Лабораторная работа №4

Выполнил: Пожидаев Б.В

Группа: 6204-010302D

1. СОЗДАНИЕ БАЗОВЫХ АНАЛИТИЧЕСКИХ ФУНКЦИЙ

Задача: Создать классы аналитических функций в пакете **functions.basic.**

Выполнение:

- Реализован интерфейс `Function` с методами получения границ области определения и значения функции

- Созданы классы:

  - `Exp` - экспонента (область определения: -∞...+∞)

  - `Log` - логарифм по произвольному основанию (область определения: 0...+∞)

  - `TrigonometricFunction` - абстрактный класс для тригонометрических функций

  - `Sin`, `Cos`, `Tan` - конкретные тригонометрические функции

Результат:

```
// Пример использования
Function sin = new Sin();
Function cos = new Cos();
Function exp = new Exp();
Function log = new Log(2); // логарифм по основанию 2

System.out.println("sin(Pi/2) = " + sin.getFunctionValue(Math.PI/2));
// Вывод: sin(Pi/2) = 1.0
```

---

3. РЕАЛИЗАЦИЯ ТАБУЛИРОВАННЫХ ФУНКЦИЙ

Выполнение:

- Созданы классы ArrayTabulatedFunction и LinkedListTabulatedFunction
- Интерфейс TabulatedFunction расширяет Function

- Реализована линейная интерполяция между узлами табуляции
- Добавлен класс TabulatedFunctions со статическими методами

Результат:

// Табулирование функции

TabulatedFunction tabulatedSin =

TabulatedFunctions.tabulate(new Sin(), 0, Math.PI, 10);

// Получение значения в произвольной точке

double value = tabulatedSin.getFunctionValue(1.5);

---

4. КОМБИНАЦИИ ФУНКЦИЙ И МЕТА-ФУНКЦИИ

Выполнение:

- Создан пакет functions.meta с классами:
    - Sum, Mult - арифметические операции
    - Power - возведение в степень
    - Scale, Shift - преобразования координат
    - Composition - композиция функций
- Добавлен класс Functions с фабричными методами

Результат:

java

// Создание сложной функции: sin²(x) + cos²(x)

Function sinSquared = Functions.power(new Sin(), 2);

Function cosSquared = Functions.power(new Cos(), 2);

Function identity = Functions.sum(sinSquared, cosSquared);

// Проверка тождества sin²(x) + cos²(x) = 1

for (double x = 0; x <= Math.PI; x += 0.1) {

System.out.printf("x=%.1f: %.6f%n", x, identity.getFunctionValue(x));

}

Вывод консоли:

0,000	1,000000
0,100	0,998987
0,200	0,999568
0,300	0,999105
0,400	0,999253
0,500	0,999339
0,600	0,999055
0,700	0,999691
0,800	0,998975
0,900	0,999852
1,000	0,999012
1,100	0,999456
1,200	0,999166
1,300	0,999178
1,400	0,999437
1,500	0,999017
1,600	0,999825
1,700	0,998974
1,800	0,999715
1,900	0,999047
2,000	0,999357
2,100	0,999238
2,200	0,999115
2,300	0,999546
2,400	0,998991
2,500	0,999971
2,600	0,998984
2,700	0,999590

2,800 0,999094

2,900 0,999268

3,000 0,999322

3,100 0,999064

---

5. СЕРИАЛИЗАЦИЯ ТАБУЛИРОВАННЫХ ФУНКЦИЙ

Выполнение:

- Реализованы два подхода к сериализации:
    1. Serializable - автоматическая сериализация
    2. Externalizable - ручное управление процессом
- Добавлены методы для работы с потоками:
    - Бинарный формат: outputTabulatedFunction(), inputTabulatedFunction()
    - Текстовый формат: writeTabulatedFunction(), readTabulatedFunction()

Результат:

```
// Сериализация в бинарный формат
try (ObjectOutputStream oos = new ObjectOutputStream(
   new FileOutputStream("function.bin"))) {
   oos.writeObject(function);
}


// Десериализация
try (ObjectInputStream ois = new ObjectInputStream(
   new FileInputStream("function.bin"))) {
   TabulatedFunction restored = (TabulatedFunction) ois.readObject();
}
```

Вывод консоли:

"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.2.3\lib\idea_rt.jar=60975" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -

Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\bpozi\OneDrive\Рабочий стол\лабы\ООП лабы\Lab_4\out\production\Lab_4" Main

=== ТЕСТИРОВАНИЕ БАЗОВЫХ ФУНКЦИЙ ===

Синус и косинус на [0, Pi] с шагом 0.1:

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 ===

Сравнение аналитических и табулированных функций:

x	Sin(x)	TabSin(x)	Cos(x)	TabCos(x)
0,000	0,000000	0,000000	1,000000	1,000000
0,100	0,099833	0,097982	0,995004	0,982723
0,200	0,198669	0,195963	0,980067	0,965446
0,300	0,295520	0,293945	0,955336	0,948170
0,400	0,389418	0,385907	0,921061	0,914355
0,500	0,479426	0,472070	0,877583	0,864608
0,600	0,564642	0,558234	0,825336	0,814862
0,700	0,644218	0,643982	0,764842	0,764620
0,800	0,717356	0,707935	0,696707	0,688404
0,900	0,783327	0,771888	0,621610	0,612188
1,000	0,841471	0,835841	0,540302	0,535972
1,100	0,891207	0,883993	0,453596	0,450633
1,200	0,932039	0,918022	0,362358	0,357141
1,300	0,963558	0,952051	0,267499	0,263648
1,400	0,985450	0,984808	0,169967	0,169931

1,500	0,997495	0,984808	0,070737	0,070437
1,600	0,999574	0,984808	-0,029200	-0,029056
1,700	0,991665	0,984808	-0,128844	-0,128549
1,800	0,973848	0,966204	-0,227202	-0,224761
1,900	0,946300	0,932175	-0,323290	-0,318254
2,000	0,909297	0,898147	-0,416147	-0,411747
2,100	0,863209	0,862441	-0,504846	-0,504272
2,200	0,808496	0,798488	-0,588501	-0,580488
2,300	0,745705	0,734535	-0,666276	-0,656704
2,400	0,675463	0,670582	-0,737394	-0,732920
2,500	0,598472	0,594072	-0,801144	-0,794171
2,600	0,515501	0,507908	-0,856889	-0,843917
2,700	0,427380	0,421745	-0,904072	-0,893664
2,800	0,334988	0,334698	-0,942222	-0,940984
2,900	0,239249	0,236716	-0,970958	-0,958261
3,000	0,141120	0,138735	-0,989992	-0,975537
3,100	0,041581	0,040753	-0,999135	-0,992814

=== ТЕСТИРОВАНИЕ КОМБИНАЦИЙ ФУНКЦИЙ ===

Сумма квадратов синуса и косинуса с разным количеством точек:

Количество точек: 5

x	$Sin^2(x)+Cos^2(x)$
0,000	1,000000
0,100	0,934912
0,200	0,888816
0,300	0,861714
0,400	0,853604

0,500	**0,864487**
0,600	**0,894363**
0,700	**0,943232**
0,800	**0,989312**
0,900	**0,926997**
1,000	**0,883675**
1,100	**0,859345**
1,200	**0,854009**
1,300	**0,867665**
1,400	**0,900315**
1,500	**0,951957**
1,600	**0,979028**
1,700	**0,919487**
1,800	**0,878938**
1,900	**0,857382**
2,000	**0,854819**
2,100	**0,871249**
2,200	**0,906671**
2,300	**0,961086**
2,400	**0,969150**
2,500	**0,912382**
2,600	**0,874606**
2,700	**0,855824**
2,800	**0,856034**
2,900	**0,875237**
3,000	**0,913432**
3,100	**0,970621**

Количество точек: 10

x	$Sin^2(x)+Cos^2(x)$
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**

Количество точек: 20

x	**$Sin^2(x)+Cos^2(x)$**
0,000	**1,000000**
0,100	**0,993480**
0,200	**0,995481**
0,300	**0,995876**
0,400	**0,993359**
0,500	**0,999363**
0,600	**0,993633**
0,700	**0,995118**
0,800	**0,996303**
0,900	**0,993269**
1,000	**0,998756**
1,100	**0,993816**
1,200	**0,994785**
1,300	**0,996760**
1,400	**0,993210**
1,500	**0,998181**
1,600	**0,994032**
1,700	**0,994484**
1,800	**0,997249**
1,900	**0,993183**

2,000	**0,997638**
2,100	**0,994278**
2,200	**0,994214**
2,300	**0,997769**
2,400	**0,993187**
2,500	**0,997125**
2,600	**0,994556**
2,700	**0,993976**
2,800	**0,998321**
2,900	**0,993222**
3,000	**0,996644**
3,100	**0,994864**

Количество точек: 50

x	**$Sin^2(x)+Cos^2(x)$**
0,000	**1,000000**
0,100	**0,998987**
0,200	**0,999568**
0,300	**0,999105**
0,400	**0,999253**
0,500	**0,999339**
0,600	**0,999055**
0,700	**0,999691**
0,800	**0,998975**
0,900	**0,999852**
1,000	**0,999012**
1,100	**0,999456**
1,200	**0,999166**

1,300	0,999178
1,400	0,999437
1,500	0,999017
1,600	0,999825
1,700	0,998974
1,800	0,999715
1,900	0,999047
2,000	0,999357
2,100	0,999238
2,200	0,999115
2,300	0,999546
2,400	0,998991
2,500	0,999971
2,600	0,998984
2,700	0,999590
2,800	0,999094
2,900	0,999268
3,000	0,999322
3,100	0,999064

=== ТЕСТИРОВАНИЕ ТЕКСТОВОЙ СЕРИАЛИЗАЦИИ ===

Текстовая сериализация экспоненты:

x	Исходная	Прочитанная
0,0	1,000000	1,000000
1,0	2,718282	2,718282
2,0	7,389056	7,389056
3,0	20,085537	20,085537
4,0	54,598150	54,598150

5,0 148,413159 148,413159

6,0 403,428793 403,428793

7,0 1096,633158 1096,633158

8,0 2980,957987 2980,957987

9,0 8103,083928 8103,083928

10,0 22026,465795 22026,465795

Содержимое файла exp_function.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 ===

Бинарная сериализация логарифма:

x Исходная Прочитанная

1,0 0,000000 0,000000

2,0 0,693147 0,693147

3,0 1,098612 1,098612

4,0 1,386294 1,386294

5,0 1,609438 1,609438

6,0 1,791759 1,791759

7,0 1,945910 1,945910

8,0 2,079442 2,079442

9,0 2,197225 2,197225

10,0 2,302585 2,302585

Размер бинарного файла: 164 байт

=== ТЕСТИРОВАНИЕ JAVA СЕРИАЛИЗАЦИИ ===

Java сериализация функции ln(exp(x)):

Исходная функция:

x Исходная значение

0,0 NaN

1,0 NaN

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

--- Serializable сериализация ---

После десериализации:

x Serializable значение

0,0 NaN

1,0 NaN

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

✓ Serializable: функции идентичны!

Размер файла: 445 байт

--- Externalizable сериализация ---

После десериализации:

x Externalizable значение

0,0 NaN

1,0 NaN

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

Externalizable: функции идентичны!

Размер файла: 250 байт

--- СРАВНЕНИЕ ФАЙЛОВ СЕРИАЛИЗАЦИИ ---

Serializable размер: 445 байт

Externalizable размер: 250 байт

Разница: 195 байт (43,8%)

Анализ содержимого файлов:

Serializable файл (445 байт):

Hex: ac ed 00 05 73 72 00 20 66 75 6e 63 74 69 6f 6e

73 2e 41 72 72 61 79 54 61 62 75 6c 61 74 65 64

46 75 6e 63 74 69 6f 6e c9 6a 0f c6 cf c9 85 23

02 00

Заголовок: ac ed (Java Serialization Stream)

Externalizable файл (250 байт):

Hex: ac ed 00 05 73 72 00 2e 66 75 6e 63 74 69 6f 6e

73 2e 41 72 72 61 79 54 61 62 75 6c 61 74 65 64

46 75 6e 63 74 69 6f 6e 45 78 74 65 72 6e 61 6c

69 7a

Заголовок: ac ed (Java Serialization Stream)

--- ВЫВОДЫ ---

Serializable:

+ Простая реализация (implements Serializable)

- Больший размер файла (метаданные классов)

- Медленнее из-за рефлексии

Externalizable:

+ Минимальный размер файла

+ Высокая производительность

+ Полный контроль над процессом

- Сложная реализация (нужно реализовать методы)

- Требуется конструктор по умолчанию