**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.

**Product.java:**

package com.ecommerce.search;

public class 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 String toString() {

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

}

}

**SearchDemo.java**

package com.ecommerce.search;

import java.util.Arrays;

import java.util.Comparator;

public class SearchDemo {

public static void main(String[] args) {

Product[] products = {

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

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

new Product(103, "Smartphone", "Electronics"),

new Product(104, "Desk", "Furniture"),

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

};

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

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

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

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

System.***out***.println("\n Binary Search for 'Desk':");

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

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

}

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) {

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

while (left <= right) {

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

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

if (comparison == 0) {

return products[mid];

} else if (comparison < 0) {

right = mid - 1;

} else {

left = mid + 1;

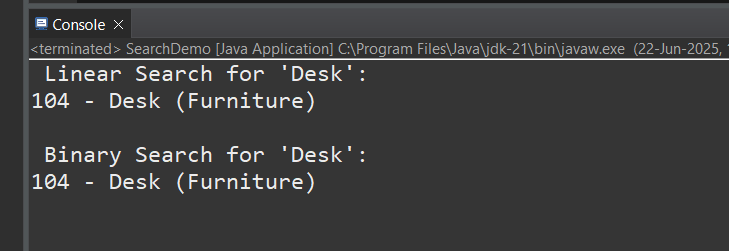
}

}

return null;

}

}

**OUTPUT:**

Exercise 7: Financial Forecasting

Scenario:

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

**FinForecast.java:**

package com.financial.forecasting;

public class FinForecast {

// Recursive method to calculate future value

public static double futureValue(double principal, double rate, int periods) {

if (periods == 0) {

return principal; // Base case

}

// Recursive call

return *futureValue*(principal, rate, periods - 1) \* (1 + rate);

}

public static void main(String[] args) {

double principal = 1000.0; // Initial investment

double rate = 0.05; // 5% growth rate per period

int periods = 5; // Number of periods

double result = *futureValue*(principal, rate, periods);

System.***out***.printf("Future Value after %d periods: %.2f%n", periods, result);

}

}

**OUTPUT:**

