**Exercise 2: E-commerce Platform Search Function**

**Scenario:**  
You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**1. Understand Asymptotic Notation**

**Big O Notation:**  
Big O notation is used to describe the performance or complexity of an algorithm. It tells us how the algorithm behaves as the input size grows.

* **O(1):** Constant time
* **O(n):** Linear time
* **O(log n):** Logarithmic time
* **O(n log n):** Linearithmic time
* **O(n^2):** Quadratic time

**Search Operation Scenarios:**

* **Best Case:** Item is found at the beginning (O(1) for linear search)
* **Average Case:** Item is somewhere in the middle
* **Worst Case:** Item is not present or at the end (O(n) for linear, O(log n) for binary)

**2. Setup**

**Product Class:**

public class Product {

int productId;

String productName;

String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

@Override

public String toString() {

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

}

}

**3. Implementation**

**Search Class:**

import java.util.\*;

public class Search {

public static Product linearSearch(Product[] products, String name) {

for (Product product : products) {

if (product.productName.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 = name.compareToIgnoreCase(products[mid].productName);

if (cmp == 0) return products[mid];

else if (cmp < 0) right = mid - 1;

else left = mid + 1;

}

return null;

}

public static void sortByName(Product[] products) {

Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

}

}

**Main Class:**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop", "Electronics"),

new Product(102, "Chair", "Furniture"),

new Product(103, "Book", "Stationary"),

new Product(104, "Phone", "Electronics"),

new Product(105, "Table", "Furniture")

};

Scanner scanner = new Scanner(System.in);

System.out.print("Enter product name to search: ");

String searchName = scanner.nextLine();

System.out.println("\n---By Linear Search ---");

Product result1 = Search.linearSearch(products, searchName);

System.out.println(result1 != null ? result1 : "Product not found.");

System.out.println("\n---By Binary Search ---");

Search.sortByName(products);

Product result2 = Search.binarySearch(products, searchName);

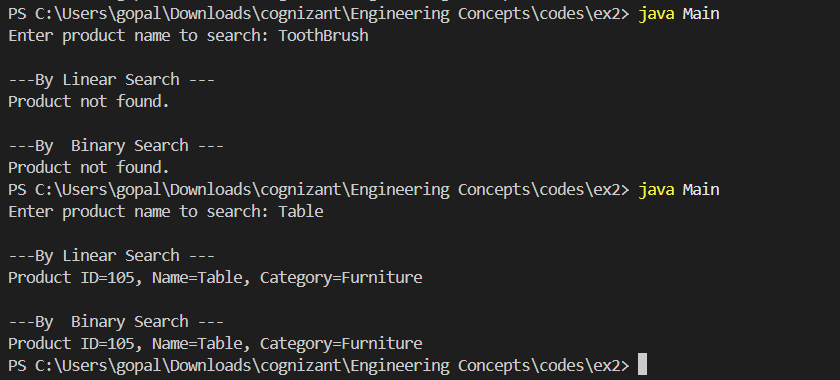
System.out.println(result2 != null ? result2 : "Product not found.");

scanner.close();

}

}

**Output:**



**4. Analysis**

**Linear Search:**

* **Time Complexity:** O(n)
* **Space Complexity:** O(1)
* **When to Use:** Unsorted data or small datasets.

**Binary Search:**

* **Time Complexity:** O(log n)
* **Space Complexity:** O(1)
* **When to Use:** Sorted arrays, large datasets.

**Conclusion:**  
For real-time applications where performance matters, binary search is preferred. However, it requires pre-sorting. Linear search is simpler and can be used when sorting is not feasible or the dataset is small.