**Exercise 2: E-commerce Platform Search Function**  
  
**Problem Statement** : You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.  
  
**CODE** :

**There are two methods:  
  
1. Using LINEAR SEARCH**

**public class SearchFunction {**

**public static Product linearSearch(Product[] products, int targetId) {**

**for (Product product : products) {**

**if (product.id == targetId) {**

**return product;**

**}**

**}**

**return null;**

**}**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

**System.out.print("Enter number of products: ");**

**int n = sc.nextInt();**

**sc.nextLine();**

**Product[] list = new Product[n];**

**for (int i = 0; i < n; i++) {**

**System.out.println("\nProduct " + (i + 1));**

**System.out.print("ID: ");**

**int id = sc.nextInt(); sc.nextLine();**

**System.out.print("Name: ");**

**String name = sc.nextLine();**

**System.out.print("Category: ");**

**String cat = sc.nextLine();**

**list[i] = new Product(id, name, cat);**

**}**

**System.out.print("\nEnter ID to search: ");**

**int searchId = sc.nextInt();**

**long start = System.nanoTime();**

**Product result = linearSearch(list, searchId);**

**long end = System.nanoTime();**

**if (result != null)**

**System.out.println("Found: " + result);**

**else**

**System.out.println("Product not found.");**

**System.out.println("Time taken: " + (end - start) + " ns");**

**sc.close();**

**}**

**}**

**2. Using BINARY SEARCH**

**class Product implements Comparable<Product> {**

**int id;**

**String name;**

**String category;**

**public Product(int id, String name, String category) {**

**this.id = id;**

**this.name = name;**

**this.category = category;**

**}**

**@Override**

**public int compareTo(Product other) {**

**return Integer.compare(this.id, other.id);**

**}**

**@Override**

**public String toString() {**

**return "Product [ID=" + id + ", Name=" + name + ", Category=" + category + "]";**

**}**

**}**

**public class SearchFunction {**

**public static Product binarySearch(Product[] products, int targetId) {**

**int left = 0, right = products.length - 1;**

**while (left <= right) {**

**int mid = left + (right - left) / 2;**

**if (products[mid].id == targetId)**

**return products[mid];**

**else if (products[mid].id < targetId)**

**left = mid + 1;**

**else**

**right = mid - 1;**

**}**

**return null;**

**}**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

**System.out.print("Enter number of products: ");**

**int n = sc.nextInt();**

**sc.nextLine();**

**Product[] list = new Product[n];**

**for (int i = 0; i < n; i++) {**

**System.out.println("\nProduct " + (i + 1));**

**System.out.print("ID: ");**

**int id = sc.nextInt(); sc.nextLine();**

**System.out.print("Name: ");**

**String name = sc.nextLine();**

**System.out.print("Category: ");**

**String cat = sc.nextLine();**

**list[i] = new Product(id, name, cat);**

**}**

**// Must sort for binary search**

**Arrays.sort(list);**

**System.out.print("\nEnter ID to search: ");**

**int searchId = sc.nextInt();**

**long start = System.nanoTime();**

**Product result = binarySearch(list, searchId);**

**long end = System.nanoTime();**

**if (result != null)**

**System.out.println("Found: " + result);**

**else**

**System.out.println("Product not found.");**

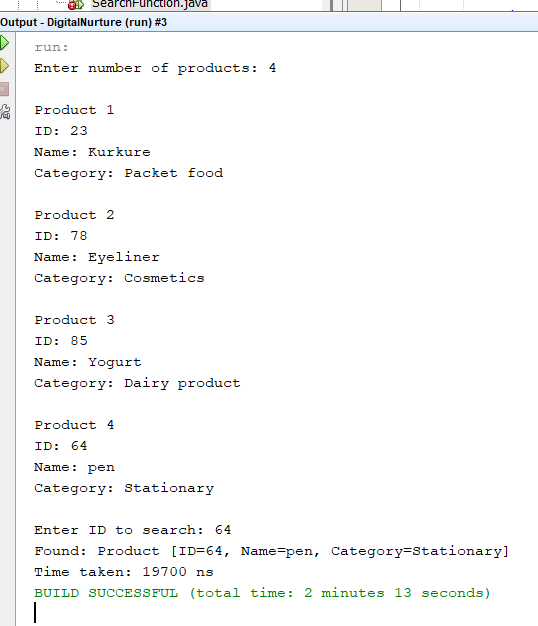
**System.out.println("Time taken: " + (end - start) + " ns");**

**sc.close();**

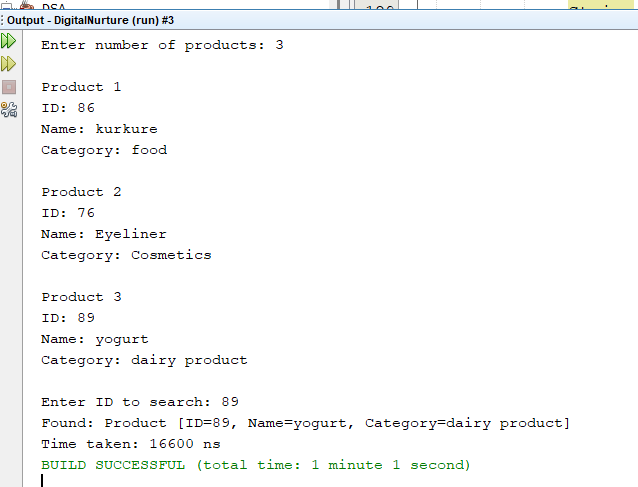
**}**

**}**

**Output of LINEAR SEARCH:**



**Output of BINARY SEARCH :**

****

**Analysis of both methods:  
  
Comparison on the basis of Time Complexity:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Search Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Time Complexity** |
| **Linear Search** | **O(1)** | **O(n/2)** | **O(n)** | **O(n)** |
| **Binary Search** | **O(1)** | **O(log n)** | **O(log n)** | **O(log n)** |

**Conclusion of the analysis:**

**When analyzing just linear and binary search for an e-commerce website, each has significant drawbacks.**

**A linear search is simple and doesn't need sorted data, which is good since e-commerce catalogs constantly change. However, it's far too slow for millions of products, potentially checking every single item. A binary search, while much faster, crucially requires all data to be perfectly sorted. For dynamic e-commerce inventory, the massive and continuous effort to keep everything sorted would make it impractical, canceling out any search speed benefits.**