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

**Code:**

*Product.java:*

**package** mypackage;

**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;

}

}

*SearchDemo.java:*

**package** mypackage;

**import** java.util.\*;

**public** **class** SearchDemo {

**public** **static** **int** lSearch(Product[] products, String targetName) {//LINEAR SEARCH

**for** (**int** i = 0; i < products.length; i++) {

**if** (products[i].productName.equalsIgnoreCase(targetName)) {

**return** i;

}

}

**return** -1;

}

**public** **static** **int** bSearch(Product[] products, String targetName) {//BINARY SEARCH

**int** left = 0;

**int** right = products.length - 1;

**while** (left <= right) {

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

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

**if** (compare == 0)

**return** mid;

**else** **if** (compare < 0)

left = mid + 1;

**else**

right = mid - 1;

}

**return** -1;

}

**public** **static** **void** main(String[] args) {

Product[] products = {

**new** Product(1, "Shoes", "Fashion"),

**new** Product(2, "Laptop", "Electronics"),

**new** Product(3, "Watch", "Accessories"),

**new** Product(4, "Phone", "Electronics")

};

String searchName = "Phone";

**int** linearResult = *lSearch*(products, searchName);

System.***out***.println("Linear Search Result: " + (linearResult != -1 ? "Found at index " + linearResult : "Not Found"));

Arrays.*sort*(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

System.***out***.println("Sorted Products :");

**for**(Product p:products) {

System.***out***.println(p.productName);

}

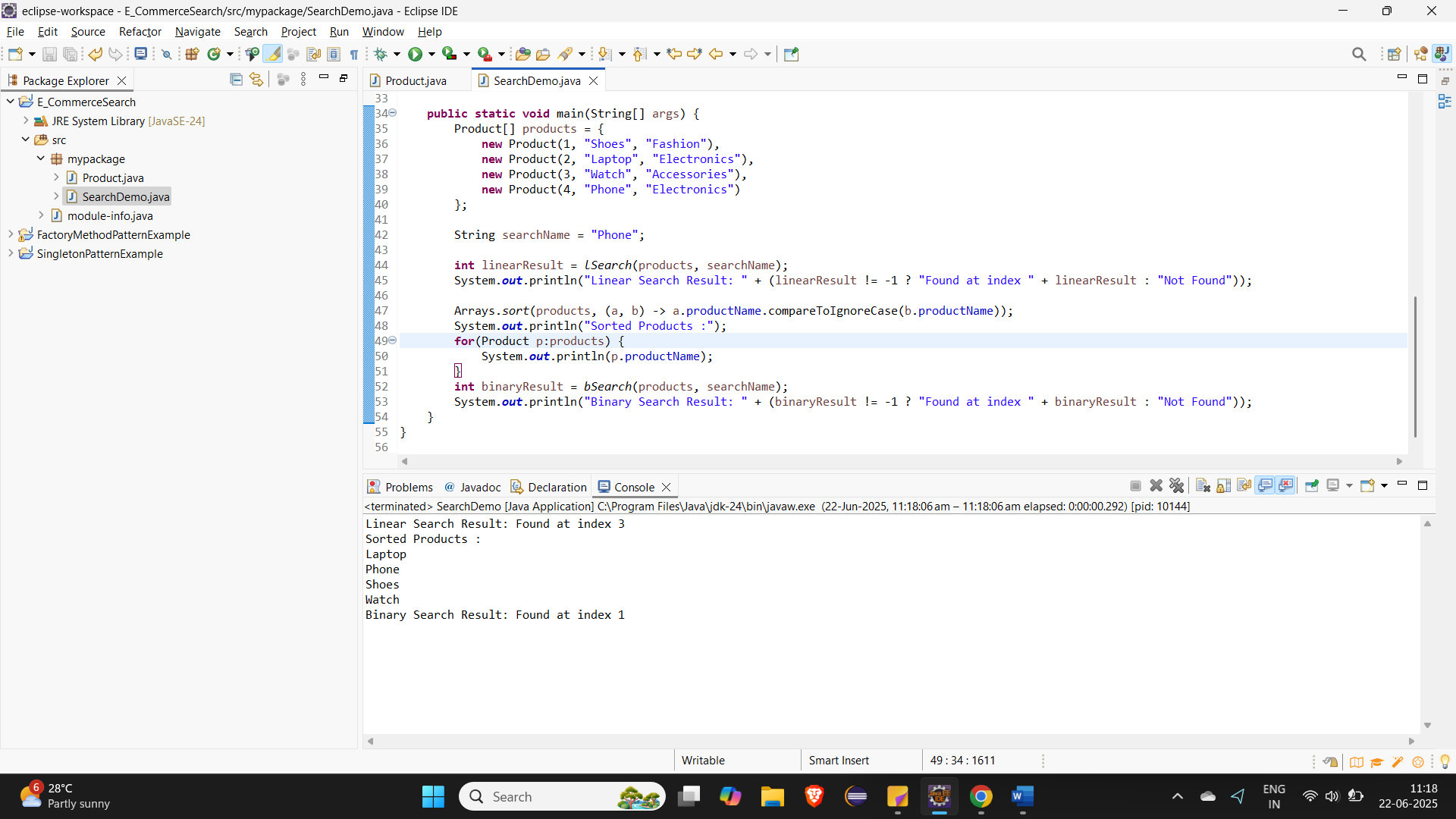
**int** binaryResult = *bSearch*(products, searchName);

System.***out***.println("Binary Search Result: " + (binaryResult != -1 ? "Found at index " + binaryResult : "Not Found"));

}

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**CODE:**

*FinalForecast.java:*

**package** mypackage;

**public** **class** FinancialForecast {

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

**if** (years == 0) {

**return** presentValue;

}

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

}

**public** **static** **double** calculateFutureValueIterative(**double** presentValue, **double** growthRate, **int** years) {

**double** futureValue = presentValue;

**for** (**int** i = 0; i < years; i++) {

futureValue \*= (1 + growthRate);

}

**return** futureValue;

}

**public** **static** **void** main(String[] args) {

**double** presentValue = 10000;

**double** growthRate = 0.65;

**int** years = 100;

**long** startTime, endTime;

startTime = System.*nanoTime*();

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

endTime = System.*nanoTime*();

System.***out***.println("Future Value (Recursive): " + futureValueRecursive+" took "+(endTime-startTime)/1000+"seconds");

startTime = System.*nanoTime*();

**double** futureValueIterative = *calculateFutureValueIterative*(presentValue, growthRate, years);

endTime = System.*nanoTime*();

System.***out***.println("Future Value (Iterative - Optimized): " + futureValueIterative+" took "+(endTime-startTime)/1000+"seconds");

}

}

**Output:**

