**Design Pattern and Principles**

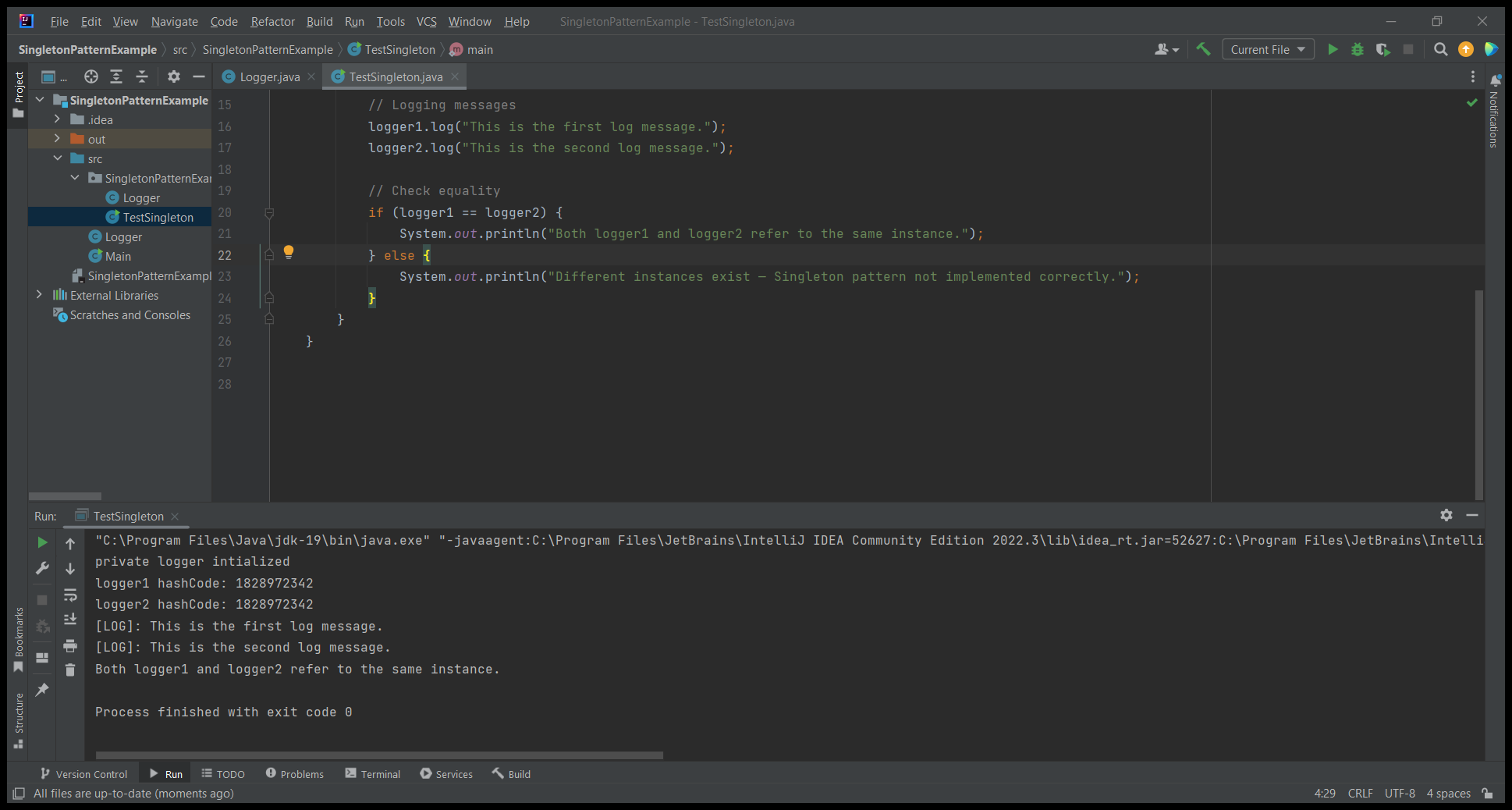
Exercise 1: Implementing the Singleton Pattern

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

package SingletonPatternExample;  
  
public class Logger {  
 private static Logger *instance*=null;  
  
 private Logger(){  
 System.*out*.println("private logger intialized");  
 }  
  
  
 public void log(String message) {  
 System.*out*.println("[LOG]: " + message);  
 }  
  
 public static Logger getInstance() {  
 if (*instance* == null) {  
 *instance* = new Logger();   
 }  
 return *instance*;  
 }  
  
   
  
  
}

package SingletonPatternExample;  
  
public class TestSingleton {  
  
 public static void main(String[] args) {  
  
  
 Logger logger1 = Logger.*getInstance*();  
 Logger logger2 = Logger.*getInstance*();  
  
  
 System.*out*.println("logger1 hashCode: " + logger1.hashCode());  
 System.*out*.println("logger2 hashCode: " + logger2.hashCode());  
  
  
 logger1.log("This is the first log message.");  
 logger2.log("This is the second log message.");  
  
   
 if (logger1 == logger2) {  
 System.*out*.println("Both logger1 and logger2 refer to the same instance.");  
 } else {  
 System.*out*.println("Different instances exist — Singleton pattern not implemented correctly.");  
 }  
 }  
 }



Exercise 2: Implementing the Factory Method Pattern

**Scenario:**

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

1.Document Interface

package com.example.documents;  
  
public interface Document {  
 void open();  
}

2.WordDocument.java

package com.example.documents;  
  
public class WordDocument implements Document {  
 public void open()  
 {  
 System.*out*.println("opening Word Document...");  
 }  
}

3.PdfDocument.java

package com.example.documents;  
  
public class PDFDocument implements Document  
i{  
 public void open()  
 {  
 System.*out*.println("opening PDF Document...");  
 }  
}

4.ExcelDocument.java

package com.example.documents;  
  
public class ExcelDocument implements Document {  
 public void open()  
 {  
 System.*out*.println("opening Excel Document...");  
 }  
}

5. WordDocumentFactory.java

package com.example.documents;  
  
public class WordDocumentFactory extends DocumentFactory{  
  
  
  
  
 @Override  
 public Document cretaeDocument() {  
 return new WordDocument();  
 }  
}

6. PdfDocumentFactory.java

package com.example.documents;  
  
public class PdfDocumentFactory extends DocumentFactory {  
  
  
  
 @Override  
 public Document cretaeDocument() {  
 return new PDFDocument();  
 }  
}

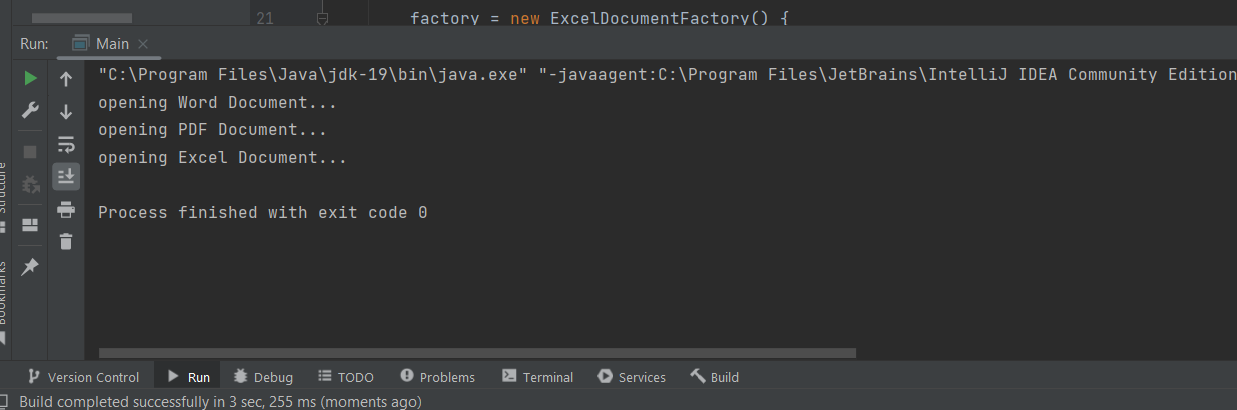
7.ExcelDocumentFactory.java

package com.example.documents;  
  
public class ExcelDocumentFactory extends DocumentFactory {  
  
  
 @Override  
 public Document cretaeDocument() {  
 return new ExcelDocument();  
 }  
}

8.Main.java

package com.example.documents;  
  
public class Main {  
 public static void main(String[]args)  
 {  
 DocumentFactory factory;  
  
 //create word document and open it  
 factory= new WordDocumentFactory() {  
  
 };  
 Document word=factory.cretaeDocument();  
 word.open();  
  
 factory = new PdfDocumentFactory() {  
  
 };  
 Document pdf=factory.cretaeDocument();  
 pdf.open();  
  
 factory = new ExcelDocumentFactory() {  
  
 };  
 Document excel=factory.cretaeDocument();  
 excel.open();  
  
 }  
}

OUTPUT:



**DSA \_Mandatory\_Exercises**

Exercise 1..E-CommercePlatformSearch

Code:

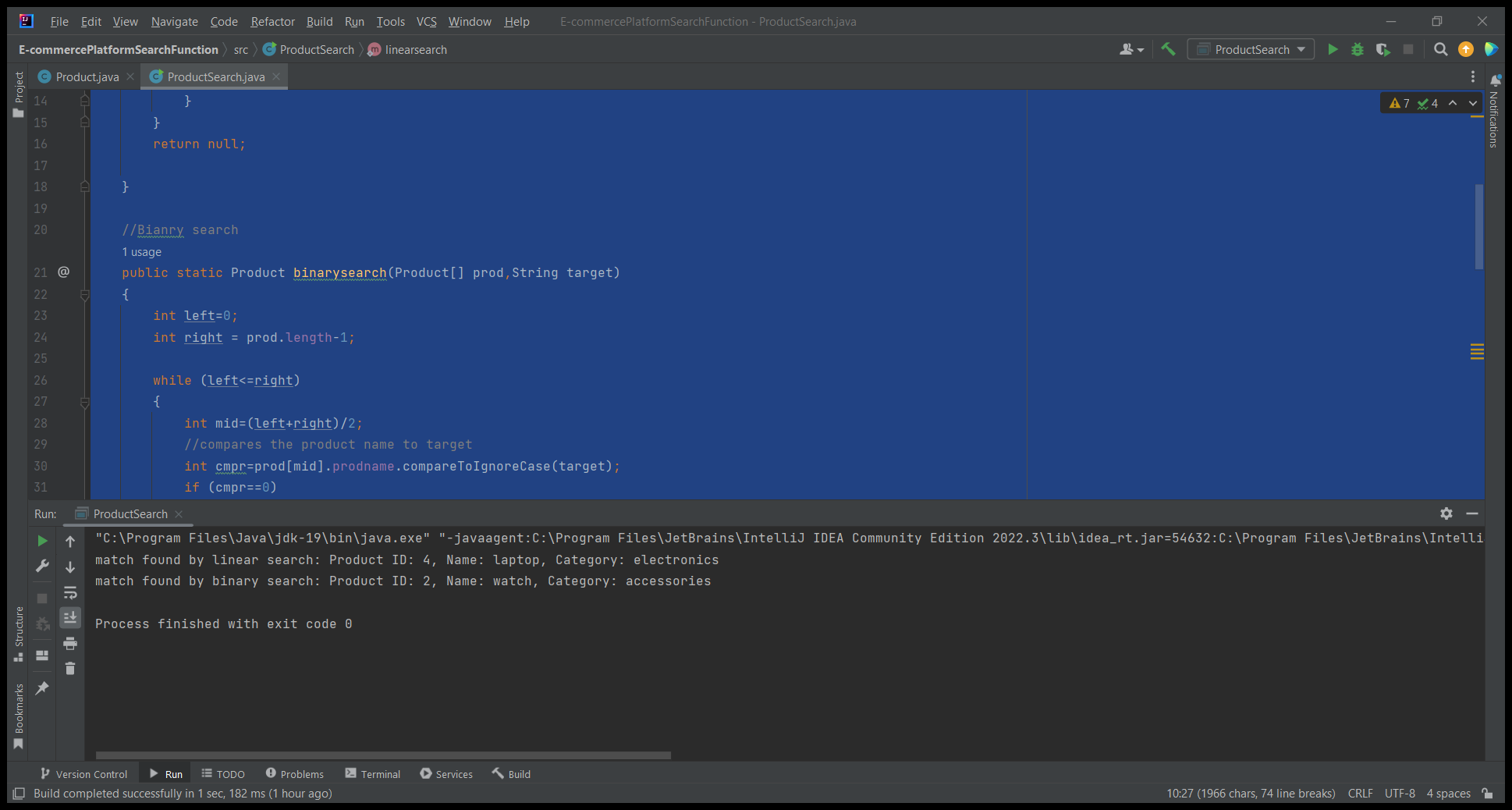
1.Product Class

public class Product {  
 int productID;  
 String prodname;  
 String prodcategory;  
 public Product(int productID, String prodname, String prodcategory)  
 {  
 this.productID=productID;  
 this.prodname=prodname;  
 this.prodcategory=prodcategory;  
 }  
  
 @Override  
 public String toString() {  
 return "Product ID: " + productID + ", Name: " + prodname + ", Category: " + prodcategory;  
 }  
}

2. ProductSearch.java

import java.lang.reflect.Array;  
import java.util.Arrays;  
  
public class ProductSearch {  
 //linear search  
 public static Product linearsearch(Product[] prod,String target)  
 {  
 for(Product p:prod)  
 {  
 //checks target with product name in the array if equal or not  
 if(p.prodname.equalsIgnoreCase(target))  
 {  
 return p;  
 }  
 }  
 return null;  
  
 }  
  
 //Bianry search  
 public static Product binarysearch(Product[] prod,String target)  
 {  
 int left=0;  
 int right = prod.length-1;  
   
 while (left<=right)  
 {  
 int mid=(left+right)/2;  
 //compares the product name to target  
 int cmpr=prod[mid].prodname.compareToIgnoreCase(target);  
 if (cmpr==0)  
 {  
 return prod[mid]; //match found strings are equal  
 } else if (cmpr<0) {  
 left=mid+1; //search in right half  
 }  
 else {  
 right=mid-1; //search in left half  
 }  
 }  
 return null;  
   
 }  
  
 public static void main(String[] args) {  
 Product [] p={  
 new Product(001,"shirt","clothing"),  
 new Product(002,"watch","accessories"),  
 new Product(003,"sauce","food"),  
 new Product(004,"laptop","electronics")  
  
 };  
  
 Product res1=*linearsearch*(p,"laptop");  
 if(res1!=null){  
 System.*out*.println("match found by linear search"+": "+res1.toString());  
 }  
 else {  
 System.*out*.println("not found");  
 }  
  
 Arrays.*sort*(p,(p1,p2) ->p1.prodname.compareToIgnoreCase(p2.prodname));  
 Product res2=*binarysearch*(p,"watch");  
 if(res2!=null)  
 {  
 System.*out*.println("match found by binary search"+": "+res2.toString());  
 }  
 else{  
 System.*out*.println("not found");  
  
 }  
 }  
  
}

OUTPUT:



**Exercise 7: Financial Forecasting**

**Scenario:**

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

**CODE:**  
public class FinancialForecast {  
 public static double forecastRecursive(double presentValue, double growthRate, int years) {  
 if (years == 0) {  
 return presentValue; // Base case  
 }  
 return *forecastRecursive*(presentValue, growthRate, years - 1) \* (1 + growthRate);  
 }  
  
 public static void main(String[] args) {  
 double initialValue = 10000;  
 double growthRate = 0.04; // 8% annual growth  
 int years = 8; // Predict 5 years into future  
  
 System.*out*.println("Financial Forecast: ");  
  
 double forecastRec = *forecastRecursive*(initialValue, growthRate, years);  
 System.*out*.printf("Recursive Forecast after %d years: ₹%.2f%n", years, forecastRec);  
  
 }  
}

OUTPUT:

