WEEK – 1

E – Commerce Platform Search Function

Nipuna Amanapu

[namanapu@gitam.in](mailto:namanapu@gitam.in)

superset id : 6434842

**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:

SearchDemo.java :

import java.util.Arrays;

class Product implements Comparable<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 int compareTo(Product other) {

        return this.productName.compareToIgnoreCase(other.productName);

    }

    @Override

    public String toString() {

        return "Product{" + "ID=" + productId + ", Name='" + productName + "', Category='" + category + "'}";

    }

}

public class SearchDemo {

    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[] sortedProducts, String targetName) {

        int left = 0;

        int right = sortedProducts.length - 1;

        while (left <= right) {

            int mid = (left + right) / 2;

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

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

            else if (cmp < 0) left = mid + 1;

            else right = mid - 1;

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shoes", "Fashion"),

            new Product(103, "Watch", "Accessories"),

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

            new Product(105, "Bag", "Fashion")

        };

        System.out.println("=== Linear Search ===");

        Product result1 = linearSearch(products, "Watch");

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

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

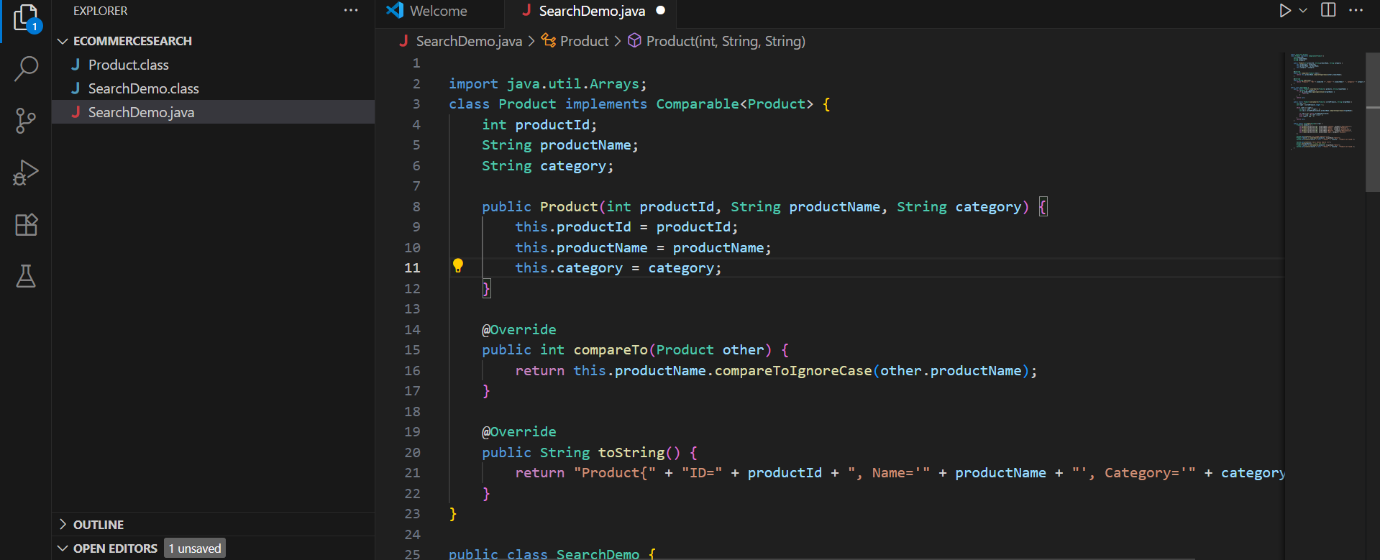
        Arrays.sort(products); // Sort the array

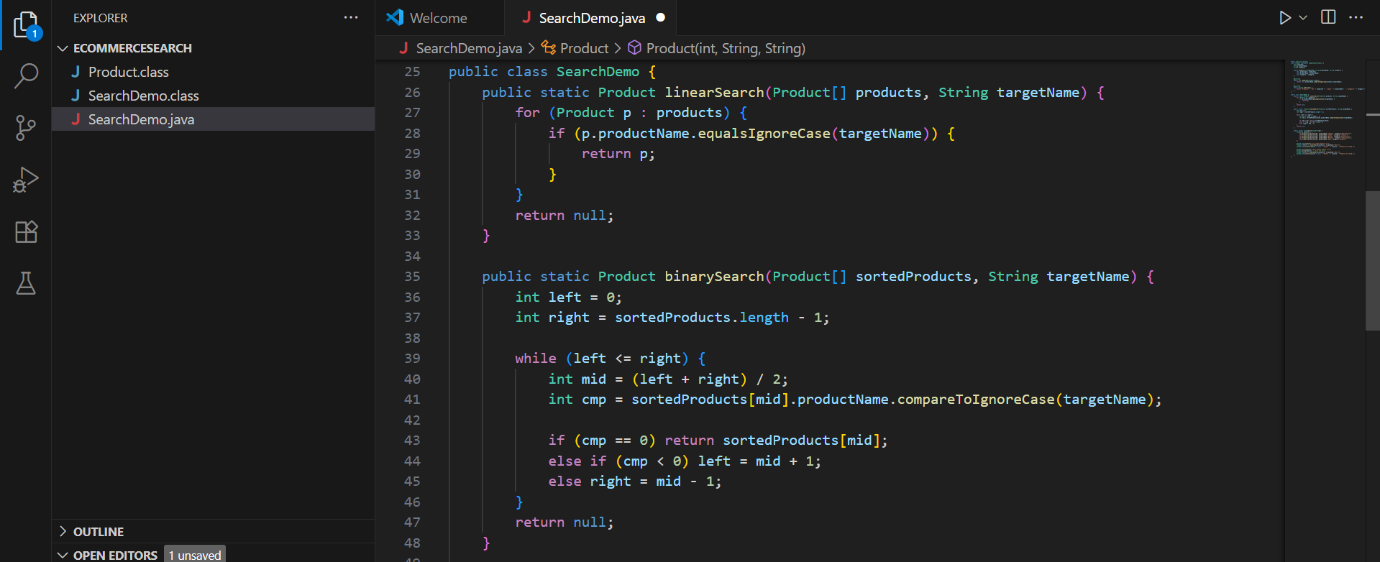
        Product result2 = binarySearch(products, "Watch");

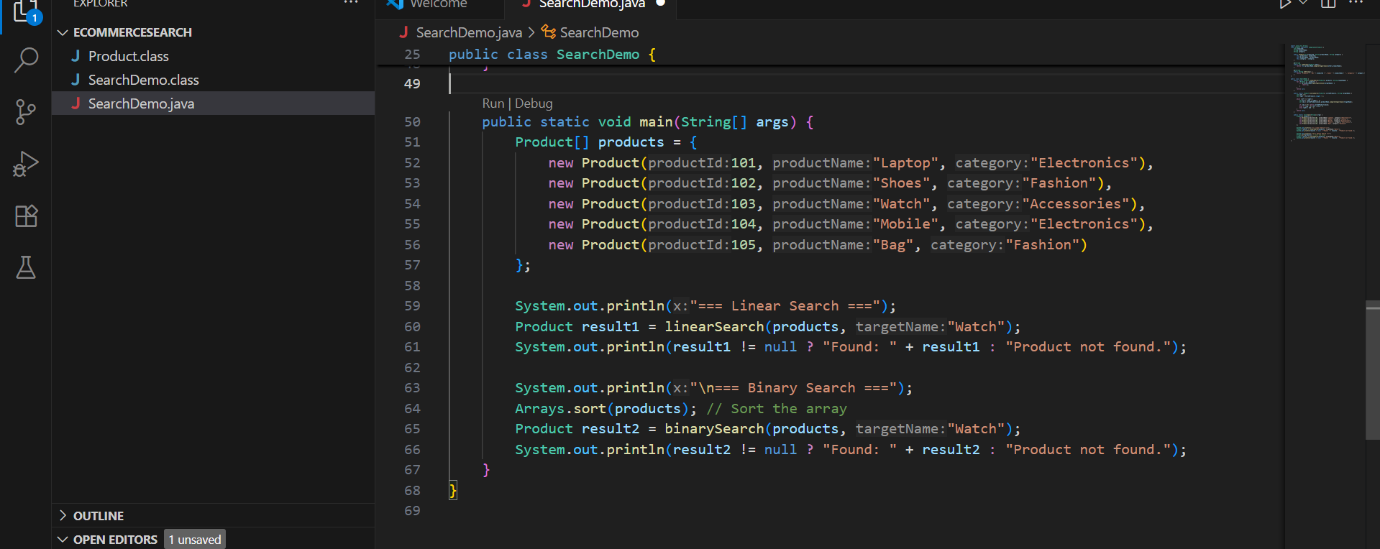
        System.out.println(result2 != null ? "Found: " + result2 : "Product not found.");

    }

}







Product.class :

class Product implements Comparable<Product> {

   int productId;

   String productName;

   String category;

   public Product(int var1, String var2, String var3) {

      this.productId = var1;

      this.productName = var2;

      this.category = var3;

   }

   public int compareTo(Product var1) {

      return this.productName.compareToIgnoreCase(var1.productName);

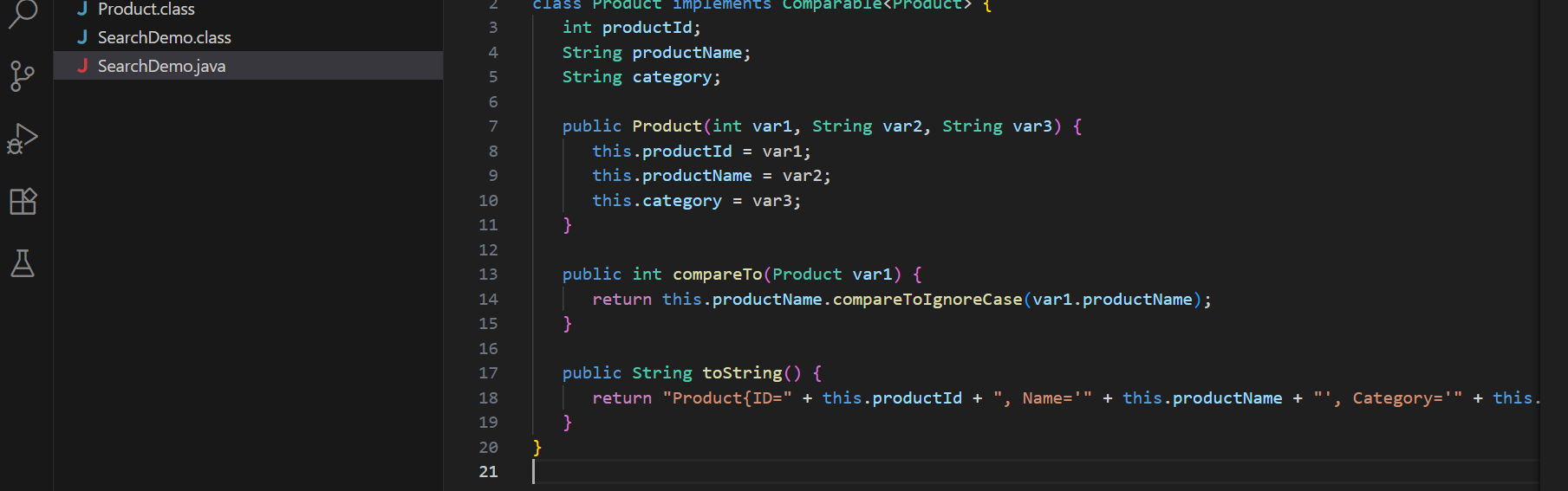
   }

   public String toString() {

      return "Product{ID=" + this.productId + ", Name='" + this.productName + "', Category='" + this.category + "'}";

   }

}



Output:

