**Exercise 2: E-commerce Platform – Search Functionality**

**Scenario:**  
You are building the search feature for an e-commerce platform. The goal is to ensure users can search for products quickly and efficiently. To achieve this, you will implement and compare different search algorithms.

**Steps to Complete the Exercise**

**1. Understand Asymptotic Notation**

* Explain **Big O Notation** as a tool for analyzing algorithm efficiency in terms of time and space complexity.
* Define and differentiate:
  + **Best Case** – Minimum time the algorithm takes under ideal conditions.
  + **Average Case** – Expected time for a general case.
  + **Worst Case** – Maximum time the algorithm may take under the most unfavorable conditions.

**2. Setup the Product Structure**

* Create a Product class with attributes relevant for searching:
  + productId (int)
  + productName (String)
  + category (String)

**3. Implement Search Algorithms**

* **Linear Search**:
  + Store products in an unsorted array.
  + Search for a product by comparing each element sequentially.
* **Binary Search**:
  + Sort the array based on productId or productName.
  + Implement binary search to find products faster in sorted arrays.

**4. Analyze and Compare**

* Evaluate time complexities:
  + **Linear Search** – O(n)
  + **Binary Search** – O(log n) (only applicable for sorted data)
* Discuss the trade-offs:
  + Linear search works on unsorted data but is slower.
  + Binary search is significantly faster but requires data to be sorted.
* Determine which approach fits better for large-scale, real-time e-commerce search needs and justify your choice (binary search or indexed structures often win in scalability).

Source code :

import java.util.Arrays;  
import java.util.Comparator;  
  
public class EcommerceSearch {  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(201, "Bluetooth Speaker", "Electronics"),  
 new Product(205, "Yoga Mat", "Fitness"),  
 new Product(203, "Water Bottle", "Home & Kitchen"),  
 new Product(202, "Graphic T-Shirt", "Clothing"),  
 new Product(204, "Wireless Mouse", "Electronics")  
 };  
  
 int targetId = 203;  
  
 int linearIndex = SearchEngine.*linearSearch*(products, targetId);  
 System.*out*.println("Linear Search Index: " + linearIndex);  
 if (linearIndex != -1) {  
 System.*out*.println("Product Found: " + products[linearIndex]);  
 } else {  
 System.*out*.println("Product not found using Linear Search.");  
 }  
  
 Arrays.*sort*(products, Comparator.*comparingInt*(Product::getId));  
  
 int binaryIndex = SearchEngine.*binarySearch*(products, targetId);  
 System.*out*.println("Binary Search Index: " + binaryIndex);  
 if (binaryIndex != -1) {  
 System.*out*.println("Product Found: " + products[binaryIndex]);  
 } else {  
 System.*out*.println("Product not found using Binary Search.");  
 }  
 }  
}  
  
class Product {  
 private int id;  
 private String name;  
 private String category;  
  
 public Product(int id, String name, String category) {  
 this.id = id;  
 this.name = name;  
 this.category = category;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String toString() {  
 return id + " - " + name + " - " + category;  
 }  
}  
  
class SearchEngine {  
 public static int linearSearch(Product[] items, int targetId) {  
 for (int i = 0; i < items.length; i++) {  
 if (items[i].getId() == targetId) {  
 return i;  
 }  
 }  
 return -1;  
 }  
  
 public static int binarySearch(Product[] items, int targetId) {  
 int low = 0, high = items.length - 1;  
  
 while (low <= high) {  
 int mid = low + (high - low) / 2;  
  
 if (items[mid].getId() == targetId) {  
 return mid;  
 } else if (items[mid].getId() < targetId) {  
 low = mid + 1;  
 } else {  
 high = mid - 1;  
 }  
 }  
 return -1;  
 }  
}

Output :

