**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.

**CODING:**

**Product.java**

package DsaAndAlgorithms;

public class Product {

int productId;

String productName;

String category;

public Product(int id,String productName,String category)

{

this.productId=id;

this.productName=productName;

this.category=category;

}

public String toString()

{

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

}

}

**SearchUtility.java**

package DsaAndAlgorithms;

public class SearchUtility {

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int result = name.compareToIgnoreCase(products[mid].productName);

if (result == 0) {

return products[mid];

} else if (result < 0) {

right = mid - 1;

} else {

left = mid + 1;

}}

return null;

}

}

**SearchTest.java**

package DsaAndAlgorithms;

import java.util.Arrays;

import java.util.Comparator;

public class SearchTest {

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Watch", "Accessories"),

new Product(103, "Laptop", "Electronics"),

new Product(104, "Bag", "Travel"),

new Product(105, "Phone", "Electronics")

};

String target = "Watch";

Product foundLinear = SearchUtility.linearSearch(products, target);

System.out.println("Linear Search Result:");

System.out.println(foundLinear != null ? foundLinear : "Product not found");

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

Product foundBinary = SearchUtility.binarySearch(products, target);

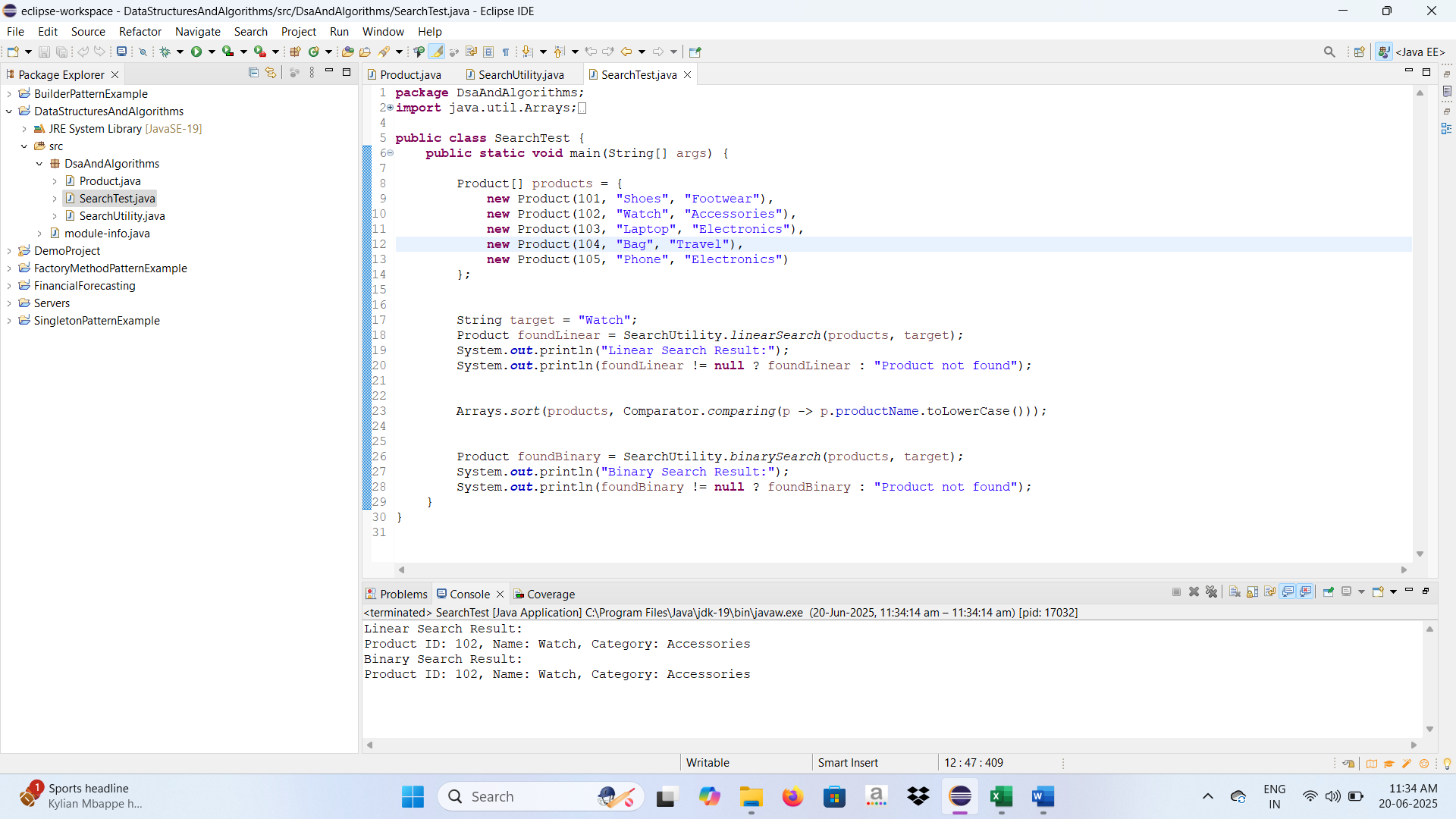
System.out.println("Binary Search Result:");

System.out.println(foundBinary != null ? foundBinary : "Product not found");

}

}

OUTPUT:



**Exercise 7: Financial Forecasting**

**Scenario:**

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

**CODING:**

**RecursionMethod.java**

package Recursion;

public class RecursionMethod {

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

if (years == 0) {

return presentValue; // Base case

}

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

public static double calculateFutureValueMemo(double presentValue, double growthRate, int years, Double[] memo) {

if (years == 0) return presentValue;

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

memo[years] = (1 + growthRate) \* *calculateFutureValueMemo*(presentValue, growthRate, years - 1, memo);

return memo[years];

}

public static void main(String[] args) {

double presentValue = 1000.0;

double growthRate = 0.05; // 5% annual growth

int years = 10;

double futureValue = *calculateFutureValue*(presentValue, growthRate, years);

System.*out*.printf("Future Value (Recursive): ₹%.2f\n", futureValue);

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

double optimizedValue = *calculateFutureValueMemo*(presentValue, growthRate, years, memo);

System.*out*.printf("Future Value (Memoized): ₹%.2f\n", optimizedValue);

}

}

**OUTPUT:**

