

## WEEK1 – Data Structures and Algorithms HandsOn

**Exercise2: Objective:** To implement linear and binary search for product search optimization in an e-commerce system and analyze their performance.

### STEPS :

#### Understanding Asymptotic Notation

Big O Notation Overview:

- Big O (O): Worst-case runtime.
- Big  $\Omega$  (Omega): Best-case runtime.
- Big  $\Theta$  (Theta): Average-case runtime.

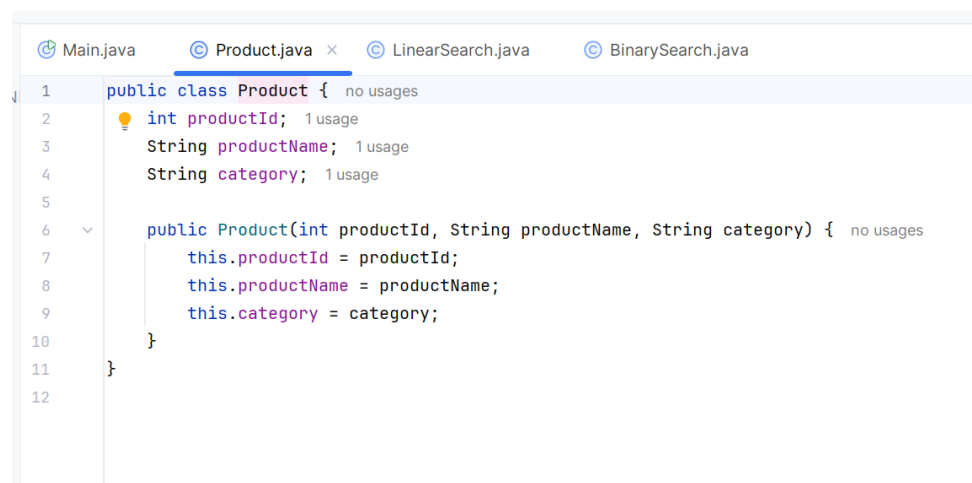
Why it matters in search:

Algorithm	Best Case	Average Case	Worst Case
-----------	-----------	--------------	------------

Linear Search	$O(1)$	$O(n)$	$O(n)$
---------------	--------	--------	--------

Binary Search	$O(1)$	$O(\log n)$	$O(\log n)$
---------------	--------	-------------	-------------

#### Product.java :



```
1 public class Product { no usages
2     int productId; 1 usage
3     String productName; 1 usage
4     String category; 1 usage
5
6     public Product(int productId, String productName, String category) { no usages
7         this.productId = productId;
8         this.productName = productName;
9         this.category = category;
10    }
11 }
12
```

## Linear Search Implementation:

### LinearSearch.java :

```
Main.java x Product.java LinearSearch.java x BinarySearch.java

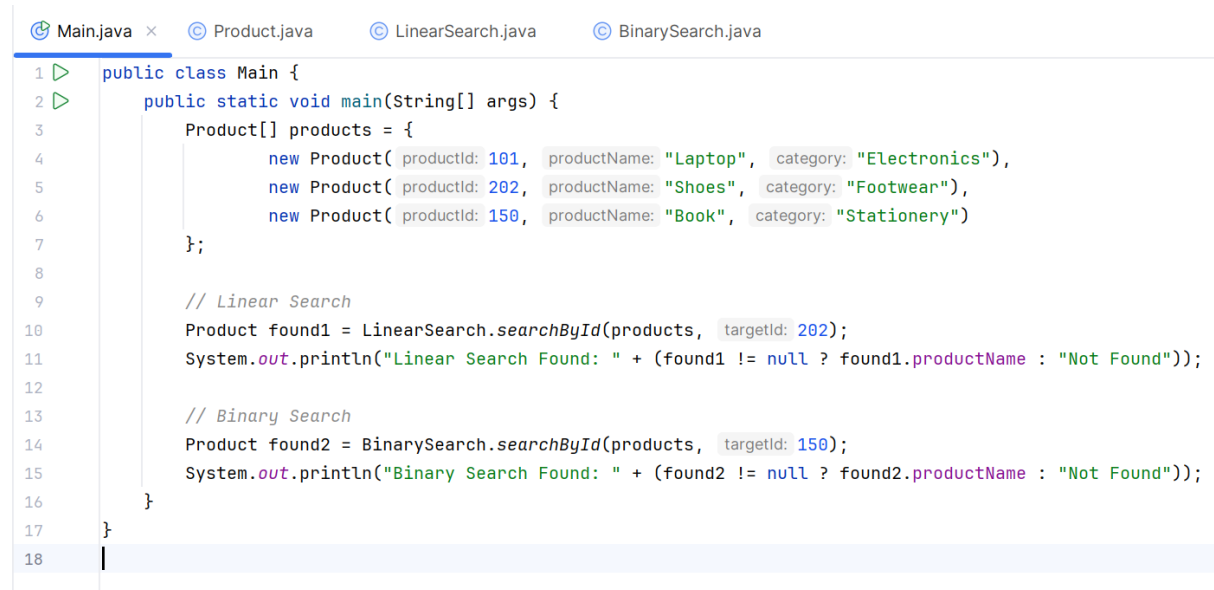
public class LinearSearch { no usages
2 @ public static Product searchById(Product[] products, int targetId) {
3   for (Product p : products) {
4     if (p.productId == targetId) {
5       return p;
6     }
7   }
8   return null;
9 }
10 }
11 }
```

## BinarySearch Implementation:

### BinarySearch.java :

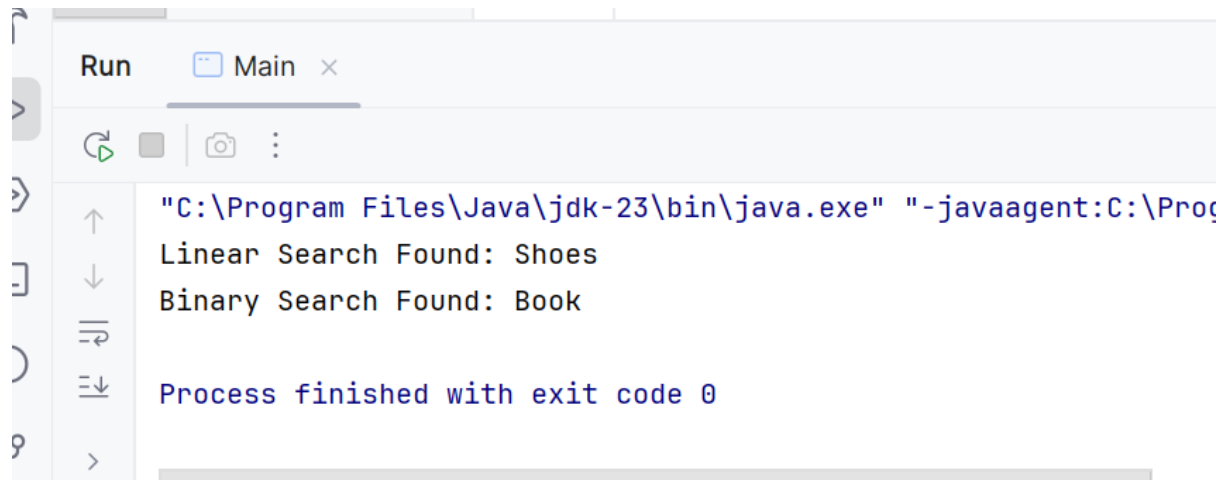
```
1 import java.util.Arrays;
2 import java.util.Comparator;
3
4 public class BinarySearch { no usages
5 @ public static Product searchById(Product[] products, int targetId) { no usages
6   // Ensure the array is sorted by productId
7   Arrays.sort(products, Comparator.comparingInt( Product p -> p.productId));
8
9   int low = 0, high = products.length - 1;
10  while (low <= high) {
11    int mid = (low + high) / 2;
12    if (products[mid].productId == targetId) {
13      return products[mid];
14    } else if (products[mid].productId < targetId) {
15      low = mid + 1;
16    } else {
17      high = mid - 1;
18    }
19  }
20  return null;
21 }
22 }
```

## Main.java :



```
1 public class Main {
2     public static void main(String[] args) {
3         Product[] products = {
4             new Product( productId: 101, productName: "Laptop", category: "Electronics"),
5             new Product( productId: 202, productName: "Shoes", category: "Footwear"),
6             new Product( productId: 150, productName: "Book", category: "Stationery")
7         };
8
9         // Linear Search
10        Product found1 = LinearSearch.searchById(products, targetId: 202);
11        System.out.println("Linear Search Found: " + (found1 != null ? found1.productName : "Not Found"));
12
13        // Binary Search
14        Product found2 = BinarySearch.searchById(products, targetId: 150);
15        System.out.println("Binary Search Found: " + (found2 != null ? found2.productName : "Not Found"));
16    }
17 }
18
```

## Output :



```
Run Main x
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Progr
Linear Search Found: Shoes
Binary Search Found: Book
Process finished with exit code 0
```