Exercise 2: E-commerce Platform Search Function

---

1. Understanding Asymptotic Notation

What is Big O?

Big O notation is a way to tell **how fast or slow an algorithm works** when we give it more data. It doesn't show the exact time but shows how the time grows.

For example:

- O(n) means if you double the data, the time also doubles.
- O(1) means the time stays the same no matter how much data.

It helps us **choose the best algorithm** for large data.

Best, Average, and Worst Cases:

- **Best Case**: The search finds the item very quickly. (e.g., first position)
- **Average Case**: Normal case, when the item is somewhere in the middle.
- **Worst Case**: The item is not found, or it’s at the end (takes the longest time).

---

2. Setup: Product Class

We will create a simple class called Product with basic details like ID, name, and category.

```
public class Product {
    int productId;
```

```
    String productName;
    String category;

    // constructor
    public Product(int id, String name, String cat) {
        productId = id;
        productName = name;
        category = cat;
    }
}
```

---

3. Implementation of Searches

We are going to do:

- **Linear search**: check one by one
- **Binary search**: check in middle (only works if list is sorted)

```
import java.util.Arrays;
import java.util.Comparator;

public class SearchExample {

    // Linear Search
    public static int linearSearch(Product[] products, String searchName) {
        for (int i = 0; i < products.length; i++) {
            if (products[i].productName.equalsIgnoreCase(searchName)) {
                return i; // found
            }
        }
        return -1; // not found
    }

    // Binary Search
```

```
    public static int binarySearch(Product[] products, String searchName) {
        int start = 0;
        int end = products.length - 1;

        while (start <= end) {
            int mid = (start + end) / 2;
            int cmp =
searchName.compareToIgnoreCase(products[mid].productName);

            if (cmp == 0) {
                return mid;
            } else if (cmp > 0) {
                start = mid + 1;
            } else {
                end = mid - 1;
            }
        }

        return -1; // not found
    }

    public static void main(String[] args) {
        Product[] products = {
            new Product(1, "Laptop", "Electronics"),
            new Product(2, "Mouse", "Accessories"),
            new Product(3, "Keyboard", "Accessories"),
            new Product(4, "Phone", "Electronics"),
            new Product(5, "Tablet", "Electronics")
        };

        // Linear search test
        int result1 = linearSearch(products, "Phone");
        System.out.println("Linear Search: Found at index = " + result1);

        // Sort the products for binary search
        Arrays.sort(products, new Comparator<Product>() {
            public int compare(Product p1, Product p2) {
```

```
            return p1.productName.compareToIgnoreCase(p2.productName);
        }
    });

    // Binary search test
    int result2 = binarySearch(products, "Phone");
    System.out.println("Binary Search: Found at index = " + result2);
  }
}
```

---

4. Output: searchhing “Phone” :

```
PS E:\GenC Hands on\Week 1\java codes> javac Product.java SearchExample.java
>>
PS E:\GenC Hands on\Week 1\java codes> java SearchExample
Linear Search: Found at index = 3
Binary Search: Found at index = 3
PS E:\GenC Hands on\Week 1\java codes>
```

4. Analysis: Which one is better?

Search Type	Time Complexity	Needs Sorted Dataa?
Linear	O(n)	NO
Binary	O(log n)	YES

Conclusion:

- **Linear search** is easy but slow when the list is big.
- **Binary search** is faster but only works if the list is sorted.

So for a big e-commerce site with lots of products, **binary search is better** — but only after sorting the data once.