Cognizant Java FSE – (Deep Skilling)

(WEEK -1)

**Module 1**: Design Patterns and Principles

**Module 2:** Data Structures and Algorithms

**Submitted by**  
**Name:** SOLLETI VENKATA KUSUMA  
**Roll No:** 111522102145  
**Email:** [22102145@rmd.ac.in](mailto:22102145@rmd.ac.in)

**College:** RMD ENGINEERING COLLEGE  
**Batch:** Java FSE – 2025

**Exercise 1: Implementing the Singleton Pattern**

**Objective**

This Java project demonstrates the implementation of the Singleton Design Pattern. The Singleton pattern ensures that a class has only one instance and provides a global point of access to that instance.

In this example, a Logger class is implemented as a singleton to provide consistent logging behavior across the application.

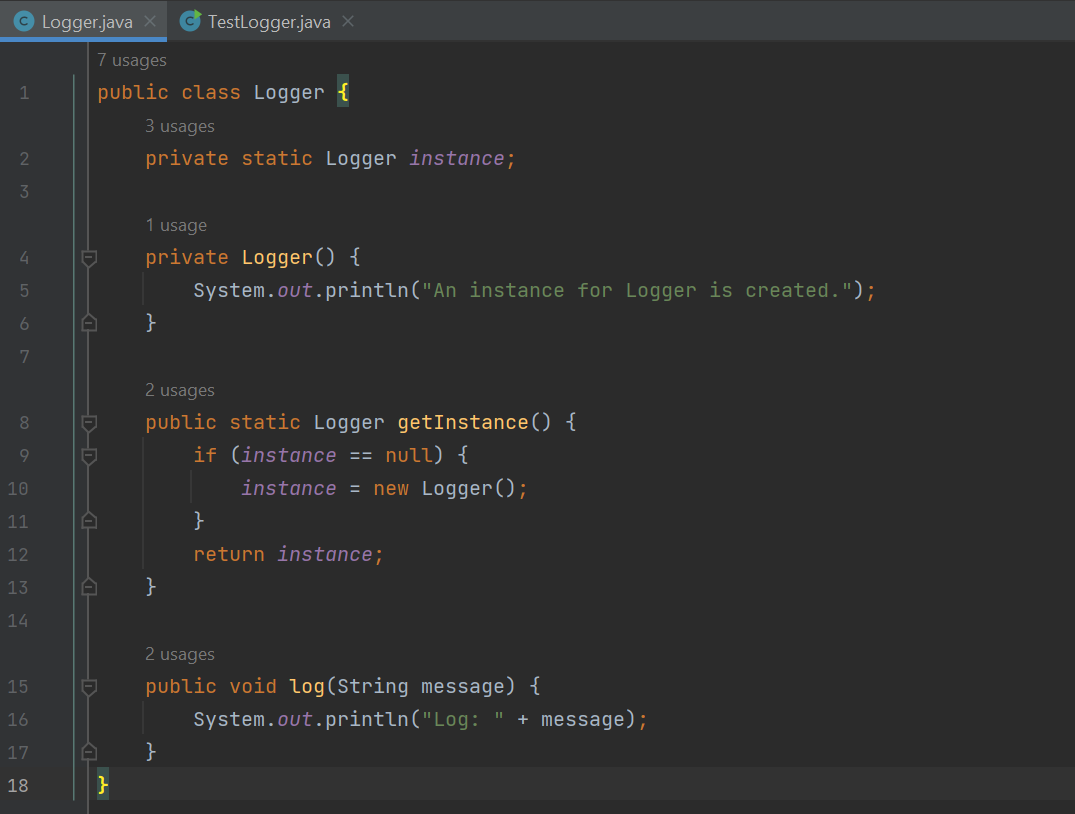
**Features**

* Ensures only one instance of the Logger class is created.
* Provides a global access point using Logger.getInstance().
* Demonstrates lazy initialization of the singleton instance.
* Simple logging functionality using the log(String message) method.
* Includes a test class to verify singleton behavior.

**Files Included:**

1. Logger.java - Singleton class with a private constructor
2. TestLogger.java - Test class to demonstrate singleton behavior

Logger.java



TestLogger.java

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A black screen with a black border

AI-generated content may be incorrect.

**Exercise 2: Implementing the Factory Method Pattern**

**Objective**

This project demonstrates the Factory Method Design Pattern in Java. The goal is to build a simple Document Management System that can create different types of documents such as Word, PDF, and Excel using a factory interface.

**Features**

* Follows Factory Method Design Pattern
* Supports multiple document types
* Easy to extend with new document formats

**Files Included**

1. Document.java (interface)
2. WordDocument.java
3. PdfDocument.java
4. ExcelDocument.java
5. DocumentFactory.java (abstract)
6. WordDocumentFactory.java
7. PdfDocumentFactory.java
8. ExcelDocumentFactory.java
9. TestFactoryMethod.java

Document.java

A screenshot of a computer

AI-generated content may be incorrect.

WordDocument.javaA screen shot of a computer code

AI-generated content may be incorrect.

PdfDocument.javaA screen shot of a computer code

AI-generated content may be incorrect.

ExcelDocument.java

A screen shot of a computer program

AI-generated content may be incorrect.

DocumentFactory.java

A screen shot of a computer

AI-generated content may be incorrect.

WordDocumentFactory.java

A screen shot of a computer program

AI-generated content may be incorrect.

PdfDocumentFactory.java

A computer screen shot of a program code

AI-generated content may be incorrect.

ExcelDocumentFactory.java

A screen shot of a computer code

AI-generated content may be incorrect.

TestFactoryMethod.java

A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A screen shot of a computer

AI-generated content may be incorrect.

**Exercise 3: E-commerce Platform Search Function**

**Objective**

The main objective of this task is to:

Design and implement efficient search functionality for an e-commerce platform by understanding and comparing different search algorithms (Linear Search and Binary Search), analyzing their performance using asymptotic notation, and applying them in a Java program.

This helps in optimizing the performance of product searches on a platform that could scale to thousands or millions of products.

**Features**

**1. Object-Oriented Design**

* Use of a Product class to represent items with productId, productName, and category.

**2. Search Algorithms**

* **Linear Search**: Works on unsorted arrays. Simple but slower for large datasets.
* **Binary Search**: Requires sorted arrays. Faster (logarithmic time complexity).

**3. Performance Analysis**

* Introduction to **Big O notation** to compare algorithm efficiency.
* Discussion of **best**, **average**, and **worst-case scenarios** for both algorithms.

**Files Included:**

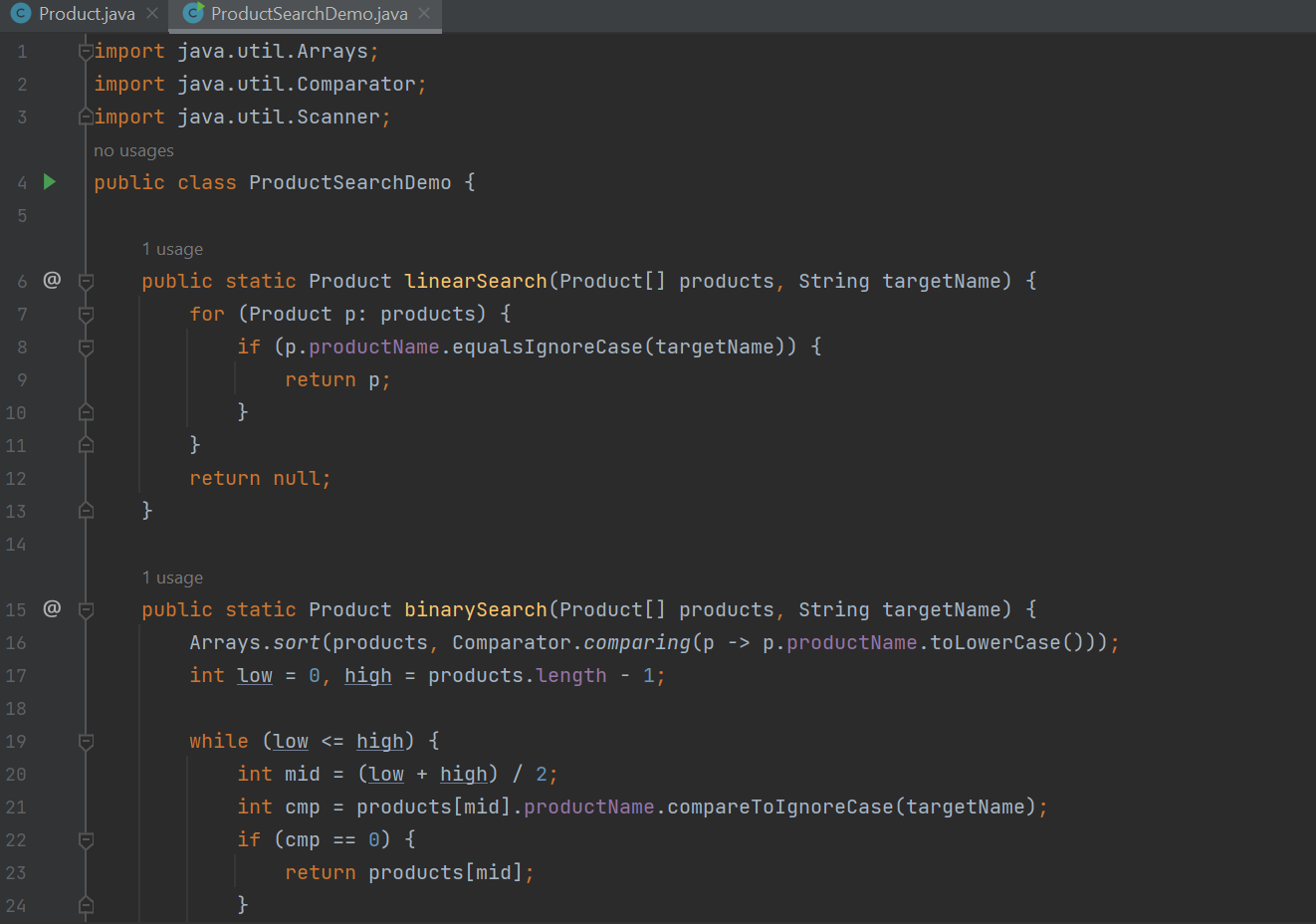
1. Product.java
2. ProductSearchDemo.java

Product.java

A screen shot of a computer

AI-generated content may be incorrect.

ProductSearchDemo.java



A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 4: Financial Forecasting**

**Objective**

The objective of this exercise is to:

Build a financial forecasting tool using recursion to compute the future value of an investment based on a constant annual growth rate, while gaining understanding of recursive algorithms, their time complexity, and optimization techniques.

**Features**

**1. Recursive Financial Forecasting**

* Uses a recursive function to compute future investment value using the formula:

FV=PV×(1+r)^n

2. **User Input**

* Accepts user input for:
  + Present Value (PV)
  + Annual Growth Rate (r)
  + Number of Years (n)

FinancialForecast.java

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A black screen with a black border

AI-generated content may be incorrect.