**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:**

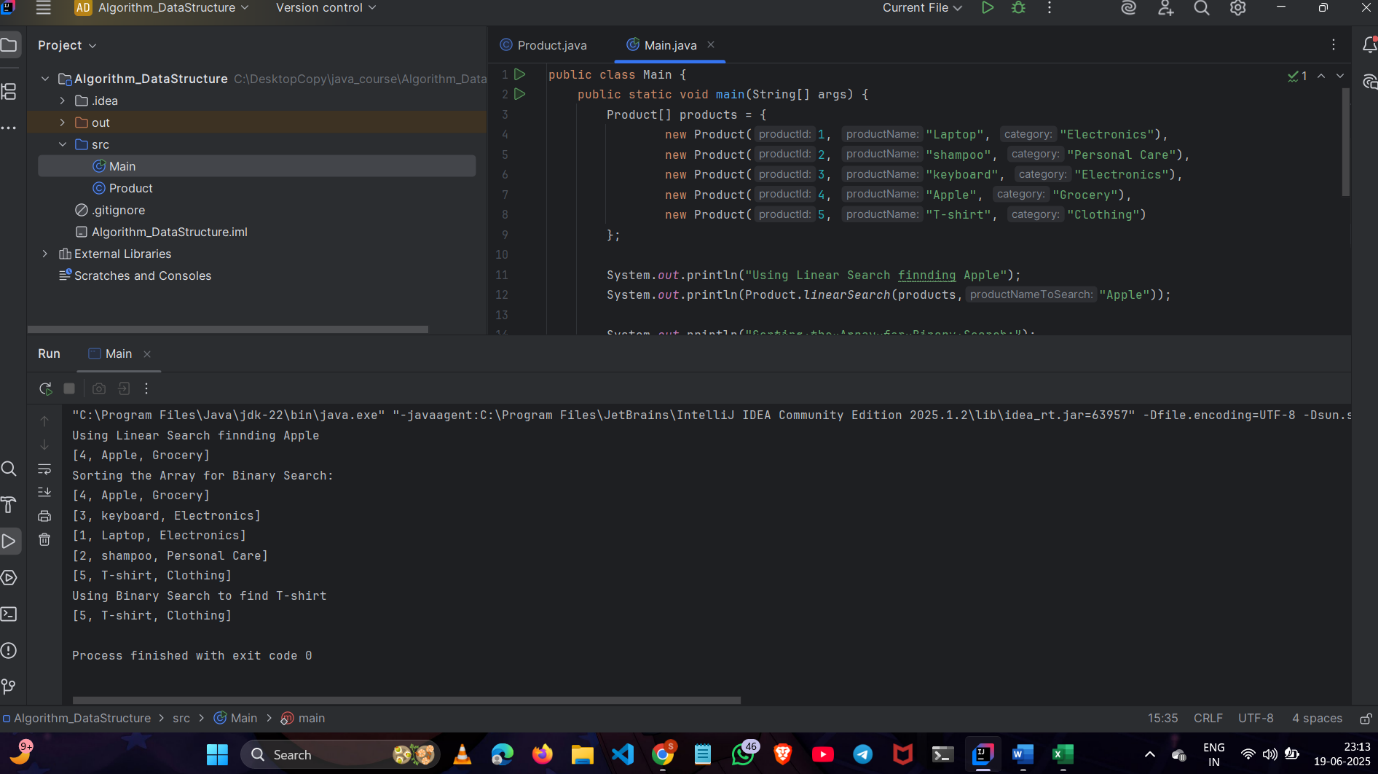
**FileName:Product.java**

package product;  
  
import java.util.Arrays;  
import java.util.Comparator;  
import java.lang.String;  
public class Product{  
 private int productId;  
 private String productName;  
 private String category;  
  
 public Product(int productId,String productName,String category){  
 this.productId = productId;  
 this.productName = productName;  
 this.category = category;  
 }  
  
 public static Product linearSearch(Product[] products,String productNameToSearch){  
  
 for(Product pId:products){  
 if(pId.productName.equalsIgnoreCase(productNameToSearch)) {  
 return pId;  
 }  
 }  
 return null;  
 }  
  
 public static Product binarySearch(Product[] products, String productNameToSearch){  
  
 int low = 0, high = products.length;  
 int mid;  
 while(low<=high){  
 mid = low + (high-low)/2;  
 int compare = products[mid].productName.compareToIgnoreCase(productNameToSearch);  
 if(compare==0){  
 return products[mid];  
 }  
 else if(compare<0){  
 low = mid+1;  
 }else{  
 high = mid-1;  
 }  
 }  
 return null;  
 }  
 public static void sorting(Product[] products){  
 Arrays.*sort*(products, Comparator.*comparing*(p -> p.productName.toLowerCase()));  
 }  
 @Override  
 public String toString() {  
 return "[" + productId + ", " + productName + ", " + category + "]";  
 }  
}

**FileName:Main.java**

package product;  
  
public class Main {  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(1, "Laptop", "Electronics"),  
 new Product(2, "shampoo", "Personal Care"),  
 new Product(3, "keyboard", "Electronics"),  
 new Product(4, "Apple", "Grocery"),  
 new Product(5, "T-shirt", "Clothing")  
 };  
  
 System.*out*.println("Using Linear Search finnding Apple");  
 System.*out*.println(Product.*linearSearch*(products,"Apple"));  
  
 System.*out*.println("Sorting the Array for Binary Search:");  
 Product.*sorting*(products);  
 for (Product p : products) {  
 System.*out*.println(p);  
 }  
  
 System.*out*.println("Using Binary Search to find T-shirt");  
 System.*out*.println(Product.*binarySearch*(products,"T-shirt"));  
  
 }  
}

**Output:**



**Exercise 7: Financial Forecasting**

**Scenario:**

**You are developing a financial forecasting tool that predicts future values based on past data.**

**Steps:**

1. **Understand Recursive Algorithms:**
   * **Explain the concept of recursion and how it can simplify certain problems.**
2. **Setup:**
   * **Create a method to calculate the future value using a recursive approach.**
3. **Implementation:**
   * **Implement a recursive algorithm to predict future values based on past growth rates.**
4. **Analysis:**
   * **Discuss the time complexity of your recursive algorithm.**
   * **Explain how to optimize the recursive solution to avoid excessive computation.**

**Solution:**

**FileName:FinancialForecast.java**

package forecast;  
  
public class FinancialForecast {  
 public static double futureValue(double presentValue, double growthRate,int years){  
 if (years==0) return presentValue;  
 return *futureValue*(presentValue + presentValue\*growthRate,growthRate,years-1);  
 }  
 public static void main(String[] args){  
 double amount = 100;  
 double growthRate = 0.5;  
 int years = 10;  
  
 System.*out*.println("To predict the total amount we make use of recursive function.");  
 System.*out*.println(*futureValue*(amount,growthRate,years));  
 }  
}

**Output:**

