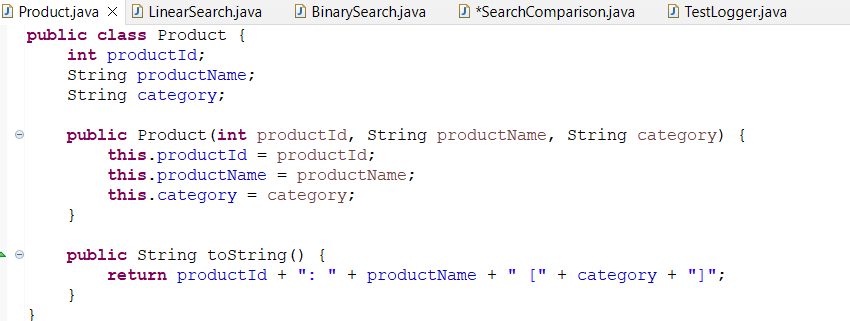
**Exercise 2: E-commerce Platform Search Function**

**Scenario:**

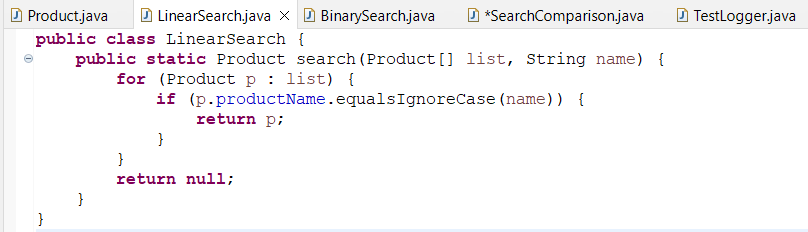
You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**SOLUTION:**

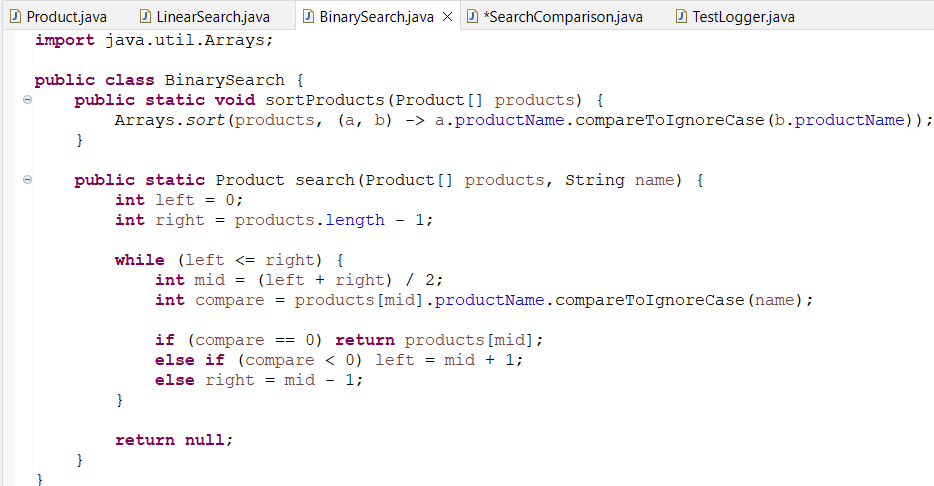
**Product.java**

****

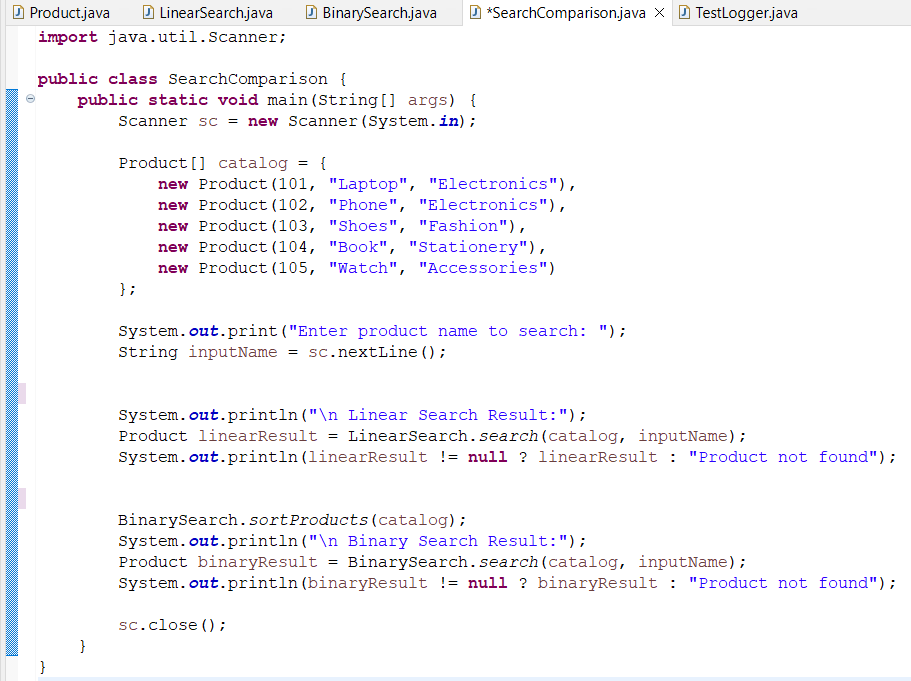
**LinearSearch.java**

****

**BinarySearch.java**

****

**SearchComparision.java**

****

**Code:**

**Step 1: Product.java**

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;

}

public String toString() {

return productId + ": " + productName + " [" + category + "]";

}

}

**Step 2: LinearSearch.java**

public class LinearSearch {

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

}

**Step 3: BinarySearch.java**

import java.util.Arrays;

public class BinarySearch {

public static void sortProducts(Product[] products) {

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

}

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

}

**Step 4: SearchComparison.java**

import java.util.Scanner;

public class SearchComparison {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Product[] catalog = {

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

new Product(102, "Phone", "Electronics"),

new Product(103, "Shoes", "Fashion"),

new Product(104, "Book", "Stationery"),

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

};

System.out.print("Enter product name to search: ");

String inputName = sc.nextLine();

System.out.println("\n Linear Search Result:");

Product linearResult = LinearSearch.search(catalog, inputName);

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

BinarySearch.sortProducts(catalog);

System.out.println("\n Binary Search Result:");

Product binaryResult = BinarySearch.search(catalog, inputName);

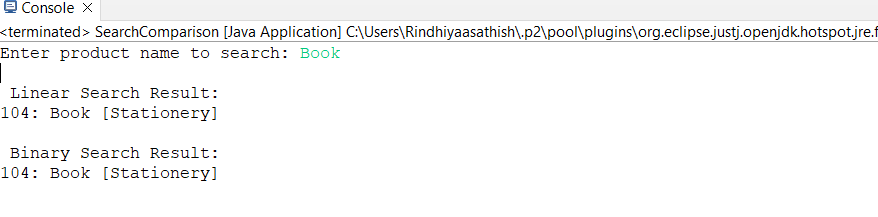
System.out.println(binaryResult != null ? binaryResult : "Product not found");

sc.close();

}

}

**Output:**

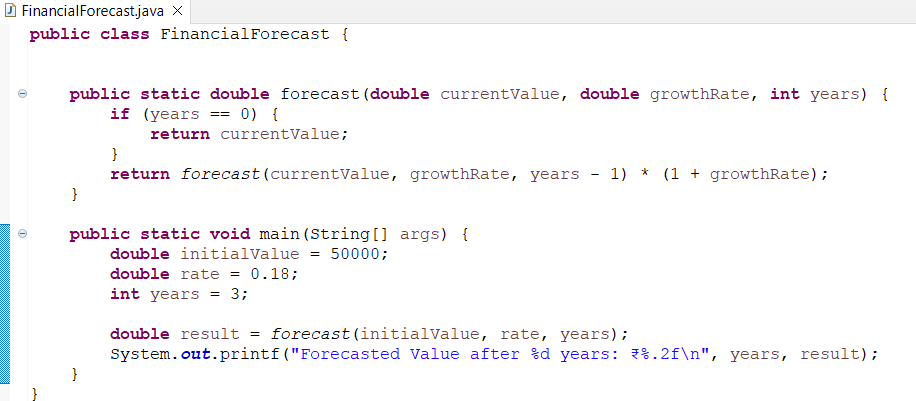
****

**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Solution:**

****

**Code**

public class FinancialForecast {

public static double forecast(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

return forecast(currentValue, growthRate, years - 1) \* (1 + growthRate);

}

public static void main(String[] args) {

double initialValue = 10000;

double rate = 0.08;

int years = 5;

double result = forecast(initialValue, rate, years);

System.out.printf("Forecasted Value after %d years: ₹%.2f\n", years, result);

}

}

**Output**

