Week – 1 - Engineering concepts

Module 1 - Design Patterns and Principles:

1.Implementing the Singleton Pattern:

Logger.java

public class Logger {

private static Logger instance;

private Logger() {

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("Log: " + message);

}

}

Main.java

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("First message");

logger2.log("Second message");

if (logger1 == logger2) {

System.out.println("Both logger instances are the same.");

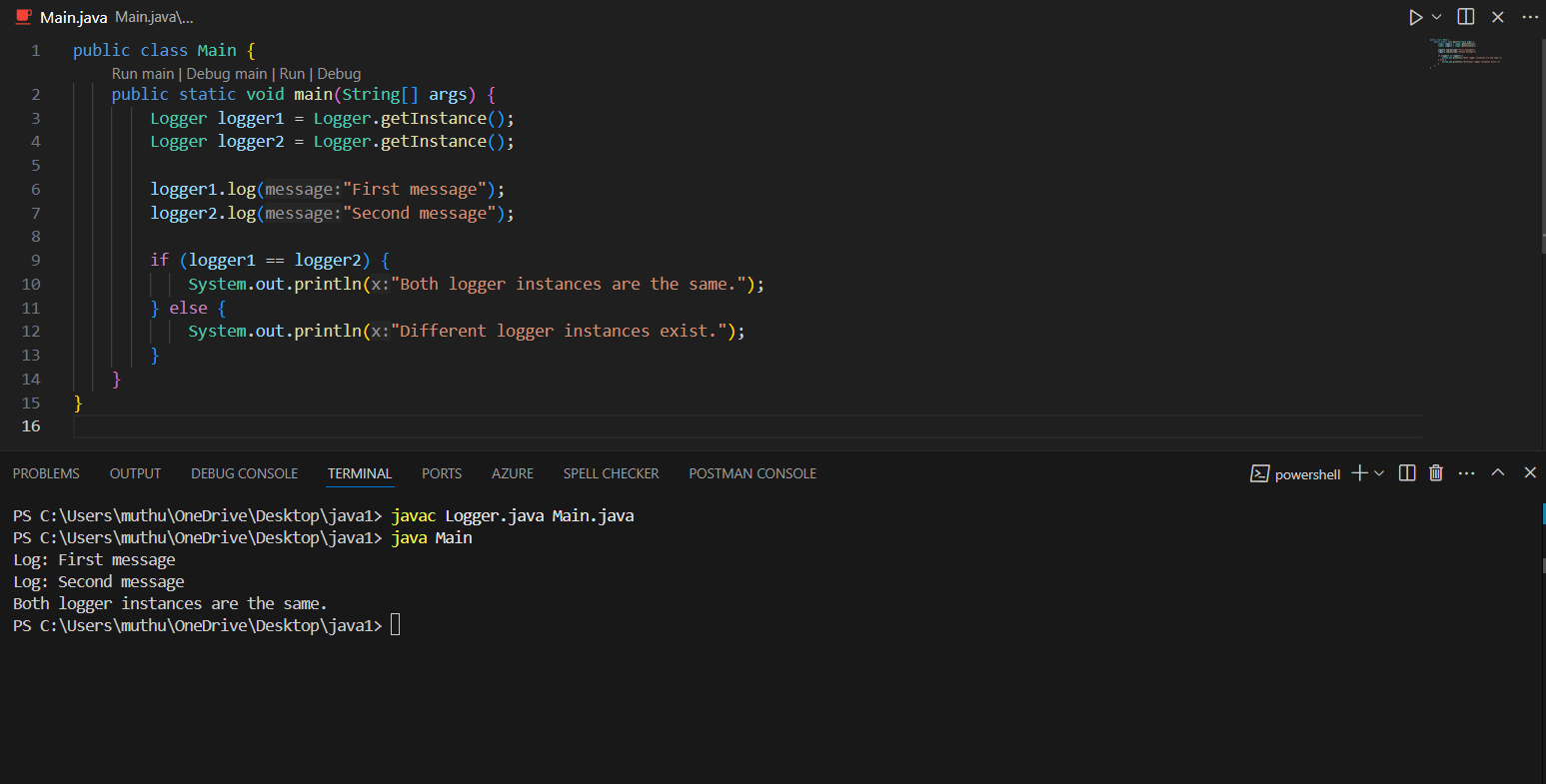
} else {

System.out.println("Different logger instances exist.");

}

}

}

Output:

2. Implementing the Factory Method Pattern:

Document.java

public interface Document {

void open();

}

WordDocument.java

public class WordDocument implements Document {

public void open() {

System.out.println("Opening Word Document...");

}

}

PdfDocument.java

public class PdfDocument implements Document {

public void open() {

System.out.println("Opening PDF Document...");

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document {

public void open() {

System.out.println("Opening Excel Document...");

}

}

DocumentFactory.java

public abstract class DocumentFactory {

public abstract Document createDocument();

}

WordDocumentFactory.java

public class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

**PdfDocumentFactory.java**

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

ExcelDocumentFactory.java

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

Main.java

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

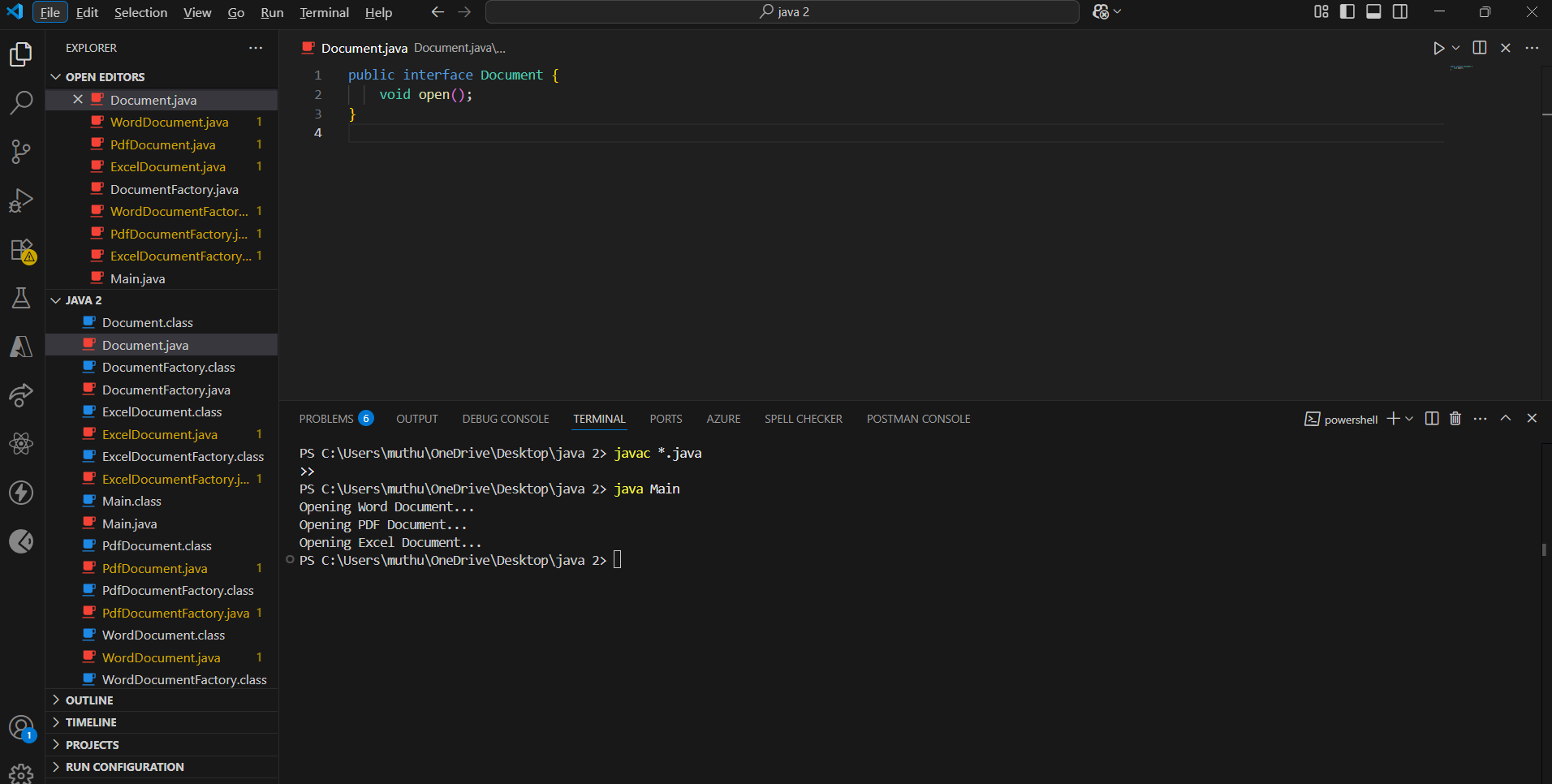
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

Output:



Module 2 - Data Structures and Algorithms

**1.E-commerce Platform Search Function:**

Product.java

public class Product {

int productId;

String productName;

String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

LinearSearch.java

public class LinearSearch {

public static Product search(Product[] products, int targetId) {

for (Product product : products) {

if (product.productId == targetId) {

return product;

}

}

return null;

}

}

BinarySearch.java

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

public static Product search(Product[] products, int targetId) {

Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (products[mid].productId == targetId) {

return products[mid];

} else if (products[mid].productId < targetId) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

}

Main.java

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(101, "Shoes", "Footwear"),

new Product(202, "T-shirt", "Clothing"),

new Product(303, "Laptop", "Electronics"),

new Product(404, "Book", "Stationery")

};

Product result1 = LinearSearch.search(products, 303);

System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Product not found"));

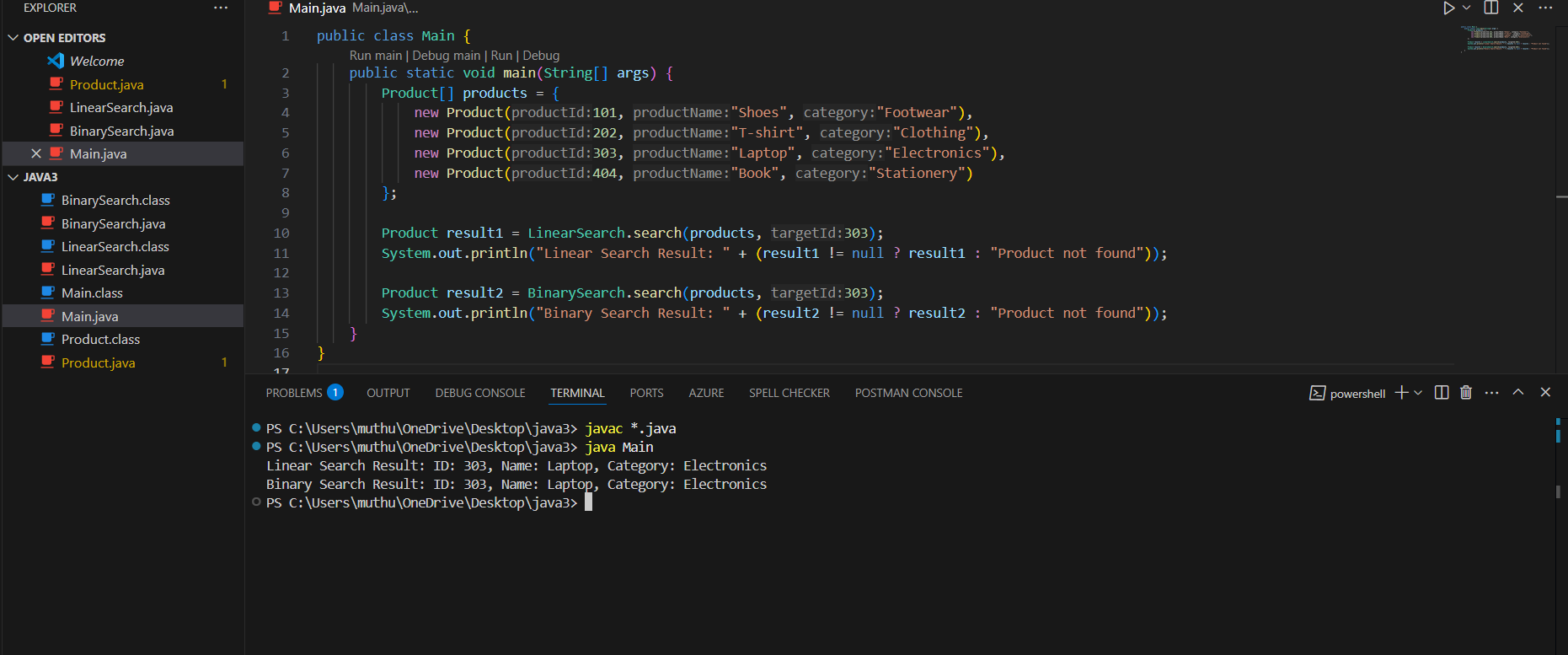
Product result2 = BinarySearch.search(products, 303);

System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Product not found"));

}

}

0utput:



2. Financial Forecasting

FinancialForecast.java

public class FinancialForecast {

public static double calculateFutureValue(double presentValue, double growthRate, int years) {

if (years == 0) {

return presentValue;

}

return calculateFutureValue(presentValue, growthRate, years - 1) \* (1 + growthRate);

}

public static void main(String[] args) {

double presentValue = 10000;

double annualGrowthRate = 0.10; // 10%

int years = 5;

double futureValue = calculateFutureValue(presentValue, annualGrowthRate, years);

System.out.printf("Future Value after %d years: ₹%.2f%n", years, futureValue);

}

}

Output:

