Министерство образования и науки Российской Федерации  
  
Федеральное государственное автономное образовательноеучреждение высшего образования«Самарский государственный аэрокосмический университет имени академика С.П. Королева(национальный исследовательский университет)»

Факультет информатики  
Кафедра технической кибернетики

**Лабораторная работа № 4  
по курсу «Объектно-ориентированное программирование»**

Студент: Борисов Д.С.   
группа 6210  
Преподаватель:

Самара 2018

Задание

**1 Задание:** В классах ArrayTabulatedFunction и LinkedListTabulatedFunction добавьте конструкторы инициализации от массива точек, учитывая исключительные ситуации.

**2 Задание:** В пакете functions создайте интерфейс FunctionImpl, описывающий функции одной переменной и содержащий следующие методы: получение левой и правой границы области определения, значение у по х. Исключите соответствующие методы из интерфейса TabulatedFunction и сделайте так, чтобы он расширял интерфейс Function. Теперь табулированные функции буду частным случаем функций одной переменной.

**3 Задание:** Создайте пакет functions.basic, в нём будут описаны классы ряда функций, заданных аналитически: Exp, Log, Sin, Cos и Tan. Последние три класса должны наследоваться от класса TrigonometricFunction, который в свою очередь должен реализовывать интерфейс Function.

**4 Задание:** Создайте пакет functions.meta, в нём будут описаны классы функций, позволяющие комбинировать функции: Sum, Mult, Scale, Shift, Composition.

**5 Задание:** 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.

**6 Задание:** 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.

Обработать исключительные ситуации.

**7 задание:** В класс TabulatedFunctions добавьте следующие методы: метод вывода табулированной функции в байтовый поток, метод ввода табулированной функции из байтового потока, метод записи табулированной функции в символьный поток, метод чтения табулированной функции из символьного потока.

**8 Задание:** Проверьте работу написанных классов.

**9 Задание:** 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

Реферат

В процессе написания тестовых заданий необходимо создать несколько классов и интерфейсов для обработки и комбинирования табулированных функций, а также методов для ввода и вывода из потоков.

Цель работы – 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.

Содержание

[Введение 6](#_Toc448163917)

[Выполнение заданий 7](#_Toc448163918)

[Заключение 8](#_Toc448163921)

Введение

Лабораторная работа №4 представляет собой ознакомление с основами создания интерфейсов, работы со статическими методами, реализацией методов ввода и вывода из потоков.

**ВЫПОЛНЕНИЕ ЗАДАНИЙ**

**1 Задание:** Конструктор инициализации от массива точек мало чем отличается от других, разве что необходимо присваивать новым точкам точки массива.

**2 Задание:** Интерфейс Function будет содержать методы получения границы области определения и метод получения значения ординаты по х. Необходимо их исключить из интерфейса TabulatedFunction, который будет расширять интерфейс Function.

**3 Задание:** В пакете functions.basic будут содержаться классы, объекты которых будут содержать соответствующие преобразованные функции с помощью статических методов класса Math. Функции косинуса, синуса и тангенса следует объединить в общий класс тригонометрических функций.

**4 Задание:** В пакете functions.meta создаются классы, комбинирующие функции соответствующим образом. Для этого достаточно написать методы возвращающие комбинируемые функции.

**5 Задание:** Класс Functions содержит методы, реализованные путём возвращения объекта заносимой функции, изменённой в соответствии именем метода.

**6 Задание:** Созданный класс TabulatedFunctions содержит статические методы, реализованные путём вызова обычного конструктора с последующим изменением значения ординат согласно получаемой функции.

**7 Задание:** Теперь необходимо в этом классе реализовать методы чтения и занесения в два типа потоков.

Методы вывода табулированной функции в байтовый поток и ввода из него реализовываются с помощью классов обёрток DataOutputStream и DataInputStream и их соответствующих методов.

Метод записи табулированной функции в символьный поток происходит с помощью класса BufferedWriter, представляющий собой буфер, в который заносятся через метод write символьный класс string.

Метод чтения табулированной функции из символьного потока происходит с помощью класса StreamTokenizer, который получает токен с помощью метода nextToken. Если полученный токен является числом, то он заносится в поле nval, иначе в sval.

**8 Задание:** Проверяется каждый метод обработки функций, конструкторы, а также методы ввода и вывода с потоков.

**9 Задание:** Для поддержки классом его сериализации, достаточно чтобы класс реализовывали один из интерфейсов java.io.Serializable или java.io.Externalizable. Для последнего необходимо ещё переопределить 2 метода сериализации. Проверка сериализации происходит всё в том же методе main().

Заключение

В процессе работы было создано множество классов, обрабатывающие и комбинирующие табулированные функции, интерфейс, описывающий функции одной переменной, а также класс со статическими методами ввода и вывода с потоков.