**Exercise 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.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.

1. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
2. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
3. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**SOLUTION:**

Big O notation quantifies how an algorithm's runtime or space requirements scale with input size, offering a standard for performance comparison independent of hardware. For search, best case is O(1) (found immediately), average and worst cases depend on the algorithm but represent typical and maximum execution times respectively.

**IMPLEMENTATION:**

**//**EcommerceSearch.java

import java.util.\*;

public class ECommerceSearch {

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

if (comparison == 0) {

return products[mid];

} else if (comparison < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(2, "T-shirt", "Fashion"),

new Product(3, "Shoes", "Footwear"),

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

new Product(5, "Watch", "Accessories")

};

Scanner scanner = new Scanner(System.in);

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

String searchName = scanner.nextLine();

Product result1 = linearSearch(products, searchName);

System.out.println("\n[Linear Search] Result: " + (result1 != null ? result1 : "Product Not Found"));

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

Product result2 = binarySearch(products, searchName);

System.out.println("[Binary Search] Result: " + (result2 != null ? result2 : "Product Not Found"));

scanner.close();

}

}  
  
//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;

}

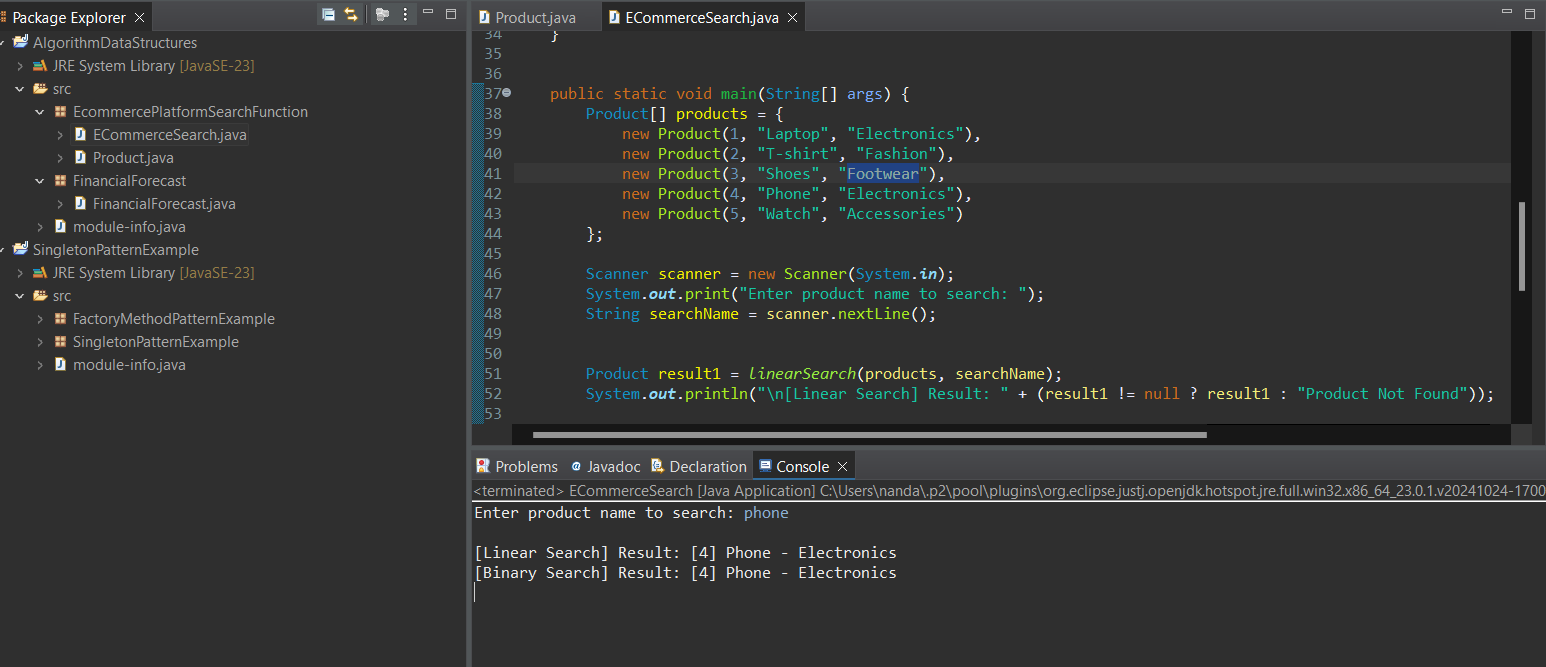
public String toString() {

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

}

}

OUTPUT



**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

CODE:

package FinancialForecast;

import java.util.Scanner;

public class FinancialForecast {

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

if (years == 0) {

return currentValue;

}

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

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the initial amount (e.g. 10000): ");

double initialAmount = scanner.nextDouble();

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

double ratePercent = scanner.nextDouble();

double annualGrowthRate = ratePercent / 100;

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

int years = scanner.nextInt();

double futureValue = calculateFutureValue(initialAmount, annualGrowthRate, years);

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

scanner.close();

}

}  
  
OUTPUT:

