Data structures and Algorithms

Exercise 2: E-commerce Platform Search Function

Scenario:

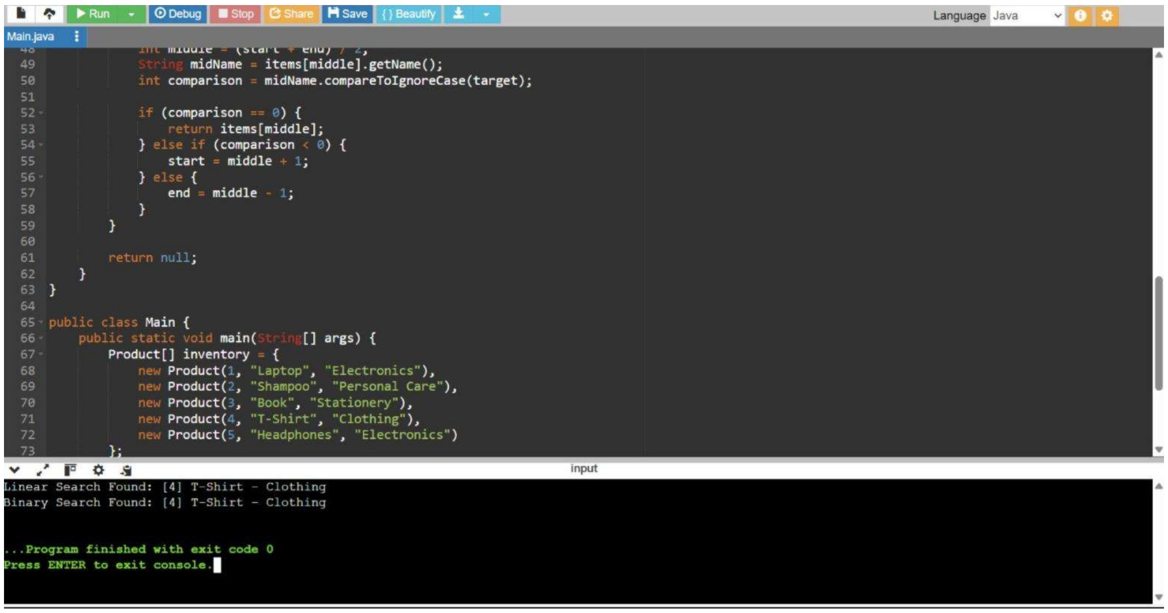
You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

Code:

```
import java.util.Arrays;
import java.util.Comparator;

class Product {
   private int id;
   private String name;
   private String category;

   public Product(int id, String name, String category) {
      this.id = id;
      this.name = name;
      this.category = category;
   }

   public int getId() {
      return id;
   }

   public String getName() {
      return name;
   }

   public String getCategory() {
      return category;
   }

   public String toString() {
      return "[" + id + "] " + name + " - " + category;
   }
}

class SearchService {
   public Product searchByLinear(Product[] items, String target) {
      for (int i = 0; i < items.length; i++) {
         if (items[i].getName().equalsIgnoreCase(target)) {
            return items[i];
         }
```

```
        }
        return null;
    }

    public Product searchByBinary(Product[] items, String target) {
        Arrays.sort(items, Comparator.comparing(Product::getName,
String.CASE_INSENSITIVE_ORDER));
        int start = 0;
        int end = items.length - 1;

        while (start <= end) {
            int middle = (start + end) / 2;
            String midName = items[middle].getName();
            int comparison = midName.compareToIgnoreCase(target);

            if (comparison == 0) {
                return items[middle];
            } else if (comparison < 0) {
                start = middle + 1;
            } else {
                end = middle - 1;
            }
        }

        return null;
    }
}

public class Main {
    public static void main(String[] args) {
        Product[] inventory = {
            new Product(1, "Laptop", "Electronics"),
            new Product(2, "Shampoo", "Personal Care"),
            new Product(3, "Book", "Stationery"),
            new Product(4, "T-Shirt", "Clothing"),
            new Product(5, "Headphones", "Electronics")
        };

        SearchService engine = new SearchService();

        Product result1 = engine.searchByLinear(inventory, "T-Shirt");
        System.out.println("Linear Search Found: " + (result1 != null ? result1 : "Product not
found"));

        Product result2 = engine.searchByBinary(inventory, "T-Shirt");
        System.out.println("Binary Search Found: " + (result2 != null ? result2 : "Product not
found"));
    }
}
```

Output:



```
            int middle = (start + end) / 2;
            String midName = items[middle].getName();
            int comparison = midName.compareToIgnoreCase(target);

            if (comparison == 0) {
                return items[middle];
            } else if (comparison < 0) {
                start = middle + 1;
            } else {
                end = middle - 1;
            }
        }

        return null;
    }
}

public class Main {
    public static void main(String[] args) {
        Product[] inventory = {
            new Product(1, "Laptop", "Electronics"),
            new Product(2, "Shampoo", "Personal Care"),
            new Product(3, "Book", "Stationery"),
            new Product(4, "T-Shirt", "Clothing"),
            new Product(5, "Headphones", "Electronics")
        };
```

```
Linear Search Found: [4] T-Shirt - Clothing
Binary Search Found: [4] T-Shirt - Clothing


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

Exercise 7: Financial Forecasting

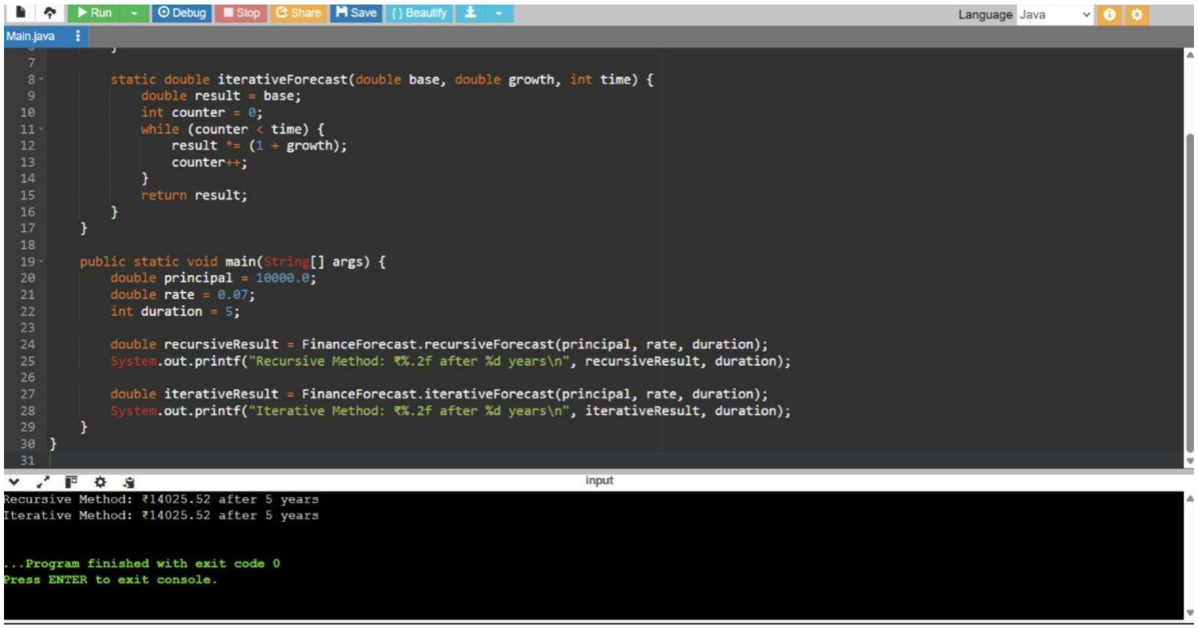
Scenario:

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

Code:

```
public class Main {

    static class FinanceForecast {
        static double recursiveForecast(double base, double growth, int time) {
            return time == 0 ? base : recursiveForecast(base, growth, time - 1) * (1 + growth);
        }

        static double iterativeForecast(double base, double growth, int time) {
            double result = base;
            int counter = 0;
            while (counter < time) {
                result *= (1 + growth);
                counter++;
            }
            return result;
        }
    }

    public static void main(String[] args) {
        double principal = 10000.0;
        double rate = 0.07;
        int duration = 5;

        double recursiveResult = FinanceForecast.recursiveForecast(principal, rate, duration);
        System.out.printf("Recursive Method: ₹%.2f after %d years\n", recursiveResult,
duration);

        double iterativeResult = FinanceForecast.iterativeForecast(principal, rate, duration);
        System.out.printf("Iterative Method: ₹%.2f after %d years\n", iterativeResult, duration);
    }
}
```

Output:



```java
        }

        static double iterativeForecast(double base, double growth, int time) {
            double result = base;
            int counter = 0;
            while (counter < time) {
                result *= (1 + growth);
                counter++;
            }
            return result;
        }
    }

    public static void main(String[] args) {
        double principal = 10000.0;
        double rate = 0.07;
        int duration = 5;

        double recursiveResult = FinanceForecast.recursiveForecast(principal, rate, duration);
        System.out.printf("Recursive Method: ₹%.2f after %d years\n", recursiveResult, duration);

        double iterativeResult = FinanceForecast.iterativeForecast(principal, rate, duration);
        System.out.printf("Iterative Method: ₹%.2f after %d years\n", iterativeResult, duration);
    }
}
```

```
Recursive Method: ₹14025.52 after 5 years
Iterative Method: ₹14025.52 after 5 years


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