**DATA STRUCTURES AND ALGORITHMS**

**1.E-COMMEREC PLATFORM SEARCH FUNCTION:**

**CODE:**

import java.util.\*;

class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

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

}

}

class Search {

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

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

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int check = products[mid].productName.compareToIgnoreCase(name);

if (check == 0) return products[mid];

if (check < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

}

public class Main {

public static void main(String[] args) {

Product[] products = {

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

new Product(2, "Phone", "Electronics"),

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

new Product(4, "Watch", "Accessories"),

new Product(5, "Tablet", "Electronics")

};

Scanner scan = new Scanner(System.in);

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

String name = scan.nextLine();

Product found1 = Search.linearSearch(products, name);

if (found1 != null) System.out.println("Linear Search: " + found1);

else System.out.println("Linear Search: Not Found");

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

Product found2 = Search.binarySearch(products, name);

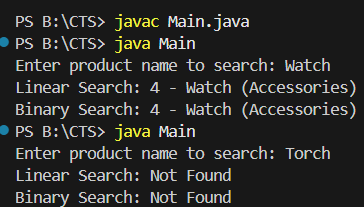
if (found2 != null) System.out.println("Binary Search: " + found2);

else System.out.println("Binary Search: Not Found");

}

}

Output:



**2.FINANCIAL FORECATING**

**CODE:**

import java.util.Scanner;

public class FinancialForecast {

public static double futureValue(double currentValue, double rate, int years) {

if (years == 0) return currentValue;

return futureValue(currentValue, rate, years - 1) \* (1 + rate);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter current value (₹): ");

double currentValue = sc.nextDouble();

System.out.print("Enter annual interest rate (e.g., 0.05 for 5%): ");

double rate = sc.nextDouble();

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

int years = sc.nextInt();

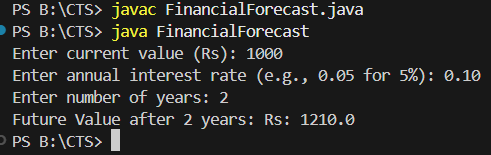
double result = futureValue(currentValue, rate, years);

System.out.println("Future Value after " + years + " years: ₹" + result);

}

}

Output:



**PRACTICE QNS:**

**DATA STRUCTURES AND ALGORITHMS**

**Customer orders:**

**Code:**

import java.util.\*;

class Order {

String orderId;

String customerName;

double totalPrice;

public Order(String orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public String toString() {

return orderId + " - " + customerName + " - $" + totalPrice;

}

}

public class SortOrders {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

}

}

}

}

public static void quickSort(Order[] orders, int low, int high) {

if (low < high) {

int pi = partition(orders, low, high);

quickSort(orders, low, pi - 1);

quickSort(orders, pi + 1, high);

}

}

public static int partition(Order[] orders, int low, int high) {

double pivot = orders[high].totalPrice;

int i = low - 1;

for (int j = low; j < high; j++) {

if (orders[j].totalPrice < pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

orders[i + 1] = orders[high];

orders[high] = temp;

return i + 1;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = Integer.parseInt(scanner.nextLine());

Order[] orders1 = new Order[n];

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

System.out.println("Enter details for order " + (i + 1) + ":");

System.out.print("Order ID: ");

String id = scanner.nextLine();

System.out.print("Customer Name: ");

String name = scanner.nextLine();

System.out.print("Total Price: ");

double price = Double.parseDouble(scanner.nextLine());

orders1[i] = new Order(id, name, price);

}

Order[] orders2 = orders1.clone();

System.out.println("\nOriginal Orders:");

for (Order o : orders1) System.out.println(o);

bubbleSort(orders1);

System.out.println("\nSorted by Bubble Sort:");

for (Order o : orders1) System.out.println(o);

quickSort(orders2, 0, orders2.length - 1);

System.out.println("\nSorted by Quick Sort:");

for (Order o : orders2) System.out.println(o);

System.out.println("\nTime Complexity:");

System.out.println("Bubble Sort: O(n^2)");

System.out.println("Quick Sort: Average O(n log n), Worst O(n^2)");

System.out.println("Quick Sort is generally preferred for better performance on large datasets.");

}

}

Output:

