Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №4

Дисциплина: «ООП»

Тема «Дополнение пакета для работы с функциями одной переменной, добавив интерфейсы и классы для аналитически заданных функций, а также методы ввода и вывода табулированных функций.»

Выполнил: Забанов Захар Денисович

Группа: 6201-120303D

Самара, 2025

Задание 1

Конструкторы для массивов точек

Ход выполнения:
В классы ArrayTabulatedFunction и LinkedListTabulatedFunction добавлены конструкторы, принимающие массив объектов FunctionPoint[]. Конструкторы проверяют:

- Наличие минимум 2 точек
- Упорядоченность точек по возрастанию X

Ключевой код:

```
public ArrayTabulatedFunction(FunctionPoint[] points) {

   if (points.length < 2) throw new IllegalArgumentException(...);

   for (int i = 1; i < points.length; i++) {

      if (points[i].getX() <= points[i-1].getX() + EPSILON)

         throw new IllegalArgumentException(...);

   }

   // Создание копии для инкапсуляции

   this.points = Arrays.copyOf(points, points.length);

}
```

Результат: Обе реализации корректно работают с готовыми массивами точек, обеспечивая инкапсуляцию данных.

Задание 2

Интерфейс Function

Ход выполнения:
Создан интерфейс Function с базовыми методами:

- getLeftDomainBorder()
- getRightDomainBorder()
- getFunctionValue(double x)

Интерфейс TabulatedFunction теперь расширяет Function, что делает табулированные функции частным случаем функций одной переменной.

Результат: Единый интерфейс для всех типов функций, обеспечивающий полиморфизм.

Задание 3

Аналитические функции

Ход выполнения:
Создан пакет functions.basic со следующими классами:

1. **Exp** - экспонента (Math.exp())
2. **Log** - логарифм с заданным основанием (Math.log())
3. **TrigonometricFunction** - базовый класс для тригонометрических функций
4. **Sin, Cos, Tan** - конкретные тригонометрические функции

Пример использования:

```
Function sinFunc = new Sin();

double value = sinFunc.getFunctionValue(Math.PI/2); // 1.0
```

Результат: Библиотека основных математических функций.

Задание 4

Мета-функции

Ход выполнения:
Создан пакет functions.meta с классами для комбинирования функций:

1. **Sum** - сумма двух функций
2. **Mult** - произведение двух функций
3. **Power** - функция в степени
4. **Scale** - масштабирование по осям
5. **Shift** - сдвиг по осям
6. **Composition** - композиция функций

Особенность: Область определения для операций с двумя функциями вычисляется как пересечение областей определения.

Задание 5

Утилитный класс Functions

Ход выполнения:
Создан класс Functions со статическими методами-фабриками:

- shift(), scale(), power(), sum(), mult(), composition()

Класс имеет приватный конструктор для предотвращения создания экземпляров.

Результат: Удобный API для создания мета-функций.

Задание 6

Табулирование функций

Ход выполнения:
В классе TabulatedFunctions реализован метод:

public static TabulatedFunction tabulate(Function function,

double leftX,

double rightX,

int pointsCount)

Метод проверяет границы области определения и создает табулированный аналог функции.

Результат: Возможность преобразования аналитических функций в табулированные.

Задание 7

Ввод/вывод функций

Ход выполнения:
Добавлены методы для работы с потоками:

1. **Бинарный формат:**
   - outputTabulatedFunction() / inputTabulatedFunction()
   - Использует DataOutputStream / DataInputStream
2. **Текстовый формат:**
   - writeTabulatedFunction() / readTabulatedFunction()
   - Использует Writer / Reader и StreamTokenizer

Принципы:

- Исключения IOException пробрасываются вызывающему коду
- Потоки не закрываются внутри методов

Задание 8

Тестирование

Ход выполнения:
Создан комплексный тест в Main.java, проверяющий:

1. **Сравнение аналитических и табулированных функций:**

// Консольный вывод:

x Sin(x) TabSin(x) Cos(x) TabCos(x)

0.0000 0.0000 0.0000 1.0000 1.0000

0.1000 0.0998 0.0990 0.9950 0.9952

...

2. **Исследование точности:**
   - При 10 точках: средняя ошибка ~0.015
   - При 20 точках: средняя ошибка ~0.004
   - При 5 точках: средняя ошибка ~0,095
3. **Форматы хранения:**
   - Бинарный: компактный (180 байт), нечитаемый
   - Текстовый: читаемый (291 байт), больше размер

Вывод: Линейная интерполяция дает погрешность $O(h^2)$, где h - шаг табулирования.

Задание 9

Сериализация

Ход выполнения:
Реализованы два способа сериализации:

1. **Serializable** (автоматическая):

public class ArrayTabulatedFunction implements TabulatedFunction, Serializable

2. **Externalizable** (ручная):

public class LinkedListTabulatedFunction implements TabulatedFunction, Externalizable

Тестирование:

Сравнение сериализации:

Serializable файл: 485 байт (с метаданными)

Externalizable файл: 105 байт (только данные)

Десериализованные значения совпадают с исходными

Вывод: Externalizable эффективнее, но требует больше кода.

Выводы

1. **Архитектура:** Создана иерархия классов с единым интерфейсом Function
2. **Гибкость:** Комбинирование функций через мета-операции
3. **Сериализация:** Реализованы два подхода с разными компромиссами
4. **Точность:** Табулированные функции аппроксимируют аналитические с погрешностью $O(1/n^2)$
5. **Форматы:** Бинарный формат компактнее, текстовый - читаемее

Работа демонстрирует принципы ООП: наследование, полиморфизм, инкапсуляция, а также практические аспекты работы с файлами и сериализацией в Java.