**Algorithms\_Data Structures**

1. **Ecommerce Platform Search Function**

**ProductSearch.java:**

package Ecommerce\_Platform\_Search;

import java.util.Arrays;

import java.util.Comparator;

public class ProductSearch {

int productId;

String productName;

String category;

ProductSearch(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

static ProductSearch linearSearch(ProductSearch[] products, String name) {

for (ProductSearch p : products) {

if (p.productName.equals(name)) return p;

}

return null;

}

static ProductSearch binarySearch(ProductSearch[] products, String name) {

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

while (low <= high) {

int mid = (low + high) / 2;

int cmp = products[mid].productName.compareTo(name);

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

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

else high = mid - 1;

}

return null;

}

public static void main(String[] args) {

ProductSearch[] products = {

new ProductSearch(1, "Shoes", "Fashion"),

new ProductSearch(2, "Phone", "Electronics"),

new ProductSearch(3, "Watch", "Accessories")

};

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

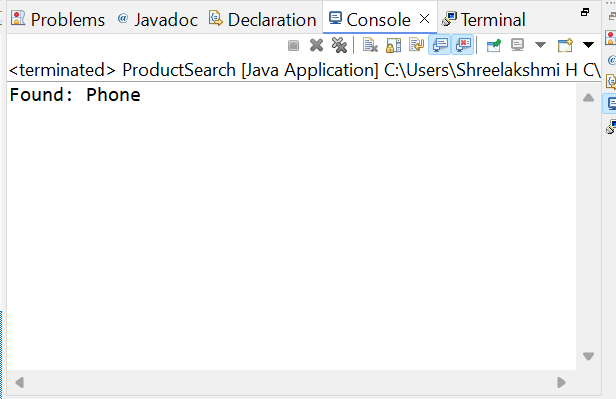
ProductSearch result = binarySearch(products, "Phone");

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

}

}

**Output:**



1. **Financial Forecasting**

**Forecast.java:**

**package** Forecasting;

**public** **class** Forecast {

**static** **double** futureValue(**double** amount, **double** rate, **int** years) {

**if** (years == 0) **return** amount;

**return** (1 + rate) \* *futureValue*(amount, rate, years - 1);

}

**static** **double** futureValueOptimized(**double** amount, **double** rate, **int** years) {

**double** result = amount;

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

result \*= (1 + rate);

}

**return** result;

}

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

System.***out***.println("Future Value: " + *futureValueOptimized*(1000, 0.05, 5));

}

}

Output:

