**Algorithms Data Structures-Mandatory Exercises**

**Exercise 2: E-commerce Platform Search Function**

**Code:**

//Product.java

**package** ecommerce;

**public** **class** Product{

String productId;

String productName;

String category;

**public** Product(String productId, String productName, String category){

**this**.productId = productId;

**this**.productName = productName;

**this**.category = category;

}

**public** String toString(){

**return** productId + " - " + productName + " (" + category + ")";

}

}

//InventoryManager.java

**package** ecommerce;

**import** java.util.Arrays;

**import** java.util.Comparator;

**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[] products, String targetName){

Arrays.*sort*(products, Comparator.*comparing*(p -> p.productName.toLowerCase()));

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

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

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

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

**if** (cmp == 0)

**return** products[mid];

**else** **if** (cmp < 0)

left = mid + 1;

**else**

right = mid - 1;

}

**return** **null**;

}

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

Product[] products = {

**new** Product("1", "Laptop", "Electronics"),

**new** Product("2", "Phone", "Electronics"),

**new** Product("3", "Shoes", "Apparel"),

**new** Product("4", "Watch", "Accessories")

};

System.***out***.println("Linear Search:");

Product result1 = *linearSearch*(products,"Shoes");

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

System.***out***.println("\nBinary Search:");

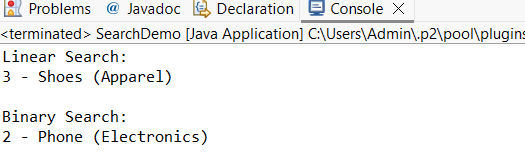
Product result2 = *binarySearch*(products,"Phone");

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

}

}

**Output:**

****

**Exercise 7: Financial Forecasting**

**Code:**

//FinancialForecaster.java

**package** Financial;

**public** **class** FinancialForecaster{

**public** **static** **double** calculateFutureValue(**double** principal, **double** rate, **int** years){

**if**(years == 0){

**return** principal;

}

**return** *calculateFutureValue*(principal \* (1 + rate), rate, years - 1);

}

**public** **static** **double** calculateFutureValueIterative(**double** principal, **double** rate, **int** years){

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

principal \*= (1 + rate);

}

**return** principal;

}

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

**double** initialAmount = 10000;

**double** annualGrowthRate = 0.10;

**int** years = 5;

**double** recursiveValue = *calculateFutureValue*(initialAmount, annualGrowthRate, years);

**double** iterativeValue = *calculateFutureValueIterative*(initialAmount, annualGrowthRate, years);

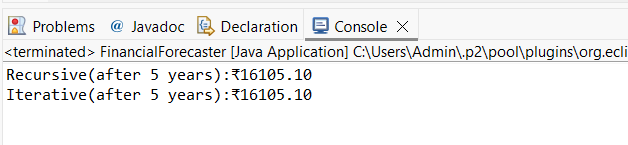
System.***out***.printf("Recursive(after %d years):₹%.2f\n", years, recursiveValue);

System.***out***.printf("Iterative(after %d years):₹%.2f\n", years, iterativeValue);

}

}

**Output:**

****