**WEEK 1**

**Design Patterns, Data Structures & Algorithms**

**Name:- Tadepalli Tarun**

**Superset ID:- 4991850**

1. **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.

Logger.Java:-

package singleton;

public class Logger {

private static Logger *singleInstance*;

private Logger() {

System.***out***.println("Logger instance created.");

}

public static Logger getInstance() {

if (*singleInstance* == null) {

*singleInstance* = new Logger();

}

return *singleInstance*;

}

public void log(String message) {

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

}

}

Main.java:

package singleton;

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

logger1.log("The First Message:- Hi Iam Tarun...");

logger2.log("The Second message: Iam From KL University. ");

if (logger1 == logger2) {

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

} else {

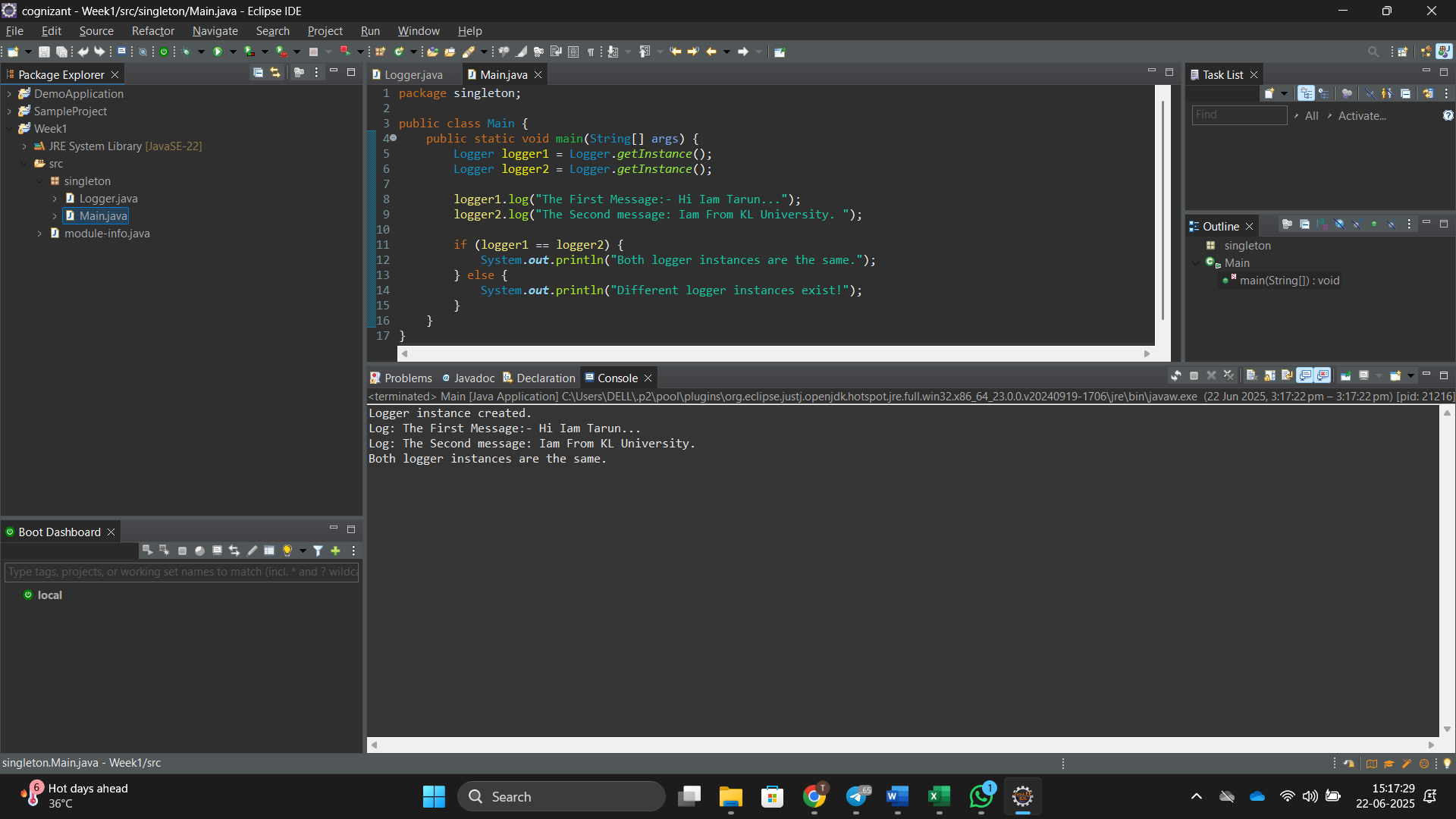
System.***out***.println("Different logger instances exist!");

}

}

}

OUTPUT:-



1. 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.

**Document.java: -**

**package** factory;

**public** **interface** Document {

**void** open();

}

**DocumentFactory.java: -**

**package** factory;

**public** **abstract** **class** DocumentFactory {

**public** **abstract** Document createDocument();

}

**ExcelDocument.java: -**

**package** factory;

**public** **class** ExcelDocument **implements** Document {

**public** **void** open() {

System.***out***.println("Opening an Excel document.");

}

}

**ExcelDocumentFactory.java: -**

**package** factory;

**public** **class** ExcelDocumentFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** ExcelDocument();

}

}

**PdfDocument.java: -**

**package** factory;

**public** **class** PdfDocument **implements** Document {

**public** **void** open() {

System.***out***.println("Opening a PDF document.");

}

}

**PdfDocumentFactory.java: -**

**package** factory;

**public** **class** PdfDocumentFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** PdfDocument();

}

}

**WordDocument.java: -**

**package** factory;

**public** **class** WordDocument **implements** Document {

**public** **void** open() {

System.***out***.println("Opening a Word document.");

}

}

**WordDocumentFactory.java: -**

**package** factory;

**public** **class** WordDocumentFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** WordDocument();

}

}

**Main.java: -**

**package** factory;

**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();

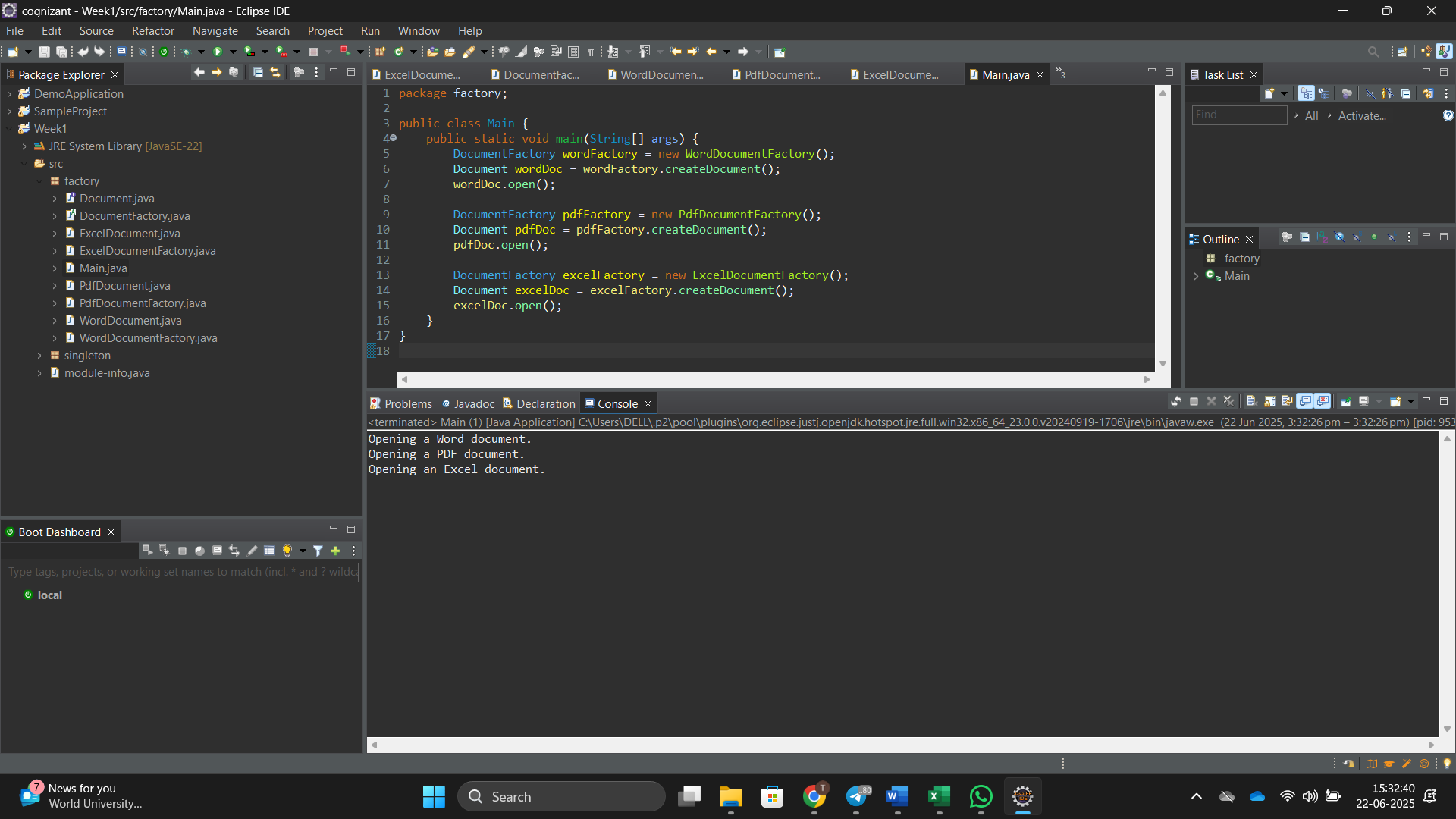
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:-**



1. 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.Java:-**

package E\_Commmerece\_Search;

public class Product {

int productId;

String productName, category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return "[" + productId + ", " + productName + ", " + category + "]";

}

}

**Searchfunc.java:-**

package E\_Commmerece\_Search;

import java.util.\*;

public class SearchFunc {

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

if (cmp == 0) return products[mid];

else if (cmp < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

public static void sortProducts(Product[] products) {

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

}

}

**Main.java:-**

package E\_Commmerece\_Search;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

Product[] products = {

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

new Product(102, "Shoes", "Footwear"),

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

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

};

System.***out***.print("Enter product name to search: ");

String inputProduct = sc.nextLine();

Product result1 = SearchFunc.*linearSearch*(products, inputProduct);

if (result1 != null) {

System.***out***.println("Product found using Linear Search: " + result1);

} else {

System.***out***.println("Product not found using Linear Search.");

}

SearchFunc.*sortProducts*(products);

Product result2 = SearchFunc.*binarySearch*(products, inputProduct);

if (result2 != null) {

System.***out***.println("Product found using Binary Search: " + result2);

} else {

System.***out***.println("Product not found using Binary Search.");

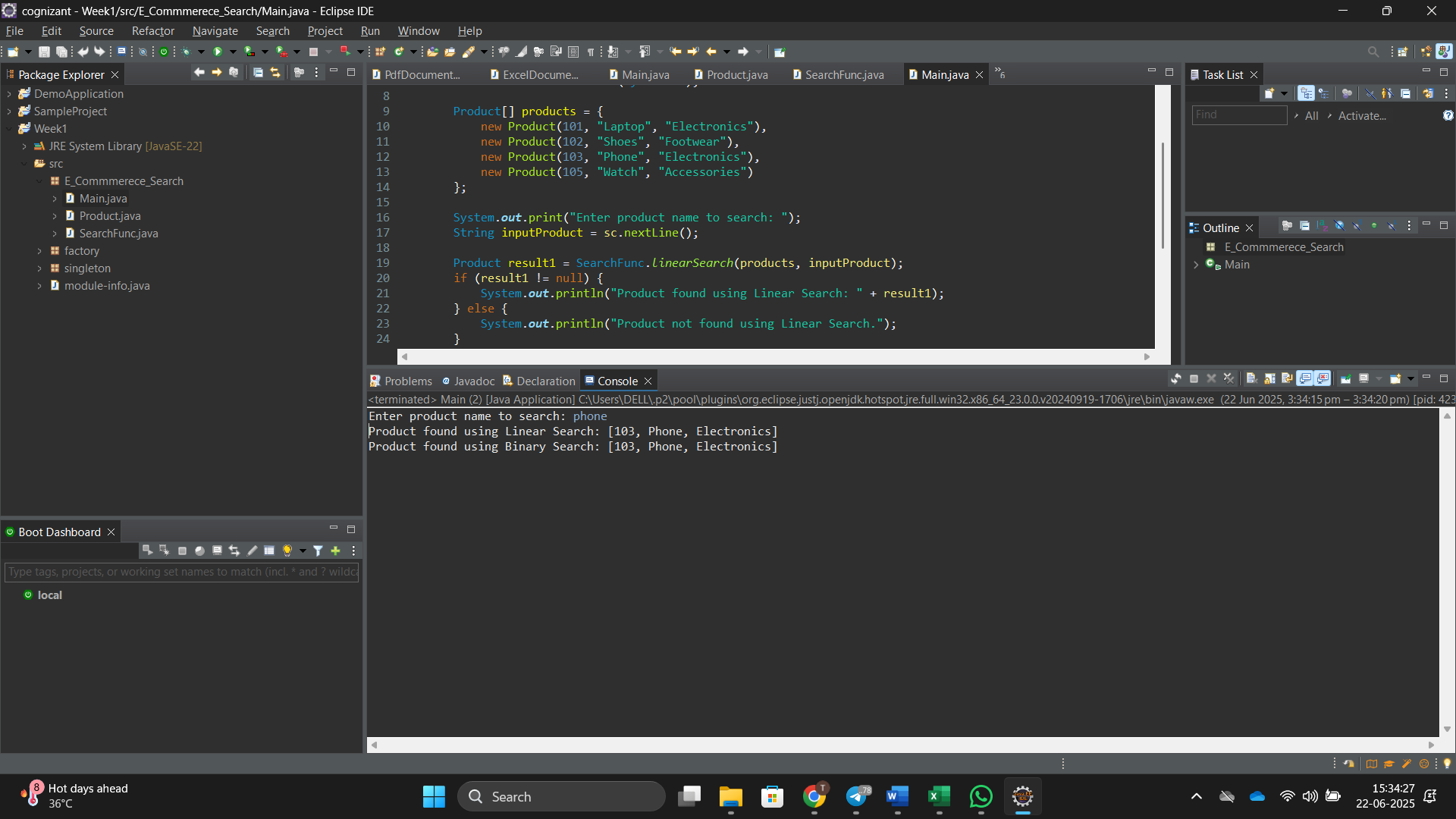
}

sc.close();

}

}

**Output:-**



1. Financial Forecasting

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

***Forecast.java:-***

package Financial\_Forecasting;

import java.util.Scanner;

public class Forecast {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter current value: ");

double currentValue = sc.nextDouble();

System.***out***.print("Enter annual growth rate (e.g., 0.05 for 5%): ");

double growthRate = sc.nextDouble();

System.***out***.print("Enter number of years to forecast: ");

int years = sc.nextInt();

double futureValue = Forecast.*calculateFutureValue*(currentValue, growthRate, years);

System.***out***.printf("Predicted future value after %d years: %.2f\n", years, futureValue);

sc.close();

}

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

if (years == 0) {

return currentValue;

}

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

}

}

**Output:-**

