**EXERCISE 2:**

**E-COMMERCE PLATFORM SEARCH FUNCTION :**

Big O Notation:

Big O notation is used to describe the performance or complexity of an algorithm. It specifically describes the worst-case scenario but can also provide insights into the average and best cases. It helps in understanding how the time or space requirements of an algorithm grow as the input size grows.  
  
class SearchProduct {  
 private String productId;  
 private String productName;  
 private String category;  
  
 public SearchProduct(String productId, String productName, String category) {  
 this.productId = productId;  
 this.productName = productName;  
 this.category = category;  
 }  
  
 public String getProductId() { return productId; }  
 public String getProductName() { return productName; }  
 public String getCategory() { return category; }  
  
 @Override  
 public String toString() {  
 return "SearchProduct{" +  
 "productId='" + productId + '\'' +  
 ", productName='" + productName + '\'' +  
 ", category='" + category + '\'' +  
 '}';  
 }  
}  
  
class Search {  
 // Linear Search  
 public static int linearSearch(SearchProduct[] products, String category) {  
 for (int i = 0; i < products.length; i++) {  
 if (products[i].getCategory().equalsIgnoreCase(category)) {  
 return i;  
 }  
 }  
 return -1;  
 }  
  
 // Binary Search (assumes the array is sorted by category)  
 public static int binarySearch(SearchProduct[] products, String category) {  
 int left = 0;  
 int right = products.length - 1;  
  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
 int comparison = products[mid].getCategory().compareToIgnoreCase(category);  
  
 if (comparison == 0) {  
 return mid;  
 } else if (comparison < 0) {  
 left = mid + 1;  
 } else {  
 right = mid - 1;  
 }  
 }  
 return -1;  
 }  
}  
  
public class ECommerceSearchExample {  
 public static void main(String[] args) {  
 SearchProduct[] products = {  
 new SearchProduct("1", "Laptop", "Electronics"),  
 new SearchProduct("2", "Phone", "Electronics"),  
 new SearchProduct("3", "Shirt", "Clothing"),  
 new SearchProduct("4", "Book", "Education")  
 };  
  
 // Linear Search by category  
 String searchKey = "Electronics";  
 int index = Search.*linearSearch*(products, searchKey);  
 if (index != -1) {  
 System.*out*.println("Product found with Linear Search by category: " + products[index]);  
 } else {  
 System.*out*.println("Product not found with Linear Search by category.");  
 }  
  
 // Sort the array for Binary Search by category  
 java.util.Arrays.*sort*(products, (p1, p2) -> p1.getCategory().compareToIgnoreCase(p2.getCategory()));  
  
 // Binary Search by category  
 index = Search.*binarySearch*(products, searchKey);  
 if (index != -1) {  
 System.*out*.println("Product found with Binary Search by category: " + products[index]);  
 } else {  
 System.*out*.println("Product not found with Binary Search by category.");  
 }  
 }  
}