Лабораторная работа №4

Иванова Виктория Павловна 6203-010302D

Задание 1

В классах ArrayTabulatedFunction и LinkedListTabulatedFunction добавила 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. Для обеспечения инкапсуляции создаются копии переданных точек.

Задание 2

В пакете functions создалп интерфейс Function, содержащий методы getLeftDomainBorder, getRightDomainBorder и getFunctionValue. Интерфейс TabulatedFunction теперь расширяет Function. Соответствующие методы были удалены из TabulatedFunction, что убрало проблему дублирования

Задание 3

Создала пакет functions.basic, в нём описала классы ряда функций, Exp реализует экспоненту с бесконечной областью определения, используя Math.exp. Log представляет логарифм с задаваемым основанием, проверяя его корректность в конструкторе. Для тригонометрических функций создана иерархия с базовым классом TrigonometricFunction, определяющим общую область определения, и производными классами Sin, Cos и Tan, использующими соответствующие методы из Math. Для получения значений воспользовалась методами Math.sin(), Math.cos() и Math.tan().

Задание 4

Создала пакет functions.meta, в нём описала классы функций, позволяющие комбинировать функции. Создала класс Sum, объекты которого представляют собой функции, являющиеся суммой двух других функций, конструктор класса получает ссылки типа Function на объекты суммируемых функций, а область определения функции получается как пересечение областей определения исходных функций. Класс Mult реализован аналогично. Класс Power создает функцию, представляющую собой возведение значений исходной функции в заданную степень. Класс Scale реализует масштабирование функции вдоль осей координат. Значение функции рассчитывается как масштабированное по Y значение исходной функции от масштабированного по X аргумента. Коэффициенты масштабирования могут быть отрицательными, что позволяет осуществлять отражение функции относительно осей координат. Класс Shift выполняет сдвиг функции вдоль координатных осей. Вычисление значения функции производится как значение исходной функции от аргумента, сдвинутого на величину shiftX, плюс сдвиг по Y. Класс Composition реализует композицию двух функций. Вычисление значения происходит путем последовательного применения функций: значение первой функции в точке x передается как аргумент второй функции.

Задание 5

В пакете functions создала класс Functions, содержащий вспомогательные статические методы для работы с функциями. Класс содержит следующие методы:
public static Function shift(Function f, double shiftX, double shiftY) – возвращает объект функции, полученной из исходной сдвигом вдоль осей;

public static Function scale(Function f, double scaleX, double scaleY) – возвращает объект функции, полученной из исходной масштабированием вдоль осей;public static Function power(Function f, double power) – возвращает объект функции, являющейся заданной степенью исходной public static Function sum(Function f1, Function f2) – возвращает объект функции, являющейся суммой двух исходных; public static Function mult(Function f1, Function f2) – возвращает объект функции, являющейся произведением двух исходных; public static Function composition(Function f1, Function f2) – возвращает объект функции, являющейся композицией двух исходных. Методы shift, scale, power, sum, mult, composition инкапсулируют создание объектов соответствующих классов, обеспечивая удобство и читаемость для пользователей системы.

Задание 6

Реализовала класс TabulatedFunctions с методом tabulate, который преобразует аналитическую функцию в табулированную на заданном отрезке. Метод проверяет, что границы табулирования не выходят за область определения функции, и выбрасывает IllegalArgumentException при нарушении этого условия. Возвращается интерфейсный тип TabulatedFunction, что обеспечивает гибкость выбора реализации.

Задание 7

В класс TabulatedFunctions добавила четыре метода для работы с потоками ввода-вывода. Метод outputTabulatedFunction выполняет запись табулированной функции в байтовый. Метод inputTabulatedFunction осуществляет чтение из байтового потока. Для работы с текстовыми потоками реализован метод writeTabulatedFunction, Метод readTabulatedFunction выполняет чтение из символьного потока с применением класса StreamTokenizer, который разбивает входной текст на токены. Метод последовательно считывает количество точек и затем пары координат, создавая на их основе табулированную функцию.

При проектировании методов пробрасывала исключения IOException наружу, чтобы предоставить вызывающему коду гибкость в обработке ошибок ввода-вывода согласно конкретному контексту использования. Потоки не закрываются внутри методов, так как управление жизненным циклом потоков должно оставаться за кодом, который их создал, что позволяет повторно использовать потоки для нескольких операций и избегать преждевременного закрытия.

Задание 8

Проверила работу написанных классов. Созданы и протестированы функции Sin и Cos, сравнены аналитические и табулированные значения. Исследована сумма квадратов $\sin^2 + \cos^2$, показавшая ожидаемое значение близкое к 1.0. Изучено влияние количества точек на точность табулирования - с увеличением точек от 5 до 40 точность возрастает. Протестирована работа с файлами для экспоненты и логарифма, подтверждена идентичность исходных и восстановленных данных.

Задание 9

Реализовала сериализацию двумя способами. Протестирована сериализация функции ln(exp(x)) = x, подтверждена полная идентичность исходных и десериализованных значений. Выявлены преимущества Serializable в простоте реализации и Externalizable в эффективности и контроле над процессом. Выбрала сериализацию Externalizable, тк у нее выше производительность, как записываются только нужные данные гибкость. Также она отличается от **Serializable ручным управлением** - сами контролируете что и как сериализуется.