**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.

using System;

public class Product

{

public int ProductId { get; set; }

public string ProductName { get; set; }

public string Category { get; set; }

public Product(int id, string name, string category)

{

ProductId = id;

ProductName = name;

Category = category;

}

}

public class SearchExample

{

public static int LinearSearch(Product[] products, string name)

{

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

{

if (products[i].ProductName == name)

return i;

}

return -1;

}

public static int BinarySearch(Product[] products, string name)

{

int left = 0, right = products.Length - 1;

while (left <= right)

{

int mid = (left + right) / 2;

int cmp = string.Compare(products[mid].ProductName, name, StringComparison.OrdinalIgnoreCase);

if (cmp == 0)

return mid;

else if (cmp < 0)

left = mid + 1;

else

right = mid - 1;

}

return -1;

}

public static void Main()

{

Product[] products = {

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

new Product(102, "Laptop", "Electronics"),

new Product(103, "Shirt", "Clothing"),

new Product(104, "Watch", "Accessories"),

new Product(105, "Zebra Toy", "Toys")

};

// For binary search, sort by ProductName

Array.Sort(products, (p1, p2) => p1.ProductName.CompareTo(p2.ProductName));

string target = "Watch";

int index1 = LinearSearch(products, target);

Console.WriteLine(index1 >= 0 ? $"[Linear] Found: {products[index1].ProductName}" : "[Linear] Not Found");

int index2 = BinarySearch(products, target);

Console.WriteLine(index2 >= 0 ? $"[Binary] Found: {products[index2].ProductName}" : "[Binary] Not Found");

}

}

