## Exercise 2: E-commerce Platform Search Function

**Scenario:**

### 1. Understand Asymptotic Notation:

- Big O Notation describes the upper bound of an algorithm's running time in terms of input size n.

- It helps analyze performance and scalability of algorithms.

- Best, Average, and Worst-case scenarios describe how an algorithm performs under different conditions.

- Linear Search: Best O(1), Average O(n), Worst O(n)

- Binary Search: Best O(1), Average O(log n), Worst O(log n)

### 2. Setup:

Create a class Product with attributes for searching: productId, productName, and category.

```
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 "Product[ID=" + productId + ", Name=" + productName + ", Category=" +
category + "]";
   }}
```

### 3. Implementation:

Implement linear search and binary search algorithms.

Store products in an array for linear search and a sorted array for binary search.

```
public static Product linearSearch(Product[] products, String name) {
   for (Product product : products) {
      if (product.getProductName().equalsIgnoreCase(name)) {
         return product;
      }
   }
   return null;
}
```

```
public static Product binarySearch(Product[] products, String name) {
   int left = 0, right = products.length - 1;

   while (left <= right) {
      int mid = left + (right - left) / 2;
      int cmp = products[mid].getProductName().compareToIgnoreCase(name);

      if (cmp == 0) return products[mid];
      else if (cmp < 0) left = mid + 1;
      else right = mid - 1;
   }

   return null;
}
```

```
public static void sortProductsByName(Product[] products) {
   Arrays.sort(products, Comparator.comparing(Product::getProductName,
String.CASE_INSENSITIVE_ORDER));
}
```

## 4. Analysis:

- Linear Search: O(n), easy but slow for large datasets.

- Binary Search: O(log n), fast but requires sorted array.

**Conclusion:**

Binary search is more suitable for large datasets where performance is critical.