**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.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **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.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

Solution:

import java.util.Arrays;

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;

}

}

class SearchEngine {

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(targetName)) {

return p;

}

}

return null;

}

public static Product binarySearch(Product[] products, String targetName) {

int low = 0, high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = products[mid].productName.compareToIgnoreCase(targetName);

if (cmp == 0)

return products[mid];

else if (cmp < 0)

low = mid + 1;

else

high = mid - 1;

}

return null;

}

}

public class Main {

public static void main(String[] args) {

Product[] products = new Product[10000];

for (int i = 0; i < products.length; i++) {

products[i] = new Product(i, "Product" + i, "Category");

}

String searchKey = "Product9999";

long startLinear = System.nanoTime();

Product found = SearchEngine.linearSearch(products, searchKey);

long endLinear = System.nanoTime();

System.out.println("Linear Search: " + (found != null ? found.productName : "Not Found"));

System.out.println("Time taken by Linear Search: " + (endLinear - startLinear) + " ns");

Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

long startBinary = System.nanoTime();

found = SearchEngine.binarySearch(products, searchKey);

long endBinary = System.nanoTime();

System.out.println("Binary Search: " + (found != null ? found.productName : "Not Found"));

System.out.println("Time taken by Binary Search: " + (endBinary - startBinary) + " ns");

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.