# Exercise 2: E-commerce Platform Search Function

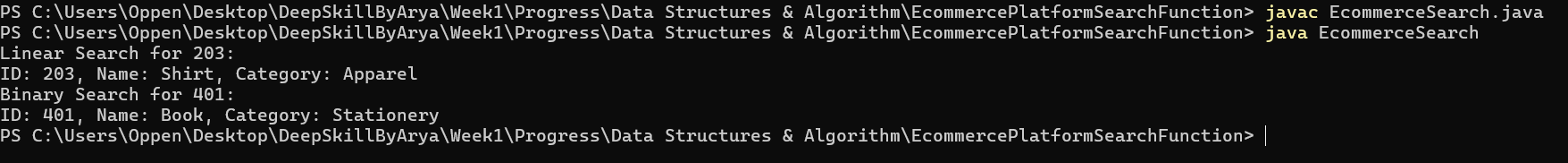
Data Structures and Algorithms

Scenario:  
You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

## Code: EcommerceSearch.java

import java.util.Arrays;  
import java.util.Scanner;  
  
class Product {  
 int productId;  
 String productName;  
 String category;  
 Product(int id, String name, String category) {  
 this.productId = id;  
 this.productName = name;  
 this.category = category;  
 }  
 public String toString() {  
 return "ID: " + productId + ", Name: " + productName + ", Category: " + category;  
 }  
}  
public class EcommerceSearch {  
 public static Product linearSearch(Product[] products, int id) {  
 for (Product p : products) {  
 if (p.productId == id) return p;  
 }  
 return null;  
 }  
 public static Product binarySearch(Product[] products, int id) {  
 int low = 0, high = products.length - 1;  
 while (low <= high) {  
 int mid = (low + high) / 2;  
 if (products[mid].productId == id) return products[mid];  
 if (products[mid].productId < id) low = mid + 1;  
 else high = mid - 1;  
 }  
 return null;  
 }  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(305, "Coffee Maker", "Home"),  
 new Product(203, "Shirt", "Apparel"),  
 new Product(101, "Laptop", "Electronics"),  
 new Product(401, "Book", "Stationery")  
 };  
 Arrays.sort(products, (a, b) -> Integer.compare(a.productId, b.productId));  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter Product ID to search: ");  
 int id = scanner.nextInt();  
 Product p1 = linearSearch(products, id);  
 System.out.println("[Linear Search Result] " + (p1 != null ? p1 : "Product not found."));  
 Product p2 = binarySearch(products, id);  
 System.out.println("[Binary Search Result] " + (p2 != null ? p2 : "Product not found."));  
 scanner.close();  
 }  
}

## Output Screenshot



# Exercise 7: Financial Forecasting

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

## Code: FinancialForecast.java

public class FinancialForecast {  
 public static double forecastRecursive(double currentValue, double rate, int years) {  
 if (years == 0) return currentValue;  
 return forecastRecursive(currentValue \* (1 + rate), rate, years - 1);  
 }  
 public static double forecastIterative(double currentValue, double rate, int years) {  
 for (int i = 0; i < years; i++) {  
 currentValue \*= (1 + rate);  
 }  
 return currentValue;  
 }  
 public static void main(String[] args) {  
 double initialValue = 10000;  
 double growthRate = 0.10;  
 int years = 5;  
 double recursiveResult = forecastRecursive(initialValue, growthRate, years);  
 double iterativeResult = forecastIterative(initialValue, growthRate, years);  
 System.out.printf("Recursive Forecast: ₹%.2f\n", recursiveResult);  
 System.out.printf("Iterative Forecast: ₹%.2f\n", iterativeResult);  
 }  
}

## Output Screenshot

