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

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

***CODE:***

**File: Main.java**

public class Main

{

public static void main(String[] args)

{

Product[] products = {

new Product(3, "Laptop", "Electronics"),

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

new Product(5, "Book", "Stationery"),

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

new Product(4, "T-Shirt", "Fashion")

};

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

Product result1 = SearchEngine.linearSearch(products, 5);

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

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

SearchEngine.sortProductsById(products);

Product result2 = SearchEngine.binarySearch(products, 5);

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

}

}

**File: 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;

}

@Override

public String toString()

{

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

**File: SearchEngine.java**

import java.util.Arrays;

import java.util.Comparator;

public class SearchEngine

{

public static Product linearSearch(Product[] products, int id)

{

for (Product p : products)

{

if (p.productId == id)

{

return p;

}

}

return null;

}

public static Product binarySearch(Product[] products, int id)

{

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

while (low <= high)

{

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

if (products[mid].productId == id)

{

return products[mid];

}

else if (products[mid].productId < id)

{

low = mid + 1;

}

else

{

high = mid - 1;

}

}

return null;

}

public static void sortProductsById(Product[] products)

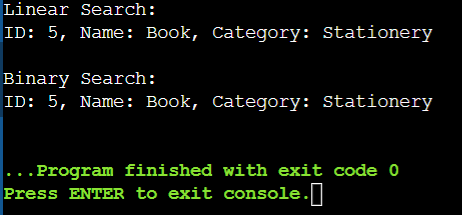
{

Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

}

}

***OUTPUT:***

****

**Exercise 7: Financial Forecasting**

A financial forecasting tool that predicts future values based on past data.

***CODE:***

**File: ForecastTool.java**

public class ForecastTool

{

public static double calculateFutureValue(double currentValue, double growthRate, int years)

{

if (years == 0)

{

return currentValue;

}

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

}

public static double calculateFutureValueMemo(double currentValue, double growthRate, int years, double[] memo)

{

if (years == 0) return currentValue;

if (memo[years] != 0) return memo[years];

memo[years] = calculateFutureValueMemo(currentValue, growthRate, years - 1, memo) \* (1 + growthRate);

return memo[years];

}

public static void main(String[] args)

{

double currentValue = 1000;

double growthRate = 0.05;

int years = 10;

double result = calculateFutureValue(currentValue, growthRate, years);

System.out.println("Future value after " + years + " years (recursive): ₹" + result);

double[] memo = new double[years + 1];

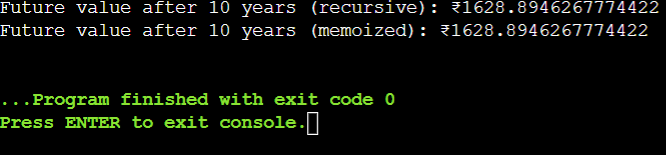
double optimizedResult = calculateFutureValueMemo(currentValue, growthRate, years, memo);

System.out.println("Future value after " + years + " years (memoized): ₹" + optimizedResult);

}

}

***OUTPUT:***

****