**Exercise 2: E-commerce Platform Search Function**

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Product**

public class Product {

int productId;

String productName;

String category;

public Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

@Override

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

**LinearSearch.java**

public class LinearSearch {

public static Product linearSearch(Product[] products, String targetName) {

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

}

**BinarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

public static Product binarySearch(Product[] products, String targetName) {

Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

int low = 0, high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = products[mid].productName.compareToIgnoreCase(targetName);

if (cmp == 0) {

return products[mid];

} else if (cmp < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

}

**TestSearch.java**

public class TestSearch {

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop", "Electronics"),

new Product(102, "Phone", "Electronics"),

new Product(103, "Shirt", "Clothing"),

new Product(104, "Shoes", "Footwear"),

new Product(105, "Watch", "Accessories")

};

// Linear Search Test

Product foundLinear = LinearSearch.linearSearch(products, "Shoes");

System.out.println("Linear Search Result: " + (foundLinear != null ? foundLinear : "Not Found"));

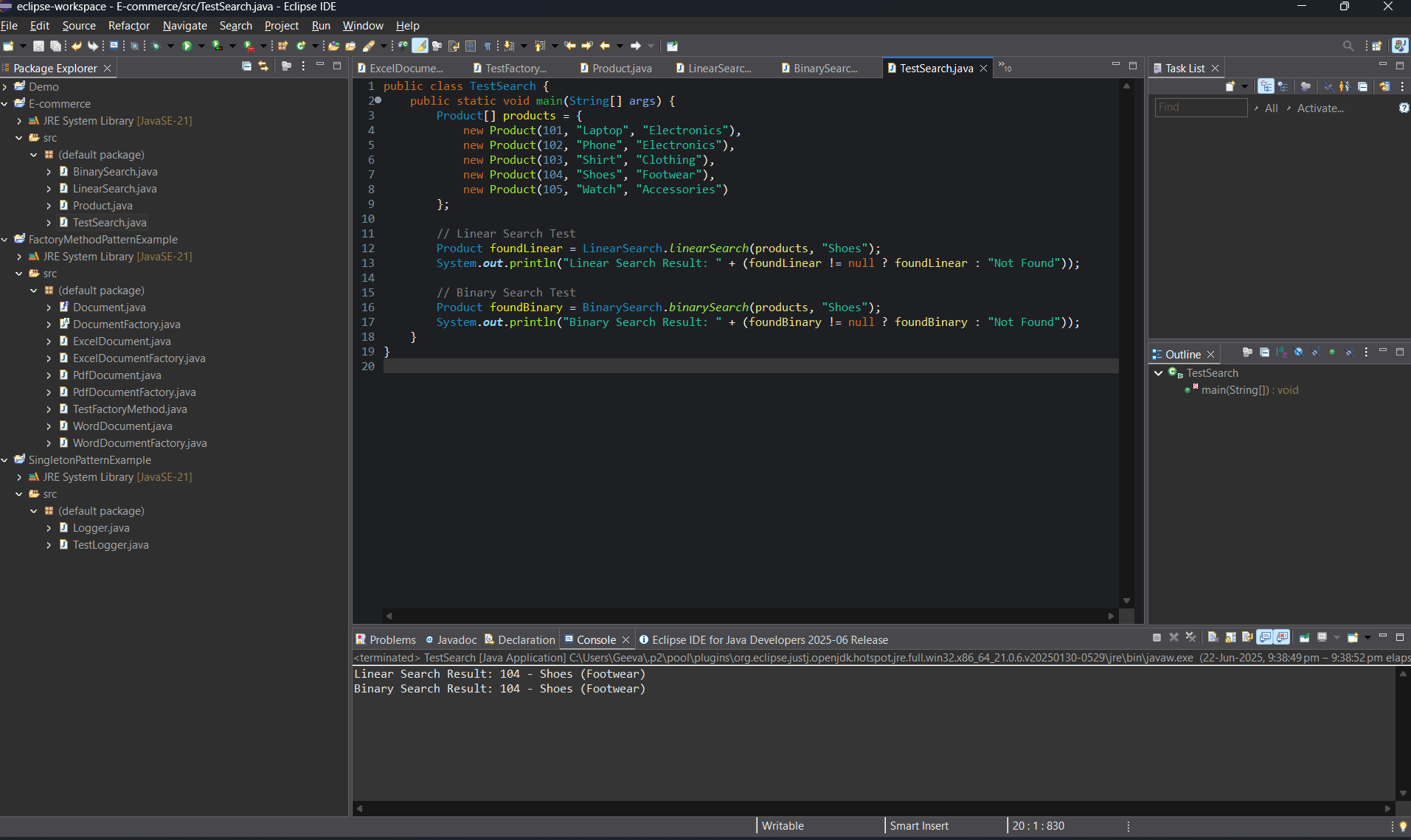
// Binary Search Test

Product foundBinary = BinarySearch.binarySearch(products, "Shoes");

System.out.println("Binary Search Result: " + (foundBinary != null ? foundBinary : "Not Found"));

}

}



**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**FinancialForecast.java**

public class FinancialForecast {

public static double futureValue(double principal, double rate, int years) {

if (years == 0) {

return principal;

}

return *futureValue*(principal, rate, years - 1) \* (1 + rate);

}

public static double futureValueMemo(double principal, double rate, int years, Double[] memo) {

if (years == 0) return principal;

if (memo[years] != null) return memo[years];

memo[years] = *futureValueMemo*(principal, rate, years - 1, memo) \* (1 + rate);

return memo[years];

}

public static void main(String[] args) {

double principal = 10000.0;

double rate = 0.05;

int years = 10;

double result = *futureValue*(principal, rate, years);

System.***out***.printf("Future Value (recursive): ₹%.2f%n", result);

Double[] memo = new Double[years + 1];

double memoizedResult = *futureValueMemo*(principal, rate, years, memo);

System.***out***.printf("Future Value (memoized): ₹%.2f%n", memoizedResult);

}

}

