DeepSkilling Assessment

ALGORITHMS\_DATA STRUCTURES

Exercise 1 : E – commerce Platform Search Function

**Code:**

**ProductSearchApp.java:**

package search;

import java.util.Arrays;

import java.util.Comparator;

class Product {

int productId;

String productName;

String category;

public Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

}

public class ProductSearchApp {

public static Product linearSearch(Product[] products, int id) {

for (Product p : products) {

if (p.productId == id)

return p;

}

return null;

}

public static Product binarySearch(Product[] products, int id) {

int low = 0, high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (products[mid].productId == id)

return products[mid];

else if (products[mid].productId < id)

low = mid + 1;

else

high = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(105, "Shoes", "Footwear"),

new Product(102, "T-shirt", "Clothing"),

new Product(110, "Phone", "Electronics")

};

Arrays.*sort*(products, Comparator.*comparingInt*(p -> p.productId));

int searchId = 101;

Product foundLinear = *linearSearch*(products, searchId);

Product foundBinary = *binarySearch*(products, searchId);

if (foundLinear != null)

System.***out***.println("Linear Search: Found " + foundLinear.productName);

else

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

if (foundBinary != null)

System.***out***.println("Binary Search: Found " + foundBinary.productName);

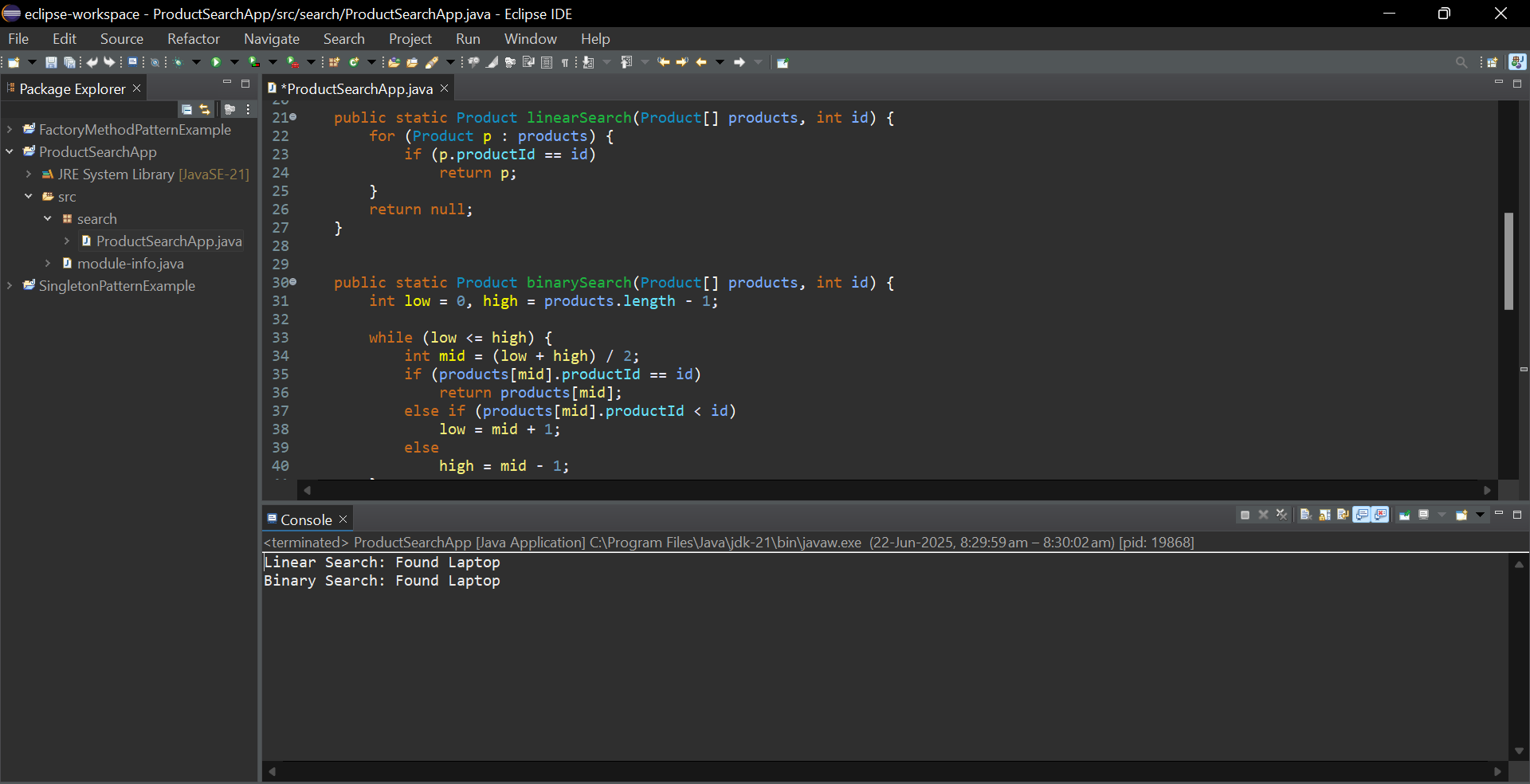
else

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

}

}

**Output:**

****

**Exercise 2 Financial Forecasting**

**Code:**

**FinancialForecastingApp:**

package weather;

public class FinancialForecast {

public static double forecast(double initialValue, double growthRate, int years) {

if (years == 0) {

return initialValue; // base case

}

return *forecast*(initialValue \* (1 + growthRate), growthRate, years - 1);

}

public static double forecastIterative(double initialValue, double growthRate, int years) {

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

initialValue \*= (1 + growthRate);

}

return initialValue;

}

public static void main(String[] args) {

double initial = 10000.0;

double rate = 0.08;

int years = 5;

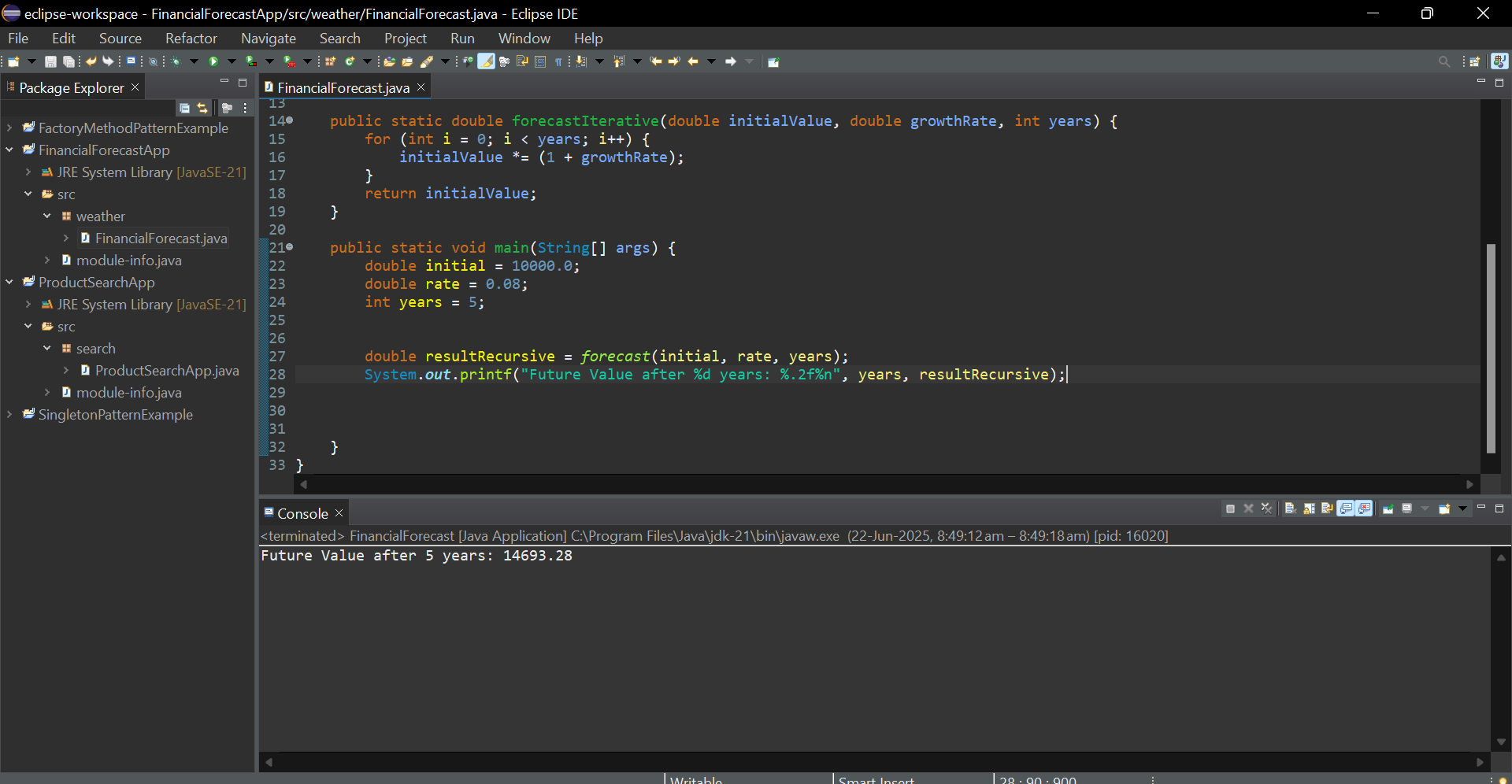
double resultRecursive = *forecast*(initial, rate, years);

System.***out***.printf("Future Value after %d years: %.2f%n", years, resultRecursive);

}

}

**Output :**

****