**1: E-commerce Platform Search Function**

**Big O Notation:**

It describes the time complexity or space complexity of an algorithm or a program. i.e.) how its performances grows relative to input.

**Product Class Implementation:**

**Product.java**

package search;  
  
public class Product {  
 private int productId;  
 private String productName;  
 private String category;  
 public Product(int productId, String productName, String category) {  
 this.productId = productId;  
 this.productName = productName;  
 this.category = category;  
 }  
 public int getProductId() {  
 return productId;  
 }  
 public String getProductName() {  
 return productName;  
 }  
 public String getCategory() {  
 return category;  
 }  
 @Override  
 public String toString() {  
 return productId + ": " + productName + " [" + category + "]";  
 }  
}

**LinearSearch.java**

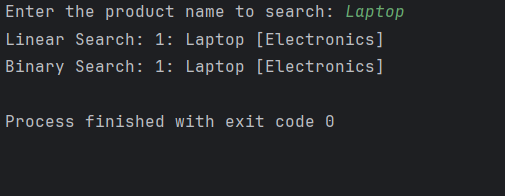
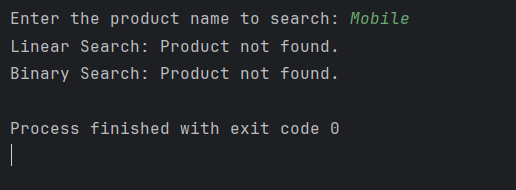
package search;  
  
import java.util.List;  
  
public class LinearSearch {  
 public static Product search(List<Product> products, String name) {  
 for (Product product : products) {  
 if (product.getProductName().equalsIgnoreCase(name)) {  
 return product;  
 }  
 }  
 return null;  
 }  
}

**BinarySearch.java**

package search;  
  
import java.util.Arrays;  
import java.util.Comparator;  
  
public class BinarySearch {  
 public static Product search(Product[] products, String name) {  
 Arrays.*sort*(products, Comparator.*comparing*(Product::getProductName));  
  
 int low = 0, high = products.length - 1;  
  
 while (low <= high) {  
 int mid = (low + high) / 2;  
 int result = name.compareToIgnoreCase(products[mid].getProductName());  
  
 if (result == 0) {  
 return products[mid];  
 } else if (result < 0) {  
 high = mid - 1;  
 } else {  
 low = mid + 1;  
 }  
 }  
  
 return null;  
 }  
}

**Main.java**

package search;  
  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args)   
 List<Product> products = new ArrayList<>();  
 products.add(new Product(1, "Laptop", "Electronics"));  
 products.add(new Product(2, "Shoes", "Footwear"));  
 products.add(new Product(3, "Camera", "Electronics"));  
 products.add(new Product(4, "T-shirt", "Clothing"));  
 products.add(new Product(5, "Mouse", "Accessories"));  
  
 // Scanner for user input  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the product name to search: ");  
 String searchQuery = scanner.nextLine();  
  
 Product linearResult = LinearSearch.*search*(products, searchQuery);  
 if (linearResult != null)  
 System.*out*.println("Linear Search: " + linearResult);  
 else  
 System.*out*.println("Linear Search: Product not found.");  
  
 Product[] productArray = products.toArray(new Product[0]); // Convert list to array  
 Product binaryResult = BinarySearch.*search*(productArray, searchQuery);  
 if (binaryResult != null)  
 System.*out*.println("Binary Search: " + binaryResult);  
 else  
 System.*out*.println("Binary Search: Product not found.");  
  
 scanner.close();  
 }  
}



**2: Financial Forecasting**

**Recursion:**

Recursion is a programming concept where a function calls itself within its own definition to solve a problem. It works by breaking down a complex problem into smaller, similar subproblems until a simple "base case" is reached, which can be solved directly without further recursion. The solutions to the subproblems are then combined to solve the original problem.

**Use of recursion:**

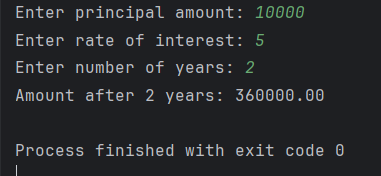
* Breaks the problem into smaller sub-problems.
* Makes mathematical problems like compound growth, Fibonacci, etc., easier to implement.
* Improves code readability.

**FinancialForecast.java**

package forecasting;  
  
public class FinancialForecast {  
 public static double calculateFutureValue(double principal, double rate, int years) {  
 if (years == 0) {  
 return principal;  
 }  
 return *calculateFutureValue*(principal, rate, years - 1) \* (1 + rate);  
 }  
}

**Main.java**

package forecasting;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 // Input from user  
 System.*out*.print("Enter principal amount: ");  
 double principal = scanner.nextDouble();  
  
 System.*out*.print("Enter rate of interest: ");  
 double rate = scanner.nextDouble();  
  
 System.*out*.print("Enter number of years: ");  
 int years = scanner.nextInt();  
  
 // Calculate future value recursively  
 double futureValue = FinancialForecast.*calculateFutureValue*(principal, rate, years);  
  
 // Display result  
 System.*out*.printf("Amount after %d years: %.2f\n", years, futureValue);  
  
 scanner.close();  
 }  
}

****