**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.

**Code for implementing the E-commerce Platform Search Function:**

**Initially, we will have a ‘Product’ class. The Product.java is given as,  
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** **int** getProductId() {

**return** productId;

}

**public** String getProductName() {

**return** productName;

}

**public** String getCategory() {

**return** category;

}

@Override

**public** String toString() {

**return** "Product{" +

"productId=" + productId +

", productName='" + productName + '\'' +

", category='" + category + '\'' +

'}';

}

}

Next, we will have to create a class called ‘SearchUtil.java’ which consists of all the search methods that are required for implementing the scenario.

**SearchUtil.java:  
public** **class** SearchUtil {

**public** **static** Product linearSearch(Product[] products, String name) {

**for** (Product product : products) {

**if** (product.getProductName().equalsIgnoreCase(name)) {

**return** product;

}

}

**return** **null**;

}

**public** **static** Product binarySearch(Product[] products, String name) {

**int** low = 0;

**int** high = products.length - 1;

**while** (low <= high) {

**int** mid = (low + high) / 2;

**int** comparison = products[mid].getProductName().compareToIgnoreCase(name);

**if** (comparison == 0) {

**return** products[mid];

} **else** **if** (comparison < 0) {

low = mid + 1;

} **else** {

high = mid - 1;

}

}

**return** **null**;

}

}

**SearchTest.java:**

**import** java.util.Arrays;

**public** **class** SearchTest {

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

Product[] products = {

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

**new** Product(2, "Shirt", "Clothing"),

**new** Product(3, "Book", "Education"),

**new** Product(4, "Phone", "Electronics")

};

// Test Linear Search

Product found1 = SearchUtil.*linearSearch*(products, "Phone");

System.***out***.println("Linear Search: " + (found1 != **null** ? found1 : "Product not found"));

// Sort products by name for binary search

Arrays.*sort*(products, (a, b) -> a.getProductName().compareToIgnoreCase(b.getProductName()));

// Test Binary Search

Product found2 = SearchUtil.*binarySearch*(products, "Phone");

System.***out***.println("Binary Search: " + (found2 != **null** ? found2 : "Product not found"));

}

}

**Screenshot of the output:  
A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 7: Financial Forecasting**

**Scenario:**

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

**The code for implementing the above scenario is given below:**

**ForecastUtil.java:  
public** **class** ForecastUtil {

**public** **static** **double** forecastFutureValue(**double** presentValue, **double** growthRate, **int** years) {

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

**return** presentValue;

}

**return** *forecastFutureValue*(presentValue, growthRate, years - 1) \* (1 + growthRate);

}

}

**ForcastTest.java:**

**public** **class** ForecastTest {

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

**double** presentValue = 1000.0; // Starting value

**double** annualGrowthRate = 0.05; // 5% growth

**int** years = 5;

**double** futureValue = ForecastUtil.*forecastFutureValue*(presentValue, annualGrowthRate, years);

System.***out***.printf("Future value after %d years: %.2f%n", years, futureValue);

}

}

**Screenshot of the output:**

**A screenshot of a computer

AI-generated content may be incorrect.**