Data structures and Algorithms

Exercise 2: E-commerce Platform Search Function

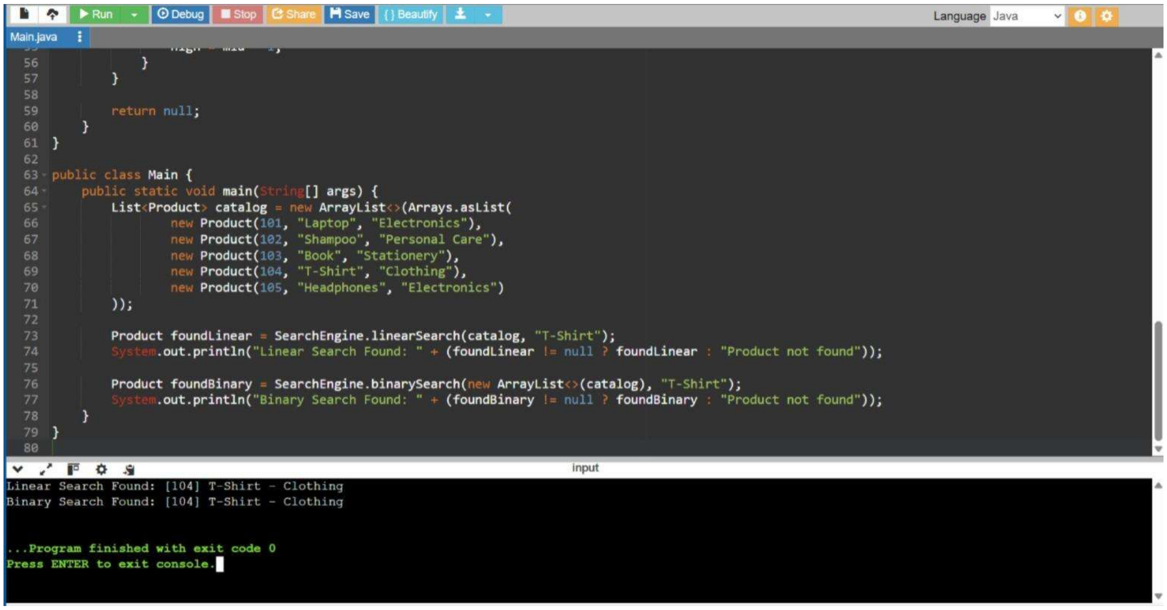
Scenario:

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

Code:

```
import java.util.*;

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 "[" + productId + "] " + productName + " - " + category;
    }
}

class SearchEngine {
    public static Product linearSearch(List<Product> products, String name) {
        return products.stream()
                .filter(p -> p.getProductName().equalsIgnoreCase(name))
                .findFirst()
                .orElse(null);
    }

    public static Product binarySearch(List<Product> products, String name) {
        products.sort(Comparator.comparing(Product::getProductName,
String.CASE_INSENSITIVE_ORDER));
        int low = 0;
        int high = products.size() - 1;

        while (low <= high) {
            int mid = (low + high) / 2;
            Product midProduct = products.get(mid);
            int cmp = midProduct.getProductName().compareToIgnoreCase(name);

            if (cmp == 0) {
                return midProduct;
            } else if (cmp < 0) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return null;
    }
}

public class Main {
    public static void main(String[] args) {
        List<Product> catalog = new ArrayList<>(Arrays.asList(
                new Product(101, "Laptop", "Electronics"),
                new Product(102, "Shampoo", "Personal Care"),
                new Product(103, "Book", "Stationery"),
                new Product(104, "T-Shirt", "Clothing"),
                new Product(105, "Headphones", "Electronics")
        ));

        Product foundLinear = SearchEngine.linearSearch(catalog, "T-Shirt");
        System.out.println("Linear Search Found: " + (foundLinear != null ? foundLinear :
"Product not found"));

        Product foundBinary = SearchEngine.binarySearch(new ArrayList<>(catalog), "T-
Shirt");
        System.out.println("Binary Search Found: " + (foundBinary != null ? foundBinary :
"Product not found"));
    }
}
```

Output:



```java
                high = mid - 1;
            }
        }

        return null;
    }
}

public class Main {
    public static void main(String[] args) {
        List<Product> catalog = new ArrayList<>(Arrays.asList(
                new Product(101, "Laptop", "Electronics"),
                new Product(102, "Shampoo", "Personal Care"),
                new Product(103, "Book", "Stationery"),
                new Product(104, "T-Shirt", "Clothing"),
                new Product(105, "Headphones", "Electronics")
        ));

        Product foundLinear = SearchEngine.linearSearch(catalog, "T-Shirt");
        System.out.println("Linear Search Found: " + (foundLinear != null ? foundLinear : "Product not found"));

        Product foundBinary = SearchEngine.binarySearch(new ArrayList<>(catalog), "T-Shirt");
        System.out.println("Binary Search Found: " + (foundBinary != null ? foundBinary : "Product not found"));
    }
}
```

```
Linear Search Found: [104] T-Shirt - Clothing
Binary Search Found: [104] T-Shirt - Clothing


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercise 7: Financial Forecasting

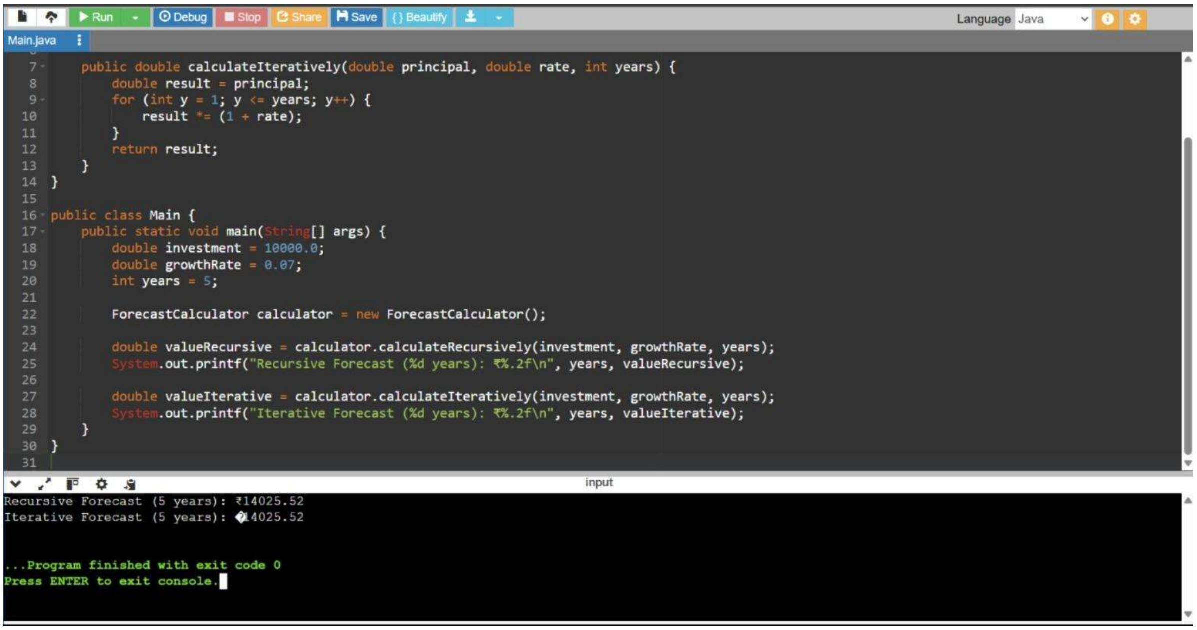
Scenario:

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

Code:

```
class ForecastCalculator {

   public double calculateRecursively(double principal, double rate, int years) {
      return (years == 0) ? principal : calculateRecursively(principal, rate, years - 1) * (1 +
rate);
   }

   public double calculateIteratively(double principal, double rate, int years) {
      double result = principal;
      for (int y = 1; y <= years; y++) {
         result *= (1 + rate);
      }
      return result;
   }
}

public class Main {
   public static void main(String[] args) {
      double investment = 10000.0;
      double growthRate = 0.07;
      int years = 5;

      ForecastCalculator calculator = new ForecastCalculator();

      double valueRecursive = calculator.calculateRecursively(investment, growthRate, years);
      System.out.printf("Recursive Forecast (%d years): ₹%.2f\n", years, valueRecursive);

      double valueIterative = calculator.calculateIteratively(investment, growthRate, years);
      System.out.printf("Iterative Forecast (%d years): ₹%.2f\n", years, valueIterative);
   }
}
```

Output:



```java
    public double calculateIteratively(double principal, double rate, int years) {
        double result = principal;
        for (int y = 1; y <= years; y++) {
            result *= (1 + rate);
        }
        return result;
    }
}

public class Main {
    public static void main(String[] args) {
        double investment = 10000.0;
        double growthRate = 0.07;
        int years = 5;

        ForecastCalculator calculator = new ForecastCalculator();

        double valueRecursive = calculator.calculateRecursively(investment, growthRate, years);
        System.out.printf("Recursive Forecast (%d years): ₹%.2f\n", years, valueRecursive);

        double valueIterative = calculator.calculateIteratively(investment, growthRate, years);
        System.out.printf("Iterative Forecast (%d years): ₹%.2f\n", years, valueIterative);
    }
}
```

```
Recursive Forecast (5 years): ₹14025.52
Iterative Forecast (5 years): �14025.52


...Program finished with exit code 0
Press ENTER to exit console.
```