Лабораторная работа №4

Задание на лабораторную работу

Расширить возможности пакета для работы с функциями одной переменной добавив интерфейсы и классы для аналитически заданных функций, а также методы ввода и вывода табулированных функций.

Задание 1 *(0,2 балла)*

В классах `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` добавьте 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 должны выбрасывать исключение `IllegalArgumentException`. При написании конструкторов обеспечьте корректную инкапсуляцию.

Задание 2 *(0,2 балла)*

В пакете `functions` создайте интерфейс `Function`, описывающий функции одной переменной и содержащий следующие методы:

- `public double getLeftDomainBorder()` – возвращает значение левой границы области определения функции;
- `public double getRightDomainBorder()` – возвращает значение правой границы области определения функции;
- `public double getFunctionValue(double x)` – возвращает значение функции в заданной точке.

Исключите соответствующие методы из интерфейса `TabulatedFunction` и сделайте так, чтобы он расширял интерфейс `Function`. Теперь табулированные функции буду частным случаем функций одной переменной.

Задание 3 *(0,7 балла)*

Создайте пакет `functions.basic`, в нём будут описаны классы ряда функций, заданных аналитически.

Создайте в пакете публичный класс `Exp`, объекты которого должны вычислять значение экспоненты. Класс должен реализовывать интерфейс `Function`. Для вычисления экспоненты следует воспользоваться методом `Math.exp()`, а для возвращения значений границ области определения – константами из класса `Double`.

Аналогично, создайте класс `Log`, объекты которого должны вычислять значение логарифма по заданному основанию. Основание должно передаваться как параметр конструктора. Для вычисления логарифма следует воспользоваться методом `Math.log()`.

Прежде, чем перейти к описанию классов для тригонометрических функций (синуса, косинуса и тангенса), обратите внимание на то, что область определения этих функций совпадает, поэтому описывать одинаковые методы в этих классах будет достаточно странным. Проще будет описать базовый класс с реализацией этих методов, а классы конкретных функций наследовать от него.

Создайте класс `TrigonometricFunction`, реализующий интерфейс `Function` и описывающий методы получения границ области определения.

Создайте наследующие от него публичные классы `Sin`, `Cos` и `Tan`, объекты которых вычисляют, соответственно, значения синуса, косинуса и тангенса. Для получения значений следует воспользоваться методами `Math.sin()`, `Math.cos()` и `Math.tan()`.

Задание 4 *(0,7 балла)*

Создайте пакет `functions.meta`, в нём будут описаны классы функций, позволяющие комбинировать функции.

Создайте класс `Sum`, объекты которого представляют собой функции, являющиеся суммой двух других функций. Класс должен реализовывать интерфейс `Function`. Конструктор класса должен получать ссылки типа `Function` на объекты суммируемых функций, а область определения функции должна получаться как пересечение областей определения исходных функций.

Аналогично, создайте класс `Mult`, объекты которого представляют собой функции, являющиеся произведением двух других функций.

Создайте класс `Power`, объекты которого представляют собой функции, являющиеся степенью другой функции. Конструктор класса должен получать ссылку на объекты базовой функции и степень, в которую должны возводиться её значения. Область определения функции можно считать совпадающей с областью определения исходной функции (хотя математически это не всегда так).

Создайте класс `Scale`, объекты которого описывают функции, полученные из исходных функций путём масштабирования вдоль осей координат. Конструктор класса должен получать ссылку на объект исходной функции, а также коэффициенты масштабирования вдоль оси абсцисс и оси ординат. Область определения функции должна получаться из области определения исходной функции масштабированием вдоль оси абсцисс, а значение функции – масштабированием значения исходной функции вдоль оси ординат. Коэффициенты масштабирования могут быть отрицательными.

Аналогично, создайте класс `Shift`, объекты которого описывают функции, полученные из исходных функций путём сдвига вдоль осей координат.

Также создайте класс `Composition`, объекты которого описывают композицию двух исходных функций. Конструктор класса должен получать ссылки на объекты первой и второй функции. Область определения функции можно считать совпадающей с областью определения исходной функции (хотя математически это не всегда так).

Задание 5 *(0,4 балла)*

В пакете `functions` создайте класс `Functions`, содержащий вспомогательные статические методы для работы с функциями. Сделайте так, чтобы в программе вне этого класса нельзя было создать его объект. Класс должен содержать 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.

При написании методов следует воспользоваться созданными ранее классами из пакета `functions.meta`.

Задание 6 *(0,3 балла)*

В пакете `functions` создайте класс `TabulatedFunctions`, содержащий вспомогательные статические методы для работы с табулированными функциями. Сделайте так, чтобы в программе вне этого класса нельзя было создать его объект.

Опишите в классе метод `public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount)`, получающий функцию и возвращающий её табулированный аналог на заданном отрезке с заданным количеством точек.

Если указанные границы для табулирования выходят за область определения функции, метод должен выбрасывать исключение `IllegalArgumentException`.

Поскольку метод возвращает ссылку интерфейсного типа, можно возвращать объект любого из классов, реализующих этот интерфейс. В последующих работах в код будет добавлена возможность выбора класса для создания экземпляра.

Задание 7 *(1,4 балла)*

В класс `TabulatedFunctions` добавьте следующие методы.

Метод вывода табулированной функции в байтовый поток `public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out)` должен в указанный поток вывести значения, по которым потом можно будет восстановить табулированную функцию, а именно количество точек в ней и значения координат точек.

Метод ввода табулированной функции из байтового потока `public static TabulatedFunction inputTabulatedFunction(InputStream in)` должен считывать из указанного потока данные о табулированной функции, создавать и настраивать её объект и возвращать его из метода.

Метод записи табулированной функции в символьный поток `public static void writeTabulatedFunction(TabulatedFunction function, Writer out)` должен в указанный поток вывести значения, по которым потом можно будет восстановить табулированную функцию, а именно количество точек в ней и значения координат точек. Проще всего считать, что значения записываются в строку и разделяются пробелами.

Метод чтения табулированной функции из символьного потока `public static TabulatedFunction readTabulatedFunction(Reader in)` должен считывать из указанного потока данные о табулированной функции, создавать и настраивать её объект и возвращать его из метода.

При написании методов в первых трёх случаях необходимо воспользоваться потоками-обёртками, облегчающими ввод и вывод данных в требующейся форме, а в четвёртом случае – классом `StreamTokenizer`.

При написании методов, считывающих табулированную функцию, следует считать, что данные в потоке записаны правильные данные (проверку корректности вводимых данных делать не следует).

Поскольку методы ввода и чтения возвращают ссылку интерфейсного типа, можно возвращать объект любого из классов, реализующих этот интерфейс. В последующих работах в код будет добавлена возможность выбора класса для создания экземпляра.

Подумайте и обоснуйте, как следует в этих методах поступить с возникающим исключением `IOException`.

Подумайте и обоснуйте, следует ли закрывать потоки внутри этих методов.

Задание 8 *(0,3 балла)*

Проверьте работу написанных классов.

Создайте по одному объекту классов `Sin` и `Cos`, выведите в консоль значения этих функций на отрезке от 0 до 2π с шагом 0,1.

С помощью метода `TabulatedFunctions.tabulate()` создайте табулированные аналоги этих функций на отрезке от 0 до 2π с 10 точками. Выведите в консоль значения этих функций на отрезке от 0 до 2π с шагом 0,1 и сравните со значениями исходных функций.

С помощью методов класса `Functions` создайте объект функции, являющейся суммой квадратов табулированных аналогов синуса и косинуса. Выведите в консоль значения этой функций на отрезке от 0 до 2π с шагом 0,1. Попробуйте изменять количество точек в табулированных аналогах и исследуйте, как при этом изменяется результирующая функция.

С помощью метода `TabulatedFunctions.tabulate()` создайте табулированный аналог экспоненты на отрезке от 0 до 10 с 11 точками. С помощью метода `TabulatedFunctions.writeTabulatedFunction()` выведите его в файл. Далее с помощью метода `TabulatedFunctions.readTabulatedFunction()` считайте табулированную функцию из этого файла. Выведите и сравните значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

С помощью метода `TabulatedFunctions.tabulate()` создайте табулированный аналог логарифма по натуральному основанию на отрезке от 0 до 10 с 11 точками. С помощью метода `TabulatedFunctions.outputTabulatedFunction()` выведите его в файл (имя файла должно отличаться от предыдущего случая). Далее с помощью метода `TabulatedFunctions.inputTabulatedFunction()` считайте табулированную функцию из этого файла. Выведите и сравните значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

Изучите содержимое всех получаемых файлов и сделайте выводы о преимуществах и недостатках каждого из форматов хранения.

Задание 9 *(0,8 балла)*

Сделайте так, чтобы объекты всех классов, реализующих интерфейс `TabulatedFunction`, были сериализуемыми.

Для этого рассмотрите два случая:

1) с использованием интерфейса java.io.Serializable

2) с использованием интерфейса java.io.Externalizable

Проверьте работу написанных классов. С помощью метода `TabulatedFunctions.tabulate()` и метода класса `Functions` создайте табулированный аналог логарифма по натуральному основанию, взятого от экспоненты на отрезке от 0 до 10 с 11 точками. Сериализуйте полученный объект в файл (имя файла должно отличаться от предыдущих случаев). Далее десериализуйте табулированную функцию из этого файла. Выведите значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

Изучите содержимое файлов, получаемых при реализации механизма сериализации с использованием интерфейса java.io.Serializable и при реализации механизма сериализации с использованием интерфейса java.io.Externalizable. Сделайте выводы о преимуществах и недостатках каждого из способов.