**Algorithms\_Data Structures**

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

Code:

package weekone;

import java.util.\*;

class TrieNode{

Map<Character, TrieNode> children = new HashMap<>();

boolean isEndOfWord;

List<String> suggestions = new ArrayList<>();

}

class productsearch{

private final TrieNode root;

public productsearch() {

root = new TrieNode();

}

public void insert(String product) {

TrieNode node = root;

for(char ch:product.toLowerCase().toCharArray()) {

node.children.putIfAbsent(ch,new TrieNode());

node = node.children.get(ch);

if(node.suggestions.size()<5)node.suggestions.add(product);

}

node.isEndOfWord=true;

}

public List<String> search(String prefix){

TrieNode node = root;

for(char ch : prefix.toLowerCase().toCharArray()) {

node = node.children.get(ch);

if(node==null) return Collections.*emptyList*();

}

return node.suggestions;

}

}

public class ecommercesearchfunction {

public static void main(String[]args) {

productsearch searchengine = new productsearch();

String[] products= {

"iPhone", "iPad", "iMac", "iPod", "iWatch",

"Samsung Galaxy", "Samsung Note", "Sony Xperia",

"Nokia Lumia", "Google Pixel"

};

for(String product: products) {

searchengine.insert(product);

}

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

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

String input = scanner.nextLine();

List<String> results = searchengine.search(input);

if (results.isEmpty()) {

System.***out***.println("No products found.");

} else {

System.***out***.println("Search Results:");

for (String res : results) {

System.***out***.println("- " + res);

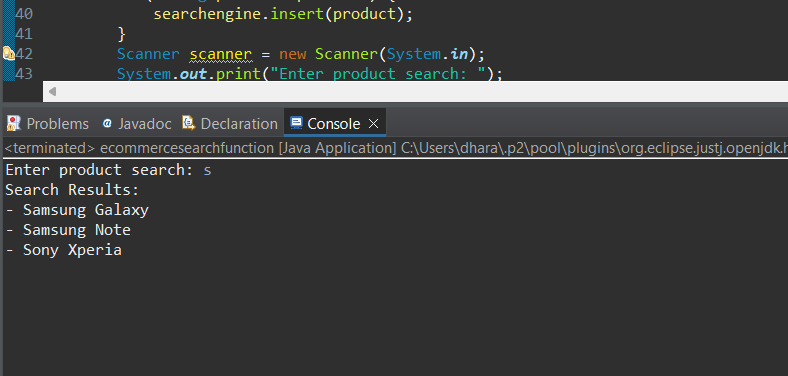
}

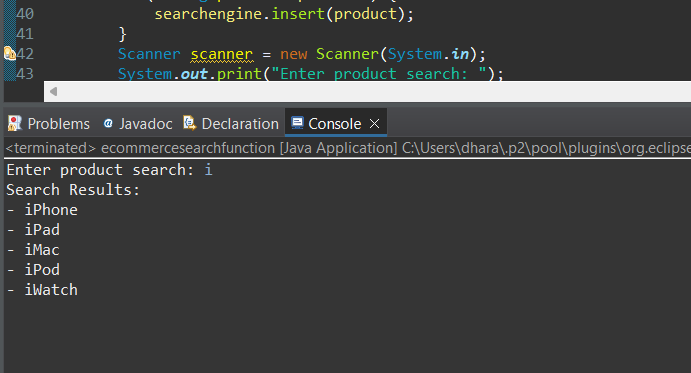
}

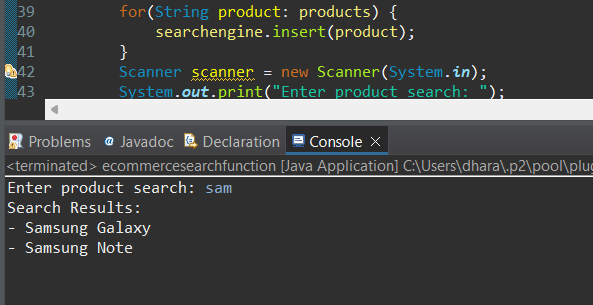
}

}

Output:







**Exercise 7: Financial Forecasting**

Code:

package weekone;

import java.util.\*;

public class FinancialForecasting {

static double[] movingAverage(int[] profits, int windowSize) {

int n = profits.length;

double[] result = new double[n - windowSize + 1];

int windowSum = 0;

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

windowSum += profits[i];

}

result[0] = windowSum / (double) windowSize;

for (int i = windowSize; i < n; i++) {

windowSum += profits[i] - profits[i - windowSize];

result[i - windowSize + 1] = windowSum / (double) windowSize;

}

return result;

}

static int forecastNextMonth(int[] profits) {

int n = profits.length;

if (n < 2) return profits[n - 1];

int diff = profits[n - 1] - profits[n - 2];

return profits[n - 1] + diff;

}

public static void main(String[] args) {

int[] monthlyProfits = {15000, 16000, 15800, 17000, 17500, 18000, 21000, 15200, 16100, 15950, 17100, 17750, 18500, 19000, 19800, 20500, 21200};

System.***out***.println("Monthly Profits: " + Arrays.*toString*(monthlyProfits));

double[] averages = *movingAverage*(monthlyProfits, 3);

System.***out***.println("3-Month Moving Averages: " + Arrays.*toString*(averages));

int forecast = *forecastNextMonth*(monthlyProfits);

System.***out***.println("Forecast for next month: ₹" + forecast);

}

}

Output:

