Cognizant FSE – Java Assignments(Deep Skilling)(WEEK -1)

**Module 1**: Design Patterns and Principles

**Module 2:** Data Structures and Algorithms

**Submitted by**  
**Name:** Chirumavilla Akshitha  
**Roll No:** 111622104013  
**Email:** chirec013@rmkcet.ac.in  
**College:** RMKCET  
**Batch:** Java FSE – 2025

Assignment 1: Singleton Pattern - Logger Class

Objective: To implement the Singleton Design Pattern by creating a Logger utility class that ensures only one instance is created and used throughout the application.

Design Pattern Used:

Singleton Pattern

* Ensures a class has only one instance.
* Provides a global point of access to it.
* Commonly used in logging, config, and cache systems.

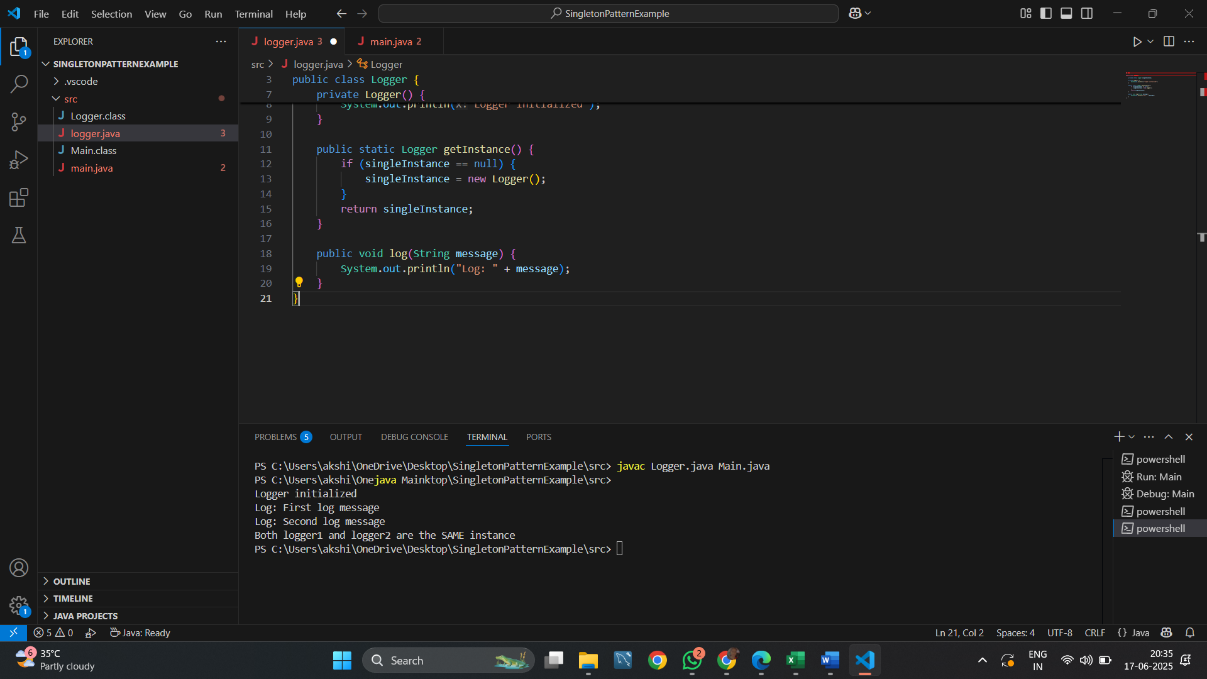
Files Included:

Logger.java – Singleton Logger class

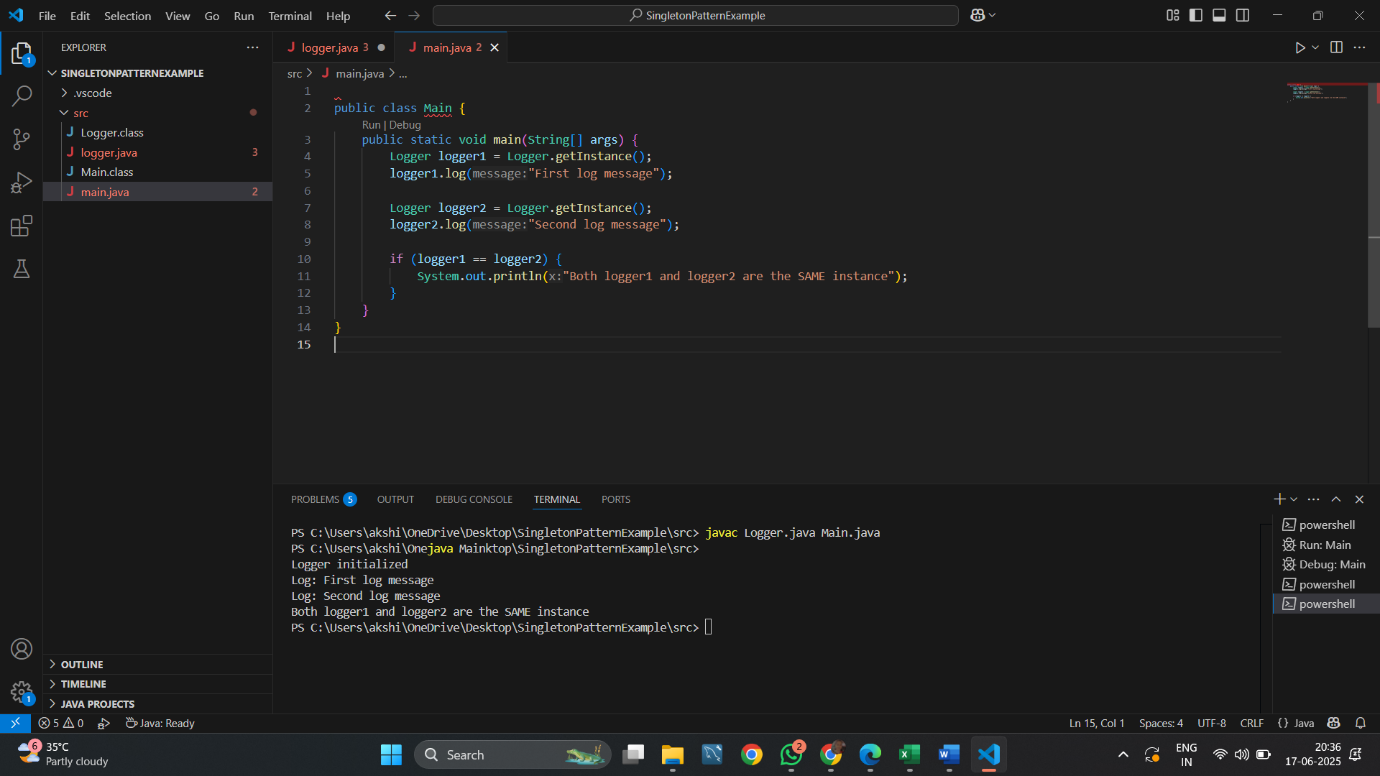
Main.java – Test class to validate the Singleton behavior

Java Code:

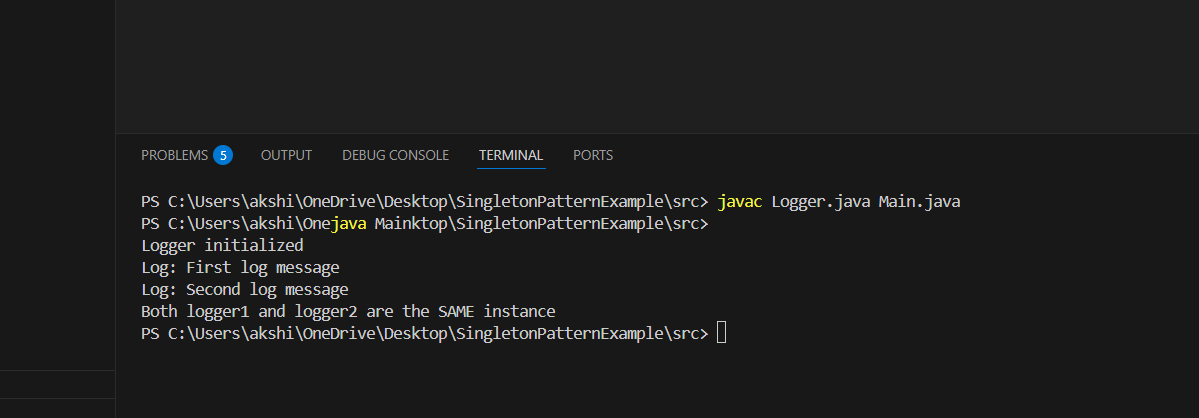
Logger.java



Main.java



Output:



Summary:

* This assignment demonstrates how to use the Singleton DesignPattern in Java.
* Only one instance of Logger is created.
* getInstance() always returns the same object.
* Helps manage logging consistently across the app.

Assignment 2: Implementing the Factory Method Pattern

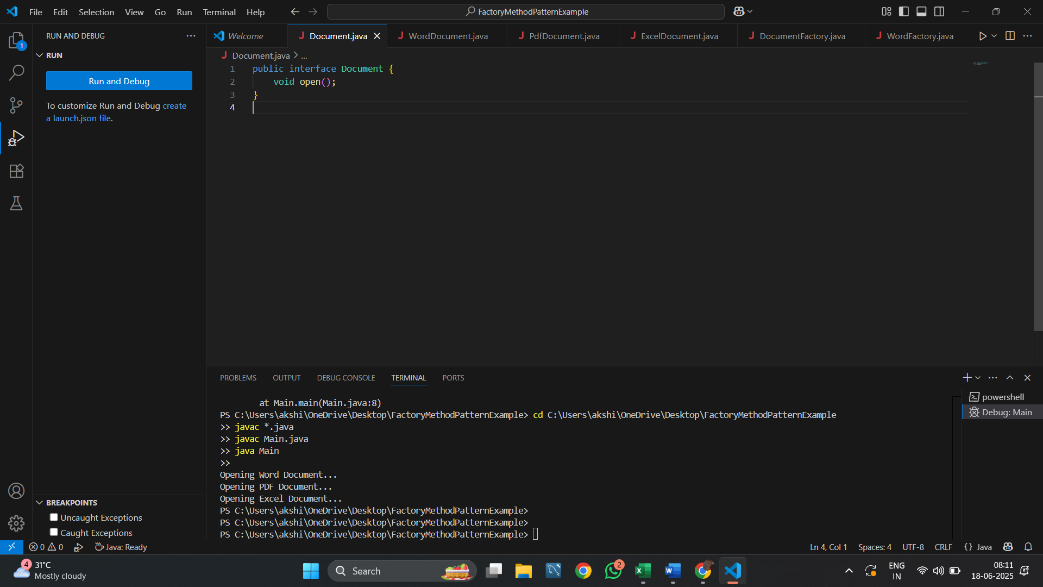
Objective: To implement the Factory Method Design Pattern in Java to create different types of documents like Word, PDF, and Excel using factories.

Java Files Used:

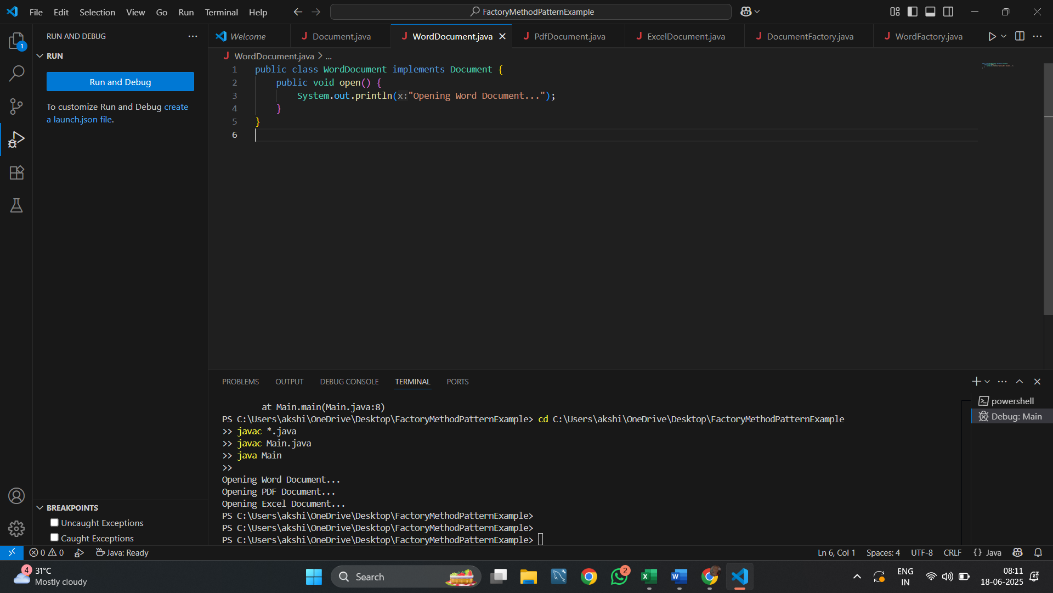
* Document.java
* WordDocument.java
* PdfDocument.java
* ExcelDocument.java
* DocumentFactory.java
* WordFactory.java
* PdfFactory.java
* ExcelFactory.java
* Main.java

Source Code:

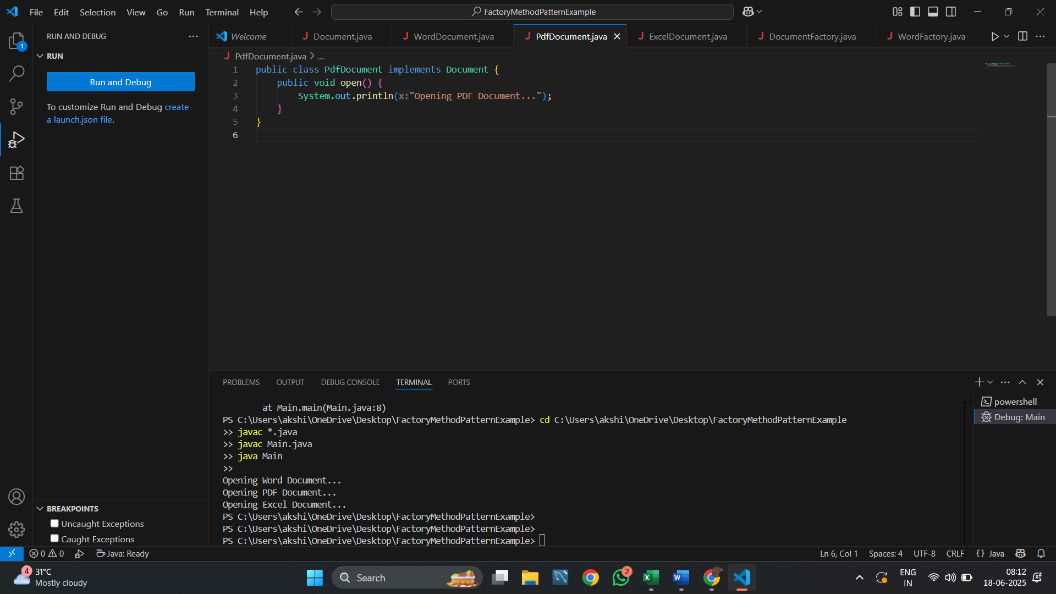
Document.java



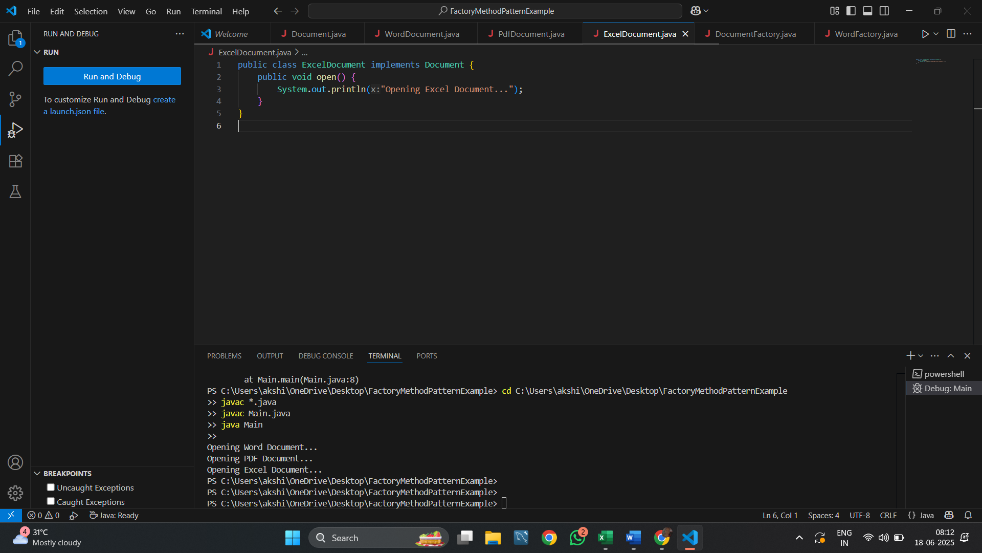
WordDocument.java



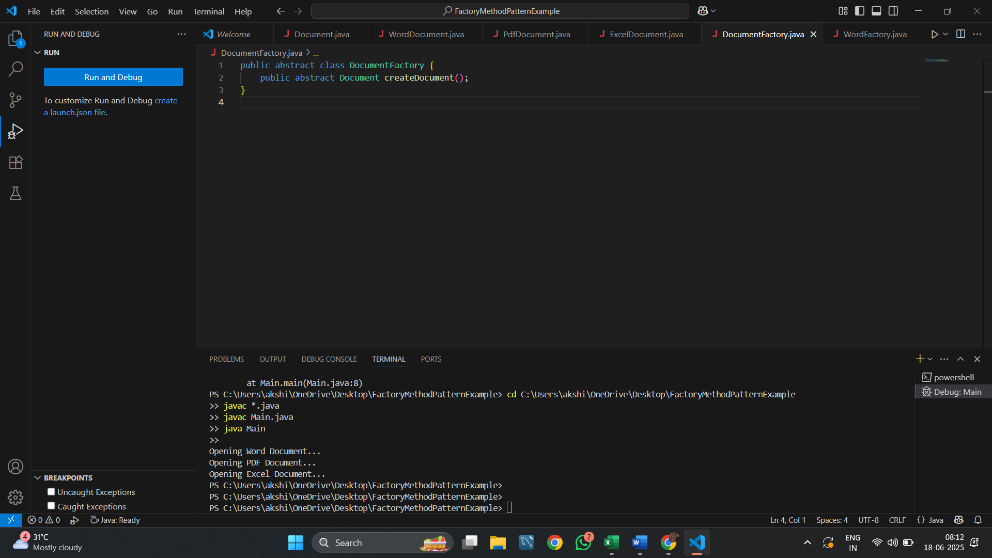
PdfDocument.java



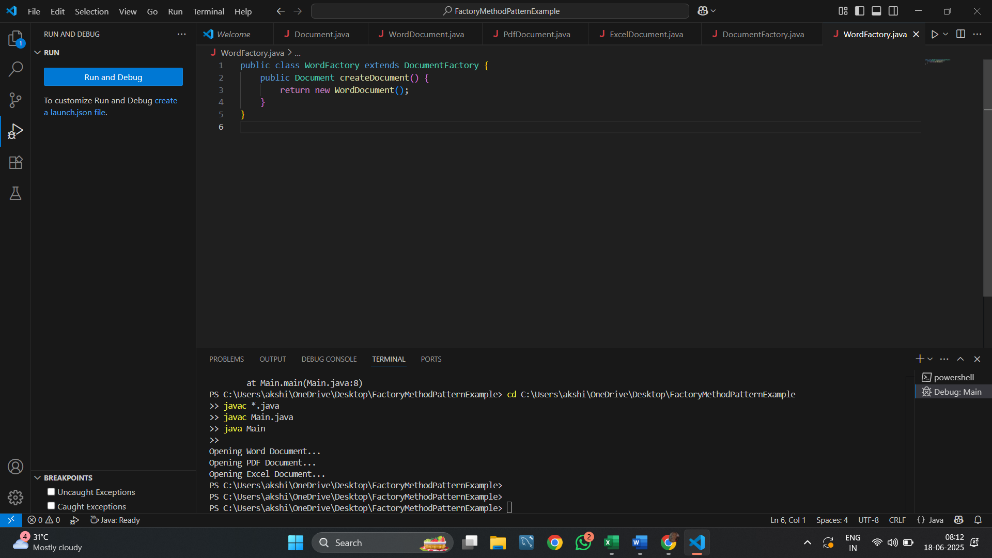
ExcelDocument.java



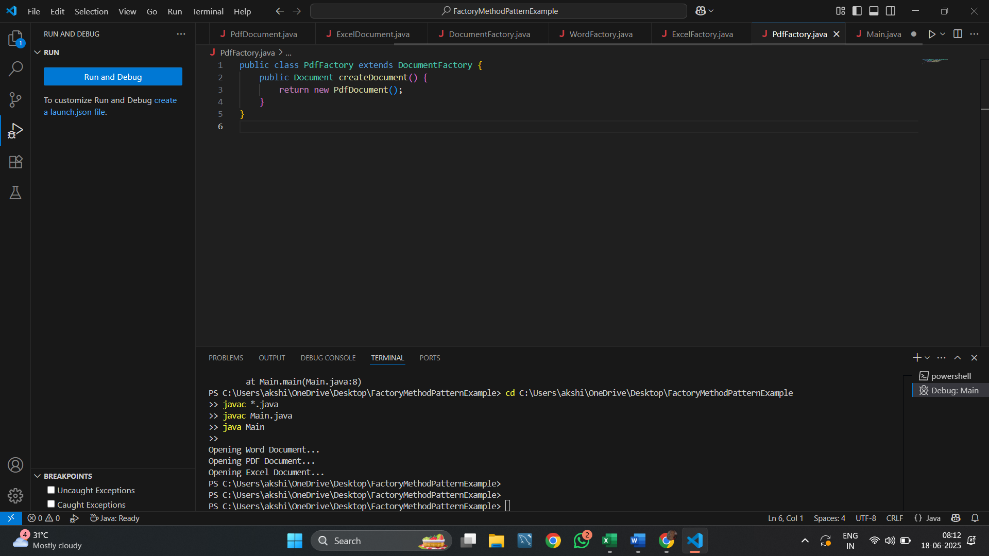
DocumentFactory.java



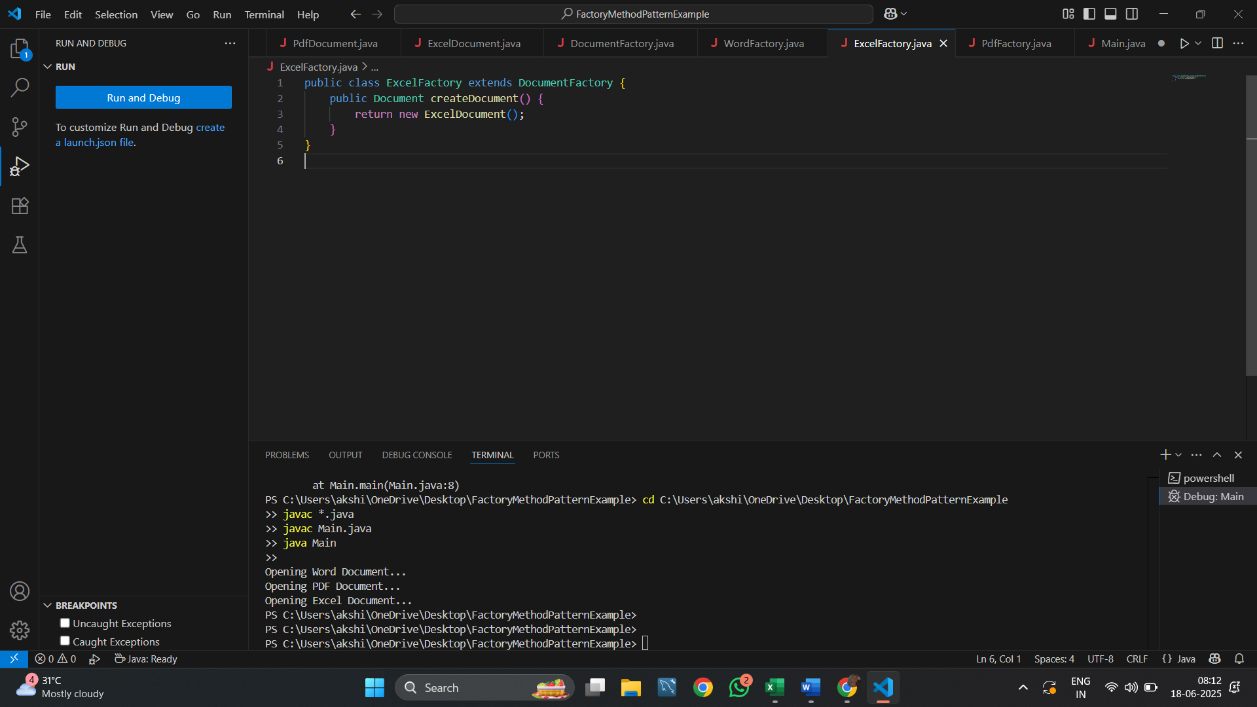
WordFactory.java



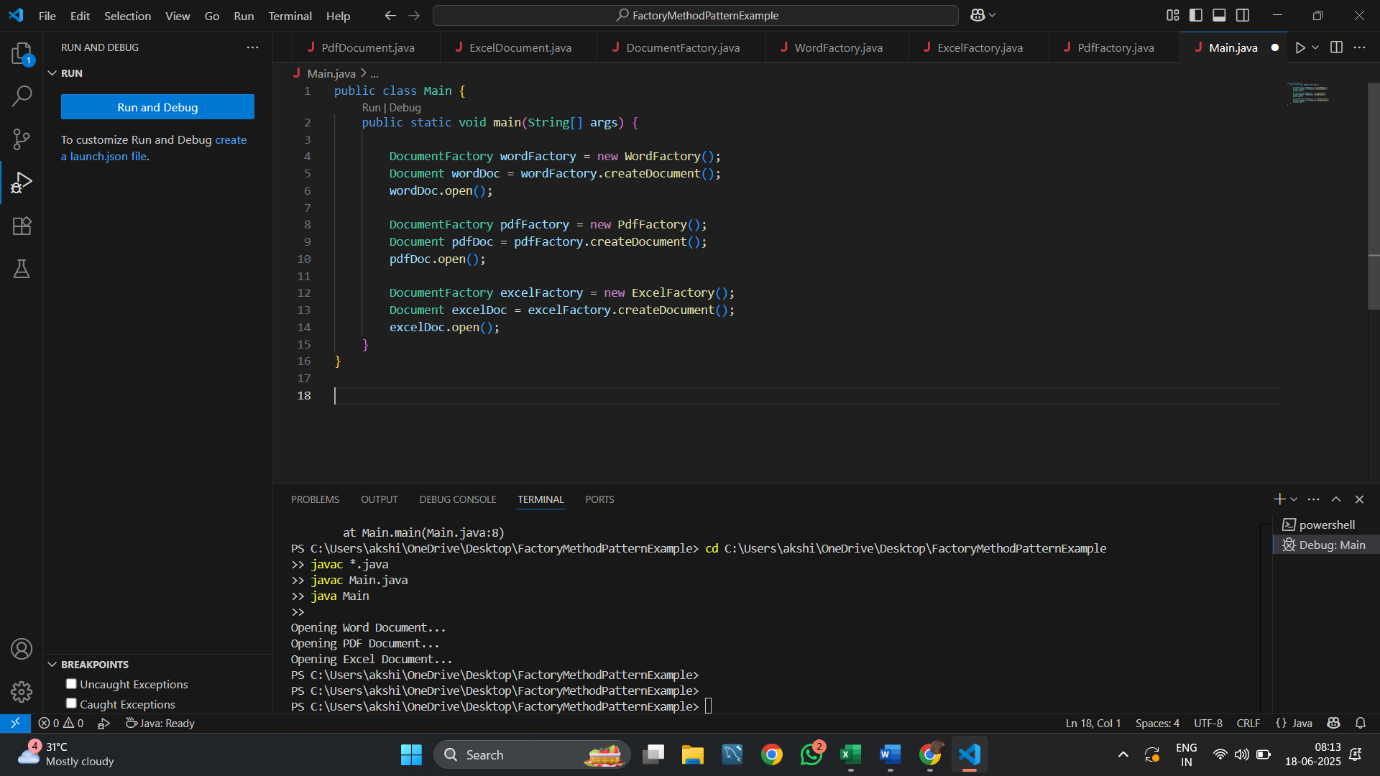
PdfFactory.java



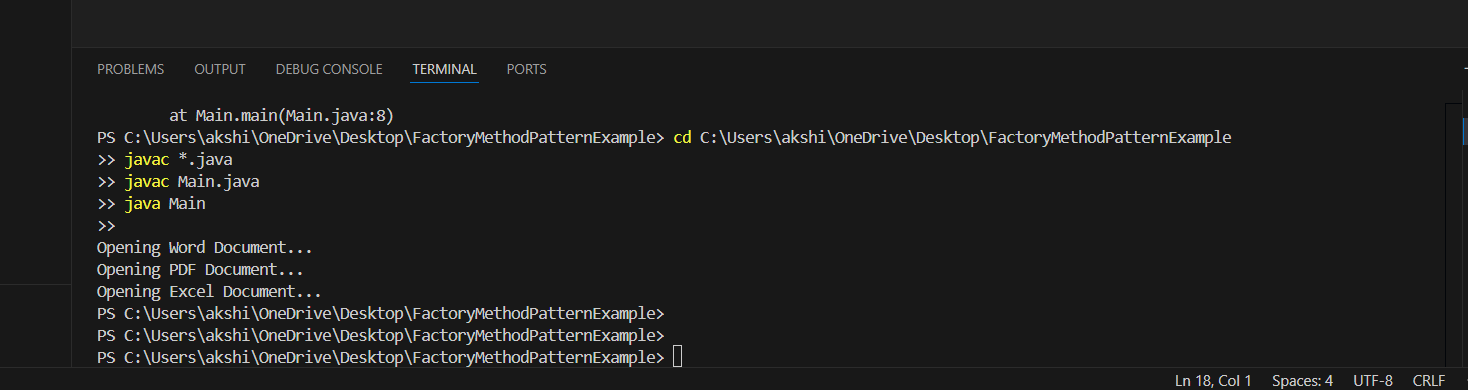
ExcelFactory.java



Main.java



Output :



Summary:

This assignment demonstrates the use of the Factory MethodPattern to dynamically create instances of different document types without altering client code. This pattern improves scalability, readability, and supports open/closed principle.

Assignment 3: Financial Forecasting

Objective: In this assignment, I developed a simple FinancialForecasting Tool using recursion in Java. The goal was to predict future values based on past financial growth, following the recursive programming technique.

Concept: Recursive Algorithms

Recursion is a method where a function calls itself to solve a smaller version of a problem. It's especially helpful in scenarios like tree traversal, factorial calculation, and also predictive modeling like this financial forecast.

Step-by-Step Implementation:

Step 1: Create Project Folder

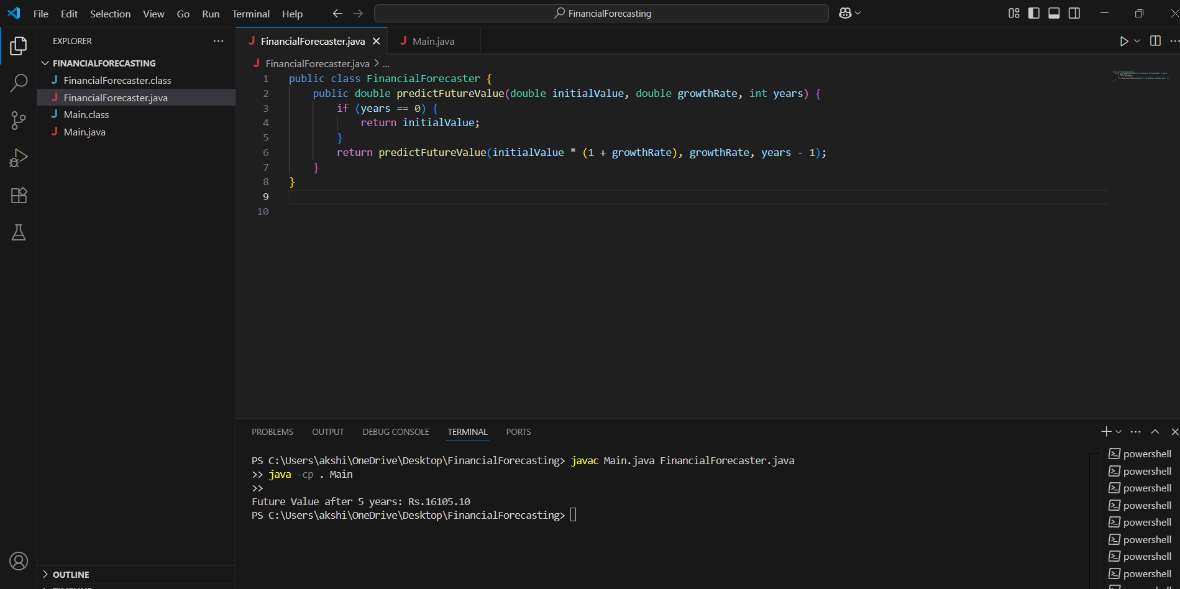
* I created a new folder on my desktop named FinancialForecasting

Step 2: Created the Java Files

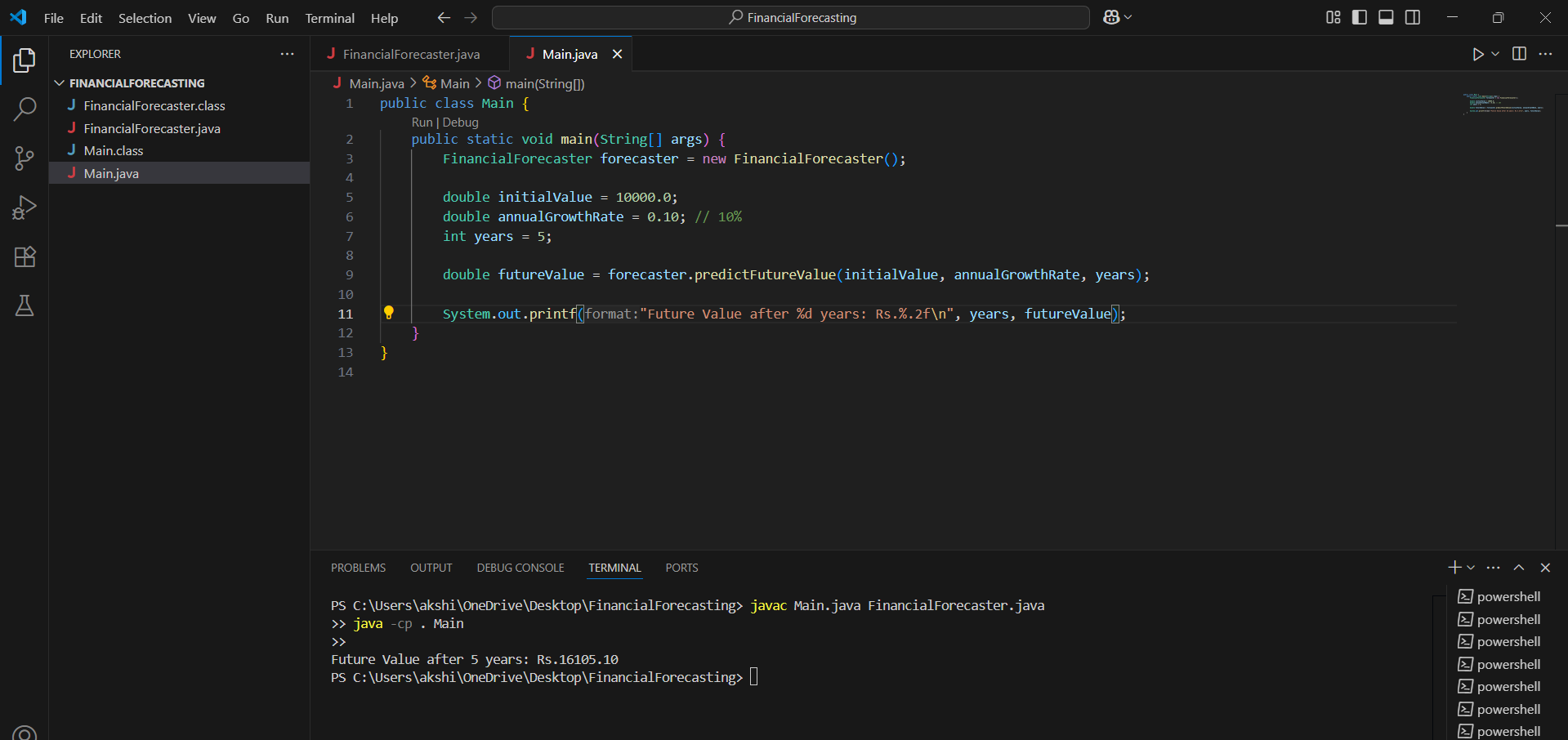
* FinancialForecaster.java – Logic for recursive calculation
* Main.java – Entry point to run the program

Source Code:

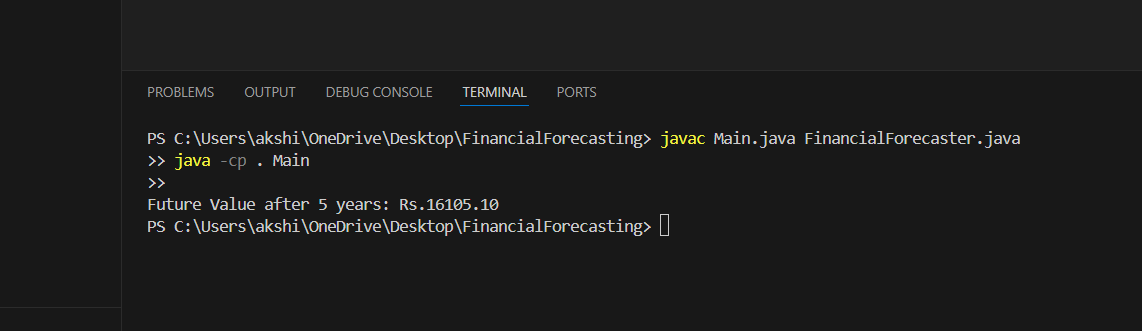
FinancialForecaster.java



Main.java



Output:



*Note:* I faced a small display issue where the “₹” symbol was showing incorrectly. I fixed it by replacing it with "Rs." in the output formatting.

Time Complexity Analysis

The recursive function makes one call per year, so the time complexity is:

O(n) where n = number of years

Optimization:

This recursive approach works well for small input sizes. However, for very large year values (e.g., hundreds or thousands), using an iterative loop would reduce stack usage and improve efficiency.

Assignment 4: E-commerce Platform Search Function

Objective: In this assignment, I implemented search functionality for an e-commerce platform using Linear Search and Binary Search. The aim was to understand search performance, analyze complexity, and compare which algorithm suits best in a real-world e-commerce system.

Concept: Asymptotic Notation

What is Big O Notation?

Big O notation describes how the runtime of an algorithm grows with the size of the input (n). It helps us analyze efficiency without needing to run the code.

* **Linear Search**: Scans every item one by one.
* **Binary Search**: Splits the sorted array in half each time to find the target.

Step-by-Step Implementation:

Step 1: Create Project Folder

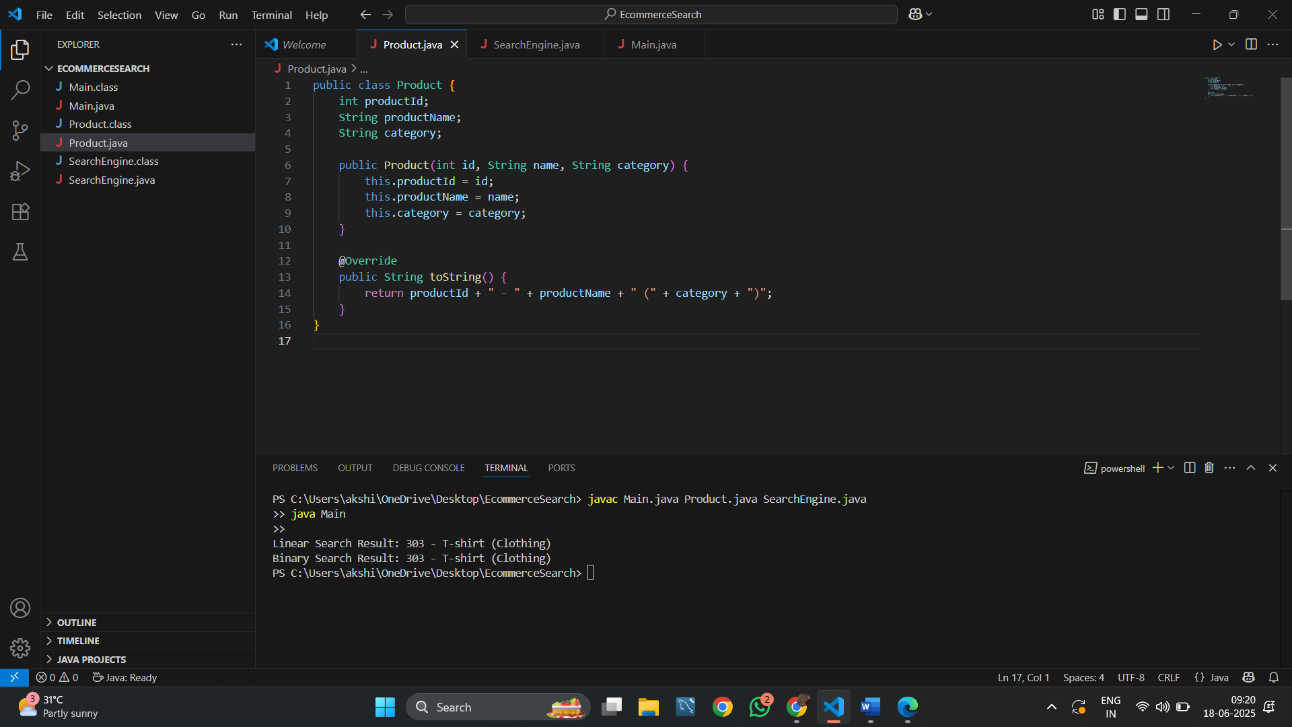
* Created a new Java project folder: EcommerceSearch.

Step 2: Create Java Files

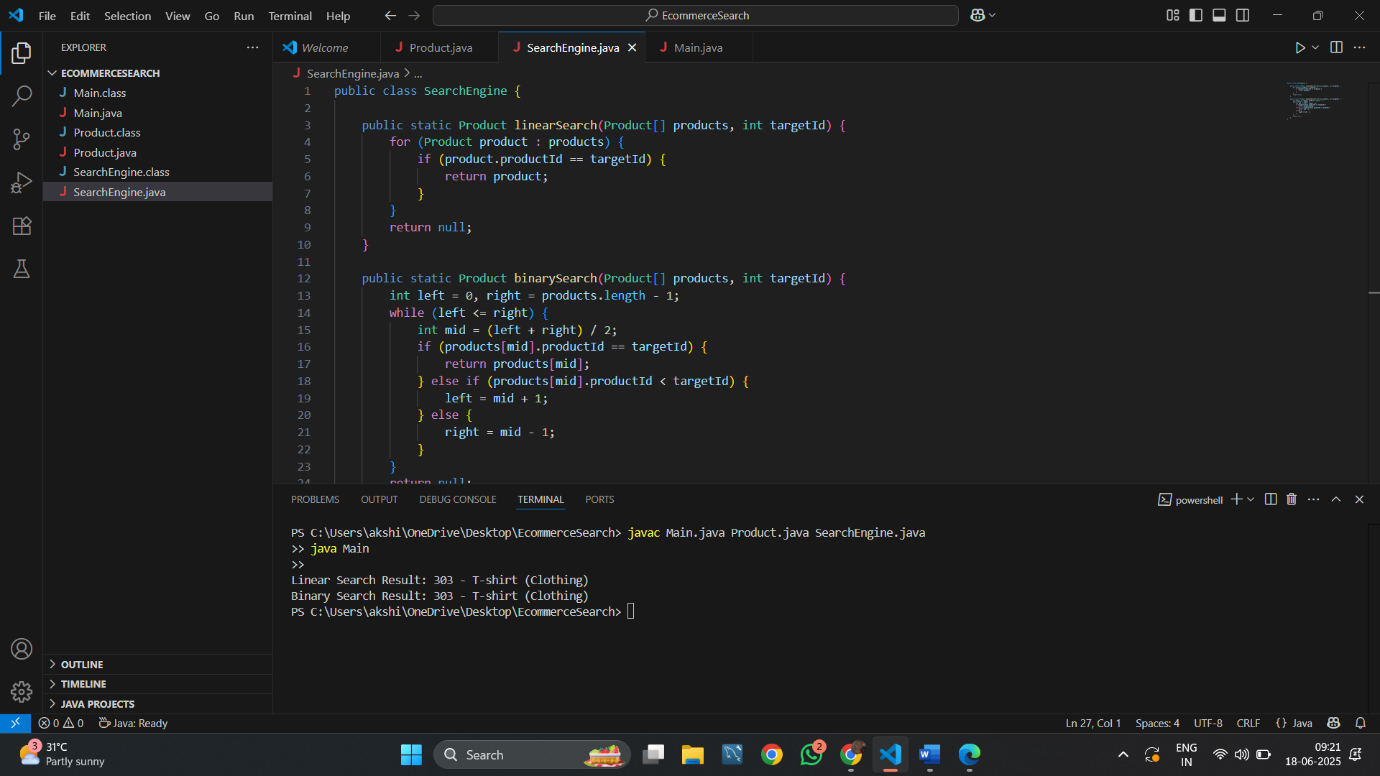
* Product.java – Blueprint for a product.
* SearchEngine.java – Implements both search methods.
* Main.java – Runs the search tests.

Source Code:

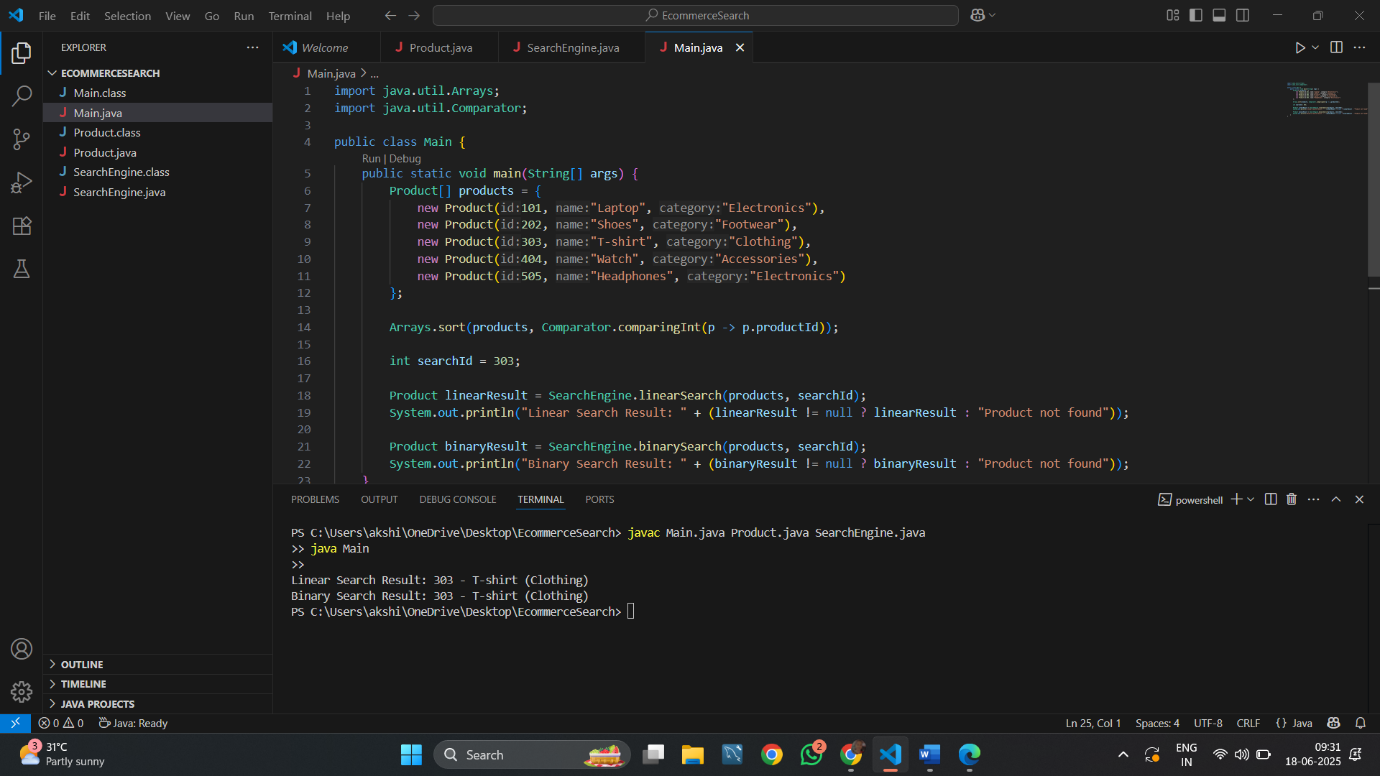
Product.java



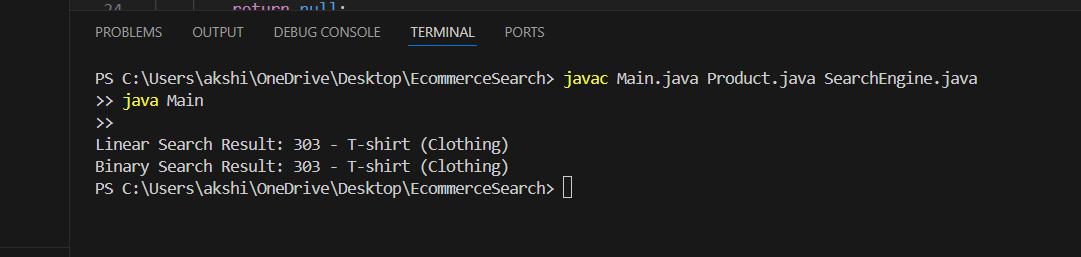
SearchEngine.java



Main.java



Output:



Observation: Binary search is significantly faster, especially for large sorted datasets — which is ideal for an e-commerce system where product listings can be huge.

Summary:

This assignment helped me understand and implement two basic search methods—linear search and binary search. I learned that binary search is much faster, but it only works when the data is sorted