классов, реализующих TabulatedFunction (ArrayTabulatedFunction, LinkedListTabulatedFunction), была добавлена поддержка сериализации двумя способами:

1) Через java.io.Serializable: Интерфейс был добавлен в список реализуемых. Это автоматический механизм, не требующий написания дополнительного кода.

2) Через java.io.Externalizable: Интерфейс был добавлен, и реализованы методы writeExternal() и readExternal(), в которых вручную прописывается логика сохранения и восстановления полей объекта.

3) Была создана функция ln(exp(x)), сериализована в файл с помощью обоих подходов и успешно десериализована.

Сравнение сериализации:

1) Serializable прост в реализации, но менее гибок и может сохранять избыточные данные.

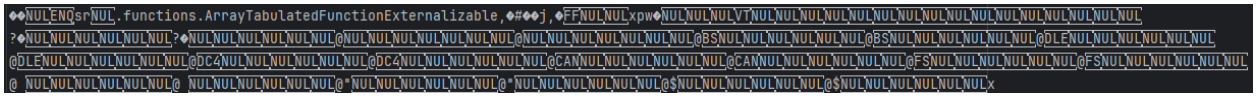
2) Externalizable требует ручного написания кода, но даёт полный контроль над форматом данных, что позволяет создавать более компактные и эффективные решения.

```
=== ТЕСТИРОВАНИЕ СЕРИАЛИЗАЦИИ ===
Исходная функция (ln(exp(x)) = x):
x       f(x)
------------------
0,0         0,000000
1,0         1,000000
2,0         2,000000
3,0         3,000000
4,0         4,000000
5,0         5,000000
6,0         6,000000
7,0         7,000000
8,0         8,000000
9,0         9,000000
10,0        10,000000

=== ТЕСТИРОВАНИЕ Serializable ===
Сериализованные данные сохранены в serializable_function.ser
Восстановленная функция:
x       f(x)
------------------
0,0         0,000000
1,0         1,000000
2,0         2,000000
3,0         3,000000
4,0         4,000000
5,0         5,000000
6,0         6,000000
7,0         7,000000
8,0         8,000000
9,0         9,000000
10,0        10,000000
Функции эквивалентны: true
```

```
=== ТЕСТИРОВАНИЕ Externalizable ===
Сериализованные данные сохранены в externalizable_function.ser
Восстановленная функция:
x       f(x)
--------------------
0,0         0,000000
1,0         1,000000
2,0         2,000000
3,0         3,000000
4,0         4,000000
5,0         5,000000
6,0         6,000000
7,0         7,000000
8,0         8,000000
9,0         9,000000
10,0        10,000000
Функции эквивалентны: true

=== СРАВНЕНИЕ РАЗМЕРОВ ФАЙЛОВ ===
Размер файла Serializable: 445 байт
Размер файла Externalizable: 250 байт
Разница: 195 байт
Externalizable файл меньше на 195 байт
```

Содержимое файлов:

1)



2)