

Лабораторная работа №4

Выполнила: Джумашева Камилла Исфатовна

Студент 6203-010302D группы

Ход выполнения

Задание 1 и 2

В ходе выполнения работы я расширила возможности пакета `functions`. В классы `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` были добавлены конструкторы, принимающие массив точек `FunctionPoint[]`. В них реализована проверка входных данных: если точек меньше двух или нарушена их упорядоченность по координате X, выбрасывается исключение `IllegalArgumentException`.

Был выделен общий интерфейс `Function`, описывающий поведение функции одной переменной (методы получения границ области определения и вычисления значения). Интерфейс `TabulatedFunction` теперь наследуется от `Function`, что делает табулированные функции частным случаем математических функций.

Задание 3

В пакете `functions.basic` я реализовала классы для аналитически заданных функций. Классы `Exp` и `Log` реализуют интерфейс `Function` и вычисляют значения экспоненты и логарифма соответственно. В классе `Log` добавлена проверка основания (должно быть > 0 и $\neq 1$).

Для тригонометрических функций был создан абстрактный класс `TrigonometricFunction`, который определяет бесконечную область определения. От него наследуются классы `Sin`, `Cos` и `Tan`, реализующие вычисление соответствующих функций через стандартную библиотеку `Math`.

Задание 4 и 5

В пакете `functions.meta` были разработаны классы для композиции и преобразования функций:

- `Sum`, `Mult` — сумма и произведение функций. Область определения вычисляется как пересечение областей исходных функций.
- `Power` — возвведение функции в степень.
- `Scale`, `Shift` — масштабирование и сдвиг функций вдоль осей координат.
- `Composition` — суперпозиция (композиция) функций.

Для удобства работы в пакете `functions` был создан утилитный класс `Functions` со статическими методами-фабриками (`sum`, `mult`, `composition` и др.), которые возвращают

объекты соответствующих мета-функций. Конструктор класса Functions скрыт (`private`), чтобы предотвратить создание его экземпляров.

Задание 6 и 7

В класс `TabulatedFunctions` был добавлен метод `tabulate`, позволяющий преобразовать любую аналитическую функцию (реализующую интерфейс `Function`) в табулированную (`TabulatedFunction`) на заданном отрезке.

Реализованы методы ввода-вывода:

- `outputTabulatedFunction` / `inputTabulatedFunction` — работают с байтовыми потоками.
- `writeTabulatedFunction` / `readTabulatedFunction` — работают с символьными потоками. Для чтения используется класс `StreamTokenizer`, что упрощает парсинг чисел, разделенных пробелами.

Потоки внутри методов не закрываются, чтобы оставить управление ими вызывающему коду (например, для записи нескольких объектов в один файл). Исключения `IOException` прорасыпаются выше для обработки на уровне логики приложения.

Задание 8 и 9

Для проверки работоспособности был обновлен класс `Main`. Проведены тесты:

- Табулирование аналитических функций (`Sin`, `Cos`) и сравнение значений.
- Операции над функциями (проверка тригонометрического тождества).
- СерIALIZАЦИЯ и десериализация.

Для реализации сериализации:

- Интерфейс `TabulatedFunction` наследуется от `Serializable`.
- Класс `FunctionPoint` реализует `Serializable`.
- В классе `LinkedListTabulatedFunction` был реализован интерфейс `Externalizable`. Это позволило оптимизировать процесс сохранения: вместо сериализации всей структуры двусвязного списка сохраняются только количество точек и их координаты. При восстановлении список пересоздается заново, что экономит память и исключает проблемы с глубиной рекурсии при стандартной сериализации.

Результаты работы (Консоль)

Ниже представлен вывод программы, демонстрирующий корректность работы всех реализованных методов.

```
Тест аналитических функций и их табулирования
Сравнение значений:
x = 0,00 | Sin: 0,0000, TabSin: 0,0000 | Cos: 1,0000, TabCos: 1,0000
x = 0,50 | Sin: 0,4794, TabSin: 0,4721 | Cos: 0,8776, TabCos: 0,8646
x = 1,00 | Sin: 0,8415, TabSin: 0,8358 | Cos: 0,5403, TabCos: 0,5360
x = 1,50 | Sin: 0,9975, TabSin: 0,9848 | Cos: 0,0707, TabCos: 0,0704
x = 2,00 | Sin: 0,9093, TabSin: 0,8981 | Cos: -0,4161, TabCos: -0,4117
x = 2,50 | Sin: 0,5985, TabSin: 0,5941 | Cos: -0,8011, TabCos: -0,7942
x = 3,00 | Sin: 0,1411, TabSin: 0,1387 | Cos: -0,9900, TabCos: -0,9755

Тест операций (Sin^2 + Cos^2)
x = 0,00: sin^2 + cos^2 = 2,0000
x = 0,50: sin^2 + cos^2 = 2,0000
x = 1,00: sin^2 + cos^2 = 2,0000
x = 1,50: sin^2 + cos^2 = 2,0000
x = 2,00: sin^2 + cos^2 = 2,0000
x = 2,50: sin^2 + cos^2 = 2,0000
x = 3,00: sin^2 + cos^2 = 2,0000

Тест символьного ввода/вывода (Exp)
Функция записана в файл exp_char.txt
Функция считана из файла. Сравнение значений:
x = 0,0: Исходная = 1,0000, Считанная = 1,0000
x = 2,0: Исходная = 7,3891, Считанная = 7,3891
x = 4,0: Исходная = 54,5982, Считанная = 54,5982
x = 6,0: Исходная = 403,4288, Считанная = 403,4288
x = 8,0: Исходная = 2980,9580, Считанная = 2980,9580
x = 10,0: Исходная = 22026,4658, Считанная = 22026,4658

Тест байтового ввода/вывода (Log)
Функция записана в файл log_byte.bin
Функция считана из файла. Сравнение значений:
Функция считана из файла. Сравнение значений:
x = 1,0: Исходная = -0,1310, Считанная = -0,1310
x = 3,0: Исходная = 1,0942, Считанная = 1,0942
x = 5,0: Исходная = 1,6084, Считанная = 1,6084
x = 7,0: Исходная = 1,9456, Считанная = 1,9456
x = 9,0: Исходная = 2,1972, Считанная = 2,1972

Тест сериализации (Externalizable: Log(Exp(x)))
Объект (Externalizable) сериализован в файл complex_ser.bin
Объект десериализован. Проверка значений:
x = 0,0: Значение = 0,0000
x = 2,0: Значение = 2,0000
x = 4,0: Значение = 4,0000
x = 6,0: Значение = 6,0000
x = 8,0: Значение = 8,0000
x = 10,0: Значение = 10,0000
```

Рис. 1

Заключение

В ходе работы наиболее трудоемкой частью стало корректное использование потоков ввода-вывода и реализация сериализации через Externalizable. Было важно правильно восстановить структуру связного списка в методе readExternal, инициализировав фиктивную голову списка (head) перед добавлением считанных точек, чтобы избежать NullPointerException. Также использование StreamTokenizer потребовало внимательной настройки флагов (например, eolIsSignificant(false)), чтобы корректно считывать последовательности чисел.