Week 1  
Data Structures and Algorithms  
  
**Exercise 2: E-commerce Platform Search Function**

Product.class

// Source code is decompiled from a .class file using FernFlower decompiler.

public class Product {

int productId;

String productName;

String category;

public Product(int var1, String var2, String var3) {

this.productId = var1;

this.productName = var2;

this.category = var3;

}

public String toString() {

return this.productId + " - " + this.productName + " (" + this.category + ")";

}

}

Product.java

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;

}

@Override

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

SearchDemo.class

// Source code is decompiled from a .class file using FernFlower decompiler.

import java.util.Arrays;

import java.util.Comparator;

public class SearchDemo {

public SearchDemo() {

}

public static Product linearSearch(Product[] var0, String var1) {

Product[] var2 = var0;

int var3 = var0.length;

for(int var4 = 0; var4 < var3; ++var4) {

Product var5 = var2[var4];

if (var5.productName.equalsIgnoreCase(var1)) {

return var5;

}

}

return null;

}

public static Product binarySearch(Product[] var0, String var1) {

int var2 = 0;

int var3 = var0.length - 1;

while(var2 <= var3) {

int var4 = var2 + (var3 - var2) / 2;

int var5 = var0[var4].productName.compareToIgnoreCase(var1);

if (var5 == 0) {

return var0[var4];

}

if (var5 < 0) {

var2 = var4 + 1;

} else {

var3 = var4 - 1;

}

}

return null;

}

public static void main(String[] var0) {

Product[] var1 = new Product[]{new Product(101, "Laptop", "Electronics"), new Product(102, "Shirt", "Clothing"), new Product(103, "Mobile", "Electronics"), new Product(104, "Watch", "Accessories"), new Product(105, "Shoes", "Footwear")};

System.out.println("\ud83d\udd0d Linear Search for 'Mobile':");

Product var2 = linearSearch(var1, "Mobile");

System.out.println(var2 != null ? var2 : "Product not found");

System.out.println("\n\ud83d\udd0d Binary Search for 'Mobile':");

Arrays.sort(var1, Comparator.comparing((var0x) -> {

return var0x.productName.toLowerCase();

}));

Product var3 = binarySearch(var1, "Mobile");

System.out.println(var3 != null ? var3 : "Product not found");

}

}

SearchDemo.java

import java.util.Arrays;

import java.util.Comparator;

public class SearchDemo {

// Linear Search

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

for (Product product : products) {

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

return product;

}

}

return null;

}

// Binary Search (array must be sorted by productName)

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

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 products[mid];

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

else right = mid - 1;

}

return null;

}

// Main method

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shirt", "Clothing"),

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

new Product(104, "Watch", "Accessories"),

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

};

System.out.println("🔍 Linear Search for 'Mobile':");

Product result1 = linearSearch(products, "Mobile");

System.out.println(result1 != null ? result1 : "Product not found");

System.out.println("\n🔍 Binary Search for 'Mobile':");

// Sort before binary search

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

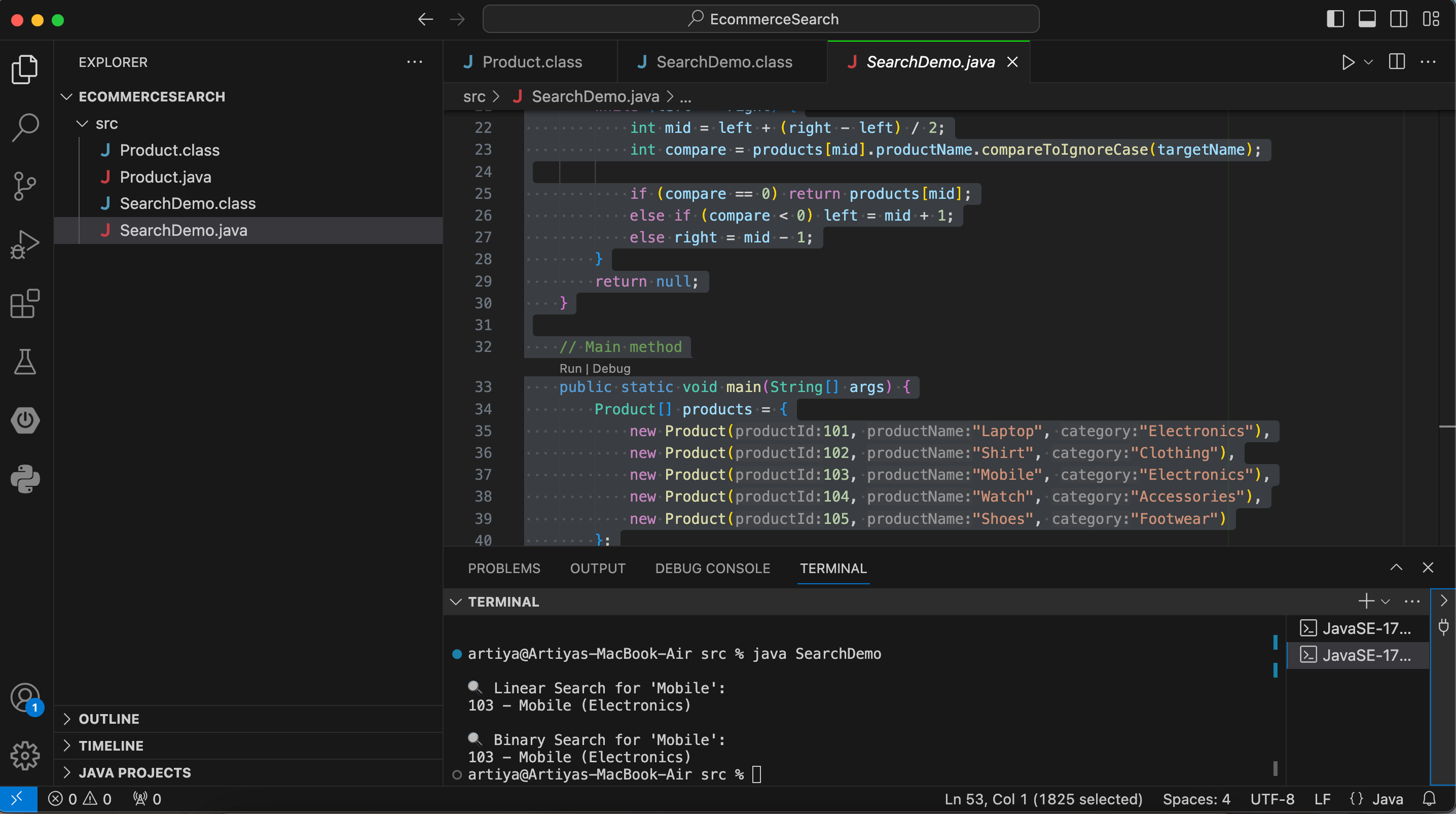
Product result2 = binarySearch(products, "Mobile");

System.out.println(result2 != null ? result2 : "Product not found");

}

}

Output

  
**Exercise 7: Financial Forecasting**

Solution:

ForecastTool.class

// Source code is decompiled from a .class file using FernFlower decompiler.

public class ForecastTool {

public ForecastTool() {

}

public static double calculateFutureValue(double var0, double var2, int var4) {

return var4 == 0 ? var0 : calculateFutureValue(var0 \* (1.0 + var2), var2, var4 - 1);

}

public static void main(String[] var0) {

double var1 = 10000.0;

double var3 = 0.08;

byte var5 = 5;

double var6 = calculateFutureValue(var1, var3, var5);

System.out.printf("Future Value after %d years: ₹%.2f%n", Integer.valueOf(var5), var6);

}

}

ForecastTool.java

public class ForecastTool {

// Recursive method to calculate future value

public static double calculateFutureValue(double presentValue, double rate, int years) {

if (years == 0) {

return presentValue;

}

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

}

public static void main(String[] args) {

double initialInvestment = 10000.0;

double annualGrowthRate = 0.08; // 8% annual growth

int forecastYears = 5;

double futureValue = calculateFutureValue(initialInvestment, annualGrowthRate, forecastYears);

System.out.printf("Future Value after %d years: ₹%.2f%n", forecastYears, futureValue);

}

}

Output  
