**NAME OF THE QUESTION:**

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

**CODE:**

**Product.java**

public class Product {

private int productId;

private String productName;

private String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public String getCategory() {

return category;

}

@Override

public String toString() {

return "ProductID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

**SearchService.java**

import java.util.Arrays;

import java.util.Comparator;

public class SearchService {

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

int left = 0;

int right = products.length - 1;

while (left <= right) {

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

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

if (result == 0) {

return products[mid];

} else if (result > 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void sortProductsByName(Product[] products) {

Arrays.sort(products, Comparator.comparing(Product::getProductName, String.CASE\_INSENSITIVE\_ORDER));

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shampoo", "Personal Care"),

new Product(3, "Notebook", "Stationery"),

new Product(4, "Headphones", "Electronics"),

new Product(5, "Chair", "Furniture")

};

System.out.println("------ Linear Search ------");

Product result1 = SearchService.linearSearch(products, "Notebook");

if (result1 != null) {

System.out.println("Product found: " + result1);

} else {

System.out.println("Product not found.");

}

System.out.println("\n------ Binary Search ------");

SearchService.sortProductsByName(products);

Product result2 = SearchService.binarySearch(products, "Notebook");

if (result2 != null) {

System.out.println("Product found: " + result2);

} else {

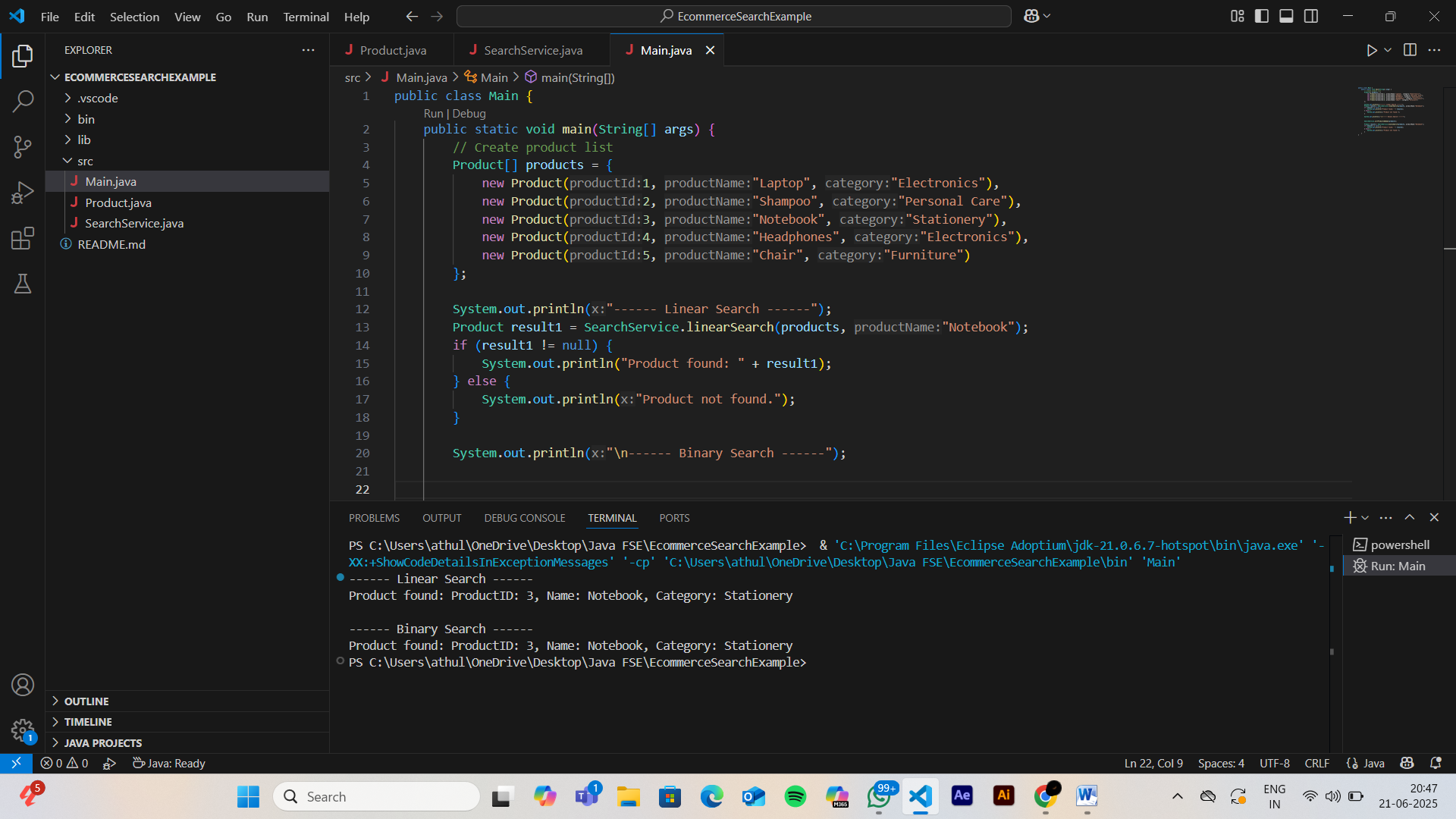
System.out.println("Product not found.");

}

}

}

**OUTPUT:**



**NAME OF THE QUESTION:**

**Exercise 7:** Financial Forecasting

**CODE:**

**FinancialForecaster.java**

public class FinancialForecaster {

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

if (years == 0) {

return currentValue;

} else {

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

}

}

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

double result = currentValue;

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

result \*= (1 + growthRate);

}

return result;

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

double currentValue = 1000.0;

double growthRate = 0.05;

int years = 10;

double futureValueRecursive = FinancialForecaster.calculateFutureValue(currentValue, growthRate, years);

double futureValueIterative = FinancialForecaster.calculateFutureValueIterative(currentValue, growthRate, years);

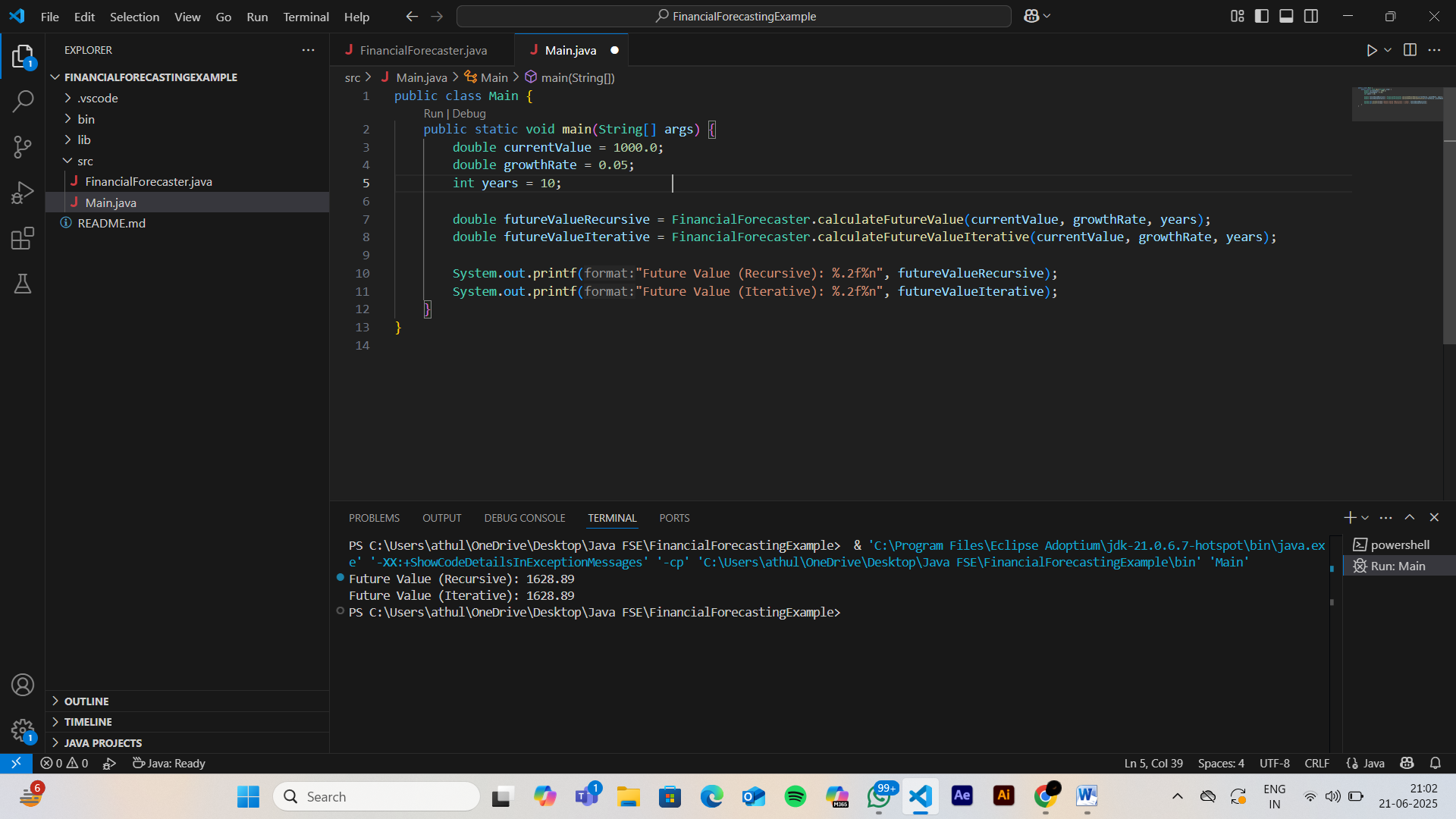
System.out.printf("Future Value (Recursive): %.2f%n", futureValueRecursive);

System.out.printf("Future Value (Iterative): %.2f%n", futureValueIterative);

}

}

**OUTPUT:**

****