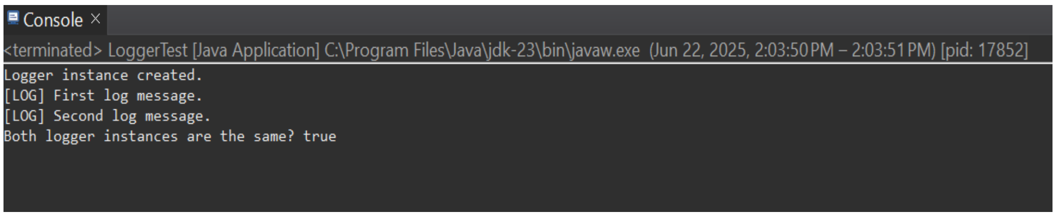
Design Patterns and Principles

Exercise 1: Implementing the Singleton Pattern

```
    public class Logger {
        private Logger() {
            System.out.println("Logger instance created.");
        }
    initialization)
        public static Logger getInstance() {
            if (instance == null) {
                instance = new Logger();
            }
            return instance;
        }
        public void log(String message) {
            System.out.println("[LOG] " + message);
        }
    }

    public class LoggerTest {
        public static void main(String[] args) {
            Logger logger1 = Logger.getInstance();
            logger1.log("First log message.");

            Logger logger2 = Logger.getInstance();
            logger2.log("Second log message.");

            // Verify that both logger instances are the same
            System.out.println("Both logger instances are the same? " +
    (logger1 == logger2));
        }
}
```

- **OUTPUT:**


Console ×

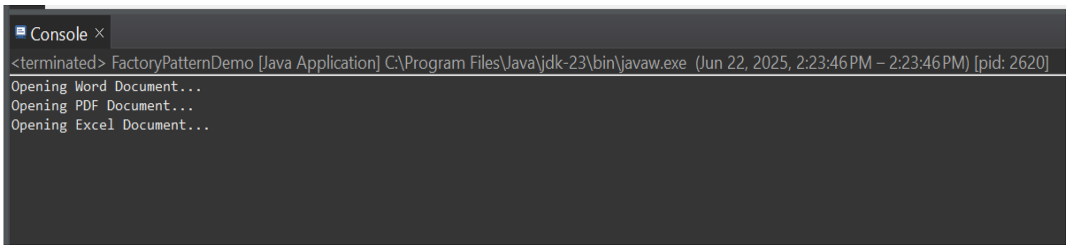
```
<terminated> LoggerTest [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (Jun 22, 2025, 2:03:50 PM – 2:03:51 PM) [pid: 17852]
Logger instance created.
[LOG] First log message.
[LOG] Second log message.
Both logger instances are the same? true
```

Exercise 2: Implementing the Factory Method Pattern

```
public interface Document {
    void open();
}

public class WordDocument implements Document {
    public void open() {
        System.out.println("Opening Word Document...");
    }
}

public class PdfDocument implements Document {
    public void open() {
        System.out.println("Opening PDF Document...");
    }
}

public class ExcelDocument implements Document {
    public void open() {
        System.out.println("Opening Excel Document...");
    }
}

public abstract class DocumentFactory {
    public abstract Document createDocument();
}
```

```
public class WordDocumentFactory extends DocumentFactory {
   public Document createDocument() {
      return new WordDocument();
   }
}

public class PdfDocumentFactory extends DocumentFactory {
   public Document createDocument() {
      return new PdfDocument();
   }
}

public class ExcelDocumentFactory extends DocumentFactory {
   public Document createDocument() {
      return new ExcelDocument();
   }
}

public class FactoryPatternDemo {
   public static void main(String[] args) {
      DocumentFactory wordFactory = new
WordDocumentFactory();
      Document wordDoc = wordFactory.createDocument();
      wordDoc.open();

      DocumentFactory pdfFactory = new
PdfDocumentFactory();
      Document pdfDoc = pdfFactory.createDocument();
      pdfDoc.open();

      DocumentFactory excelFactory = new
ExcelDocumentFactory();
      Document excelDoc = excelFactory.createDocument();
      excelDoc.open();
   }
}
```

- **OUTPUT:**



```
<terminated> FactoryPatternDemo [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (Jun 22, 2025, 2:23:46 PM – 2:23:46 PM) [pid: 2620]
Opening Word Document...
Opening PDF Document...
Opening Excel Document...
```

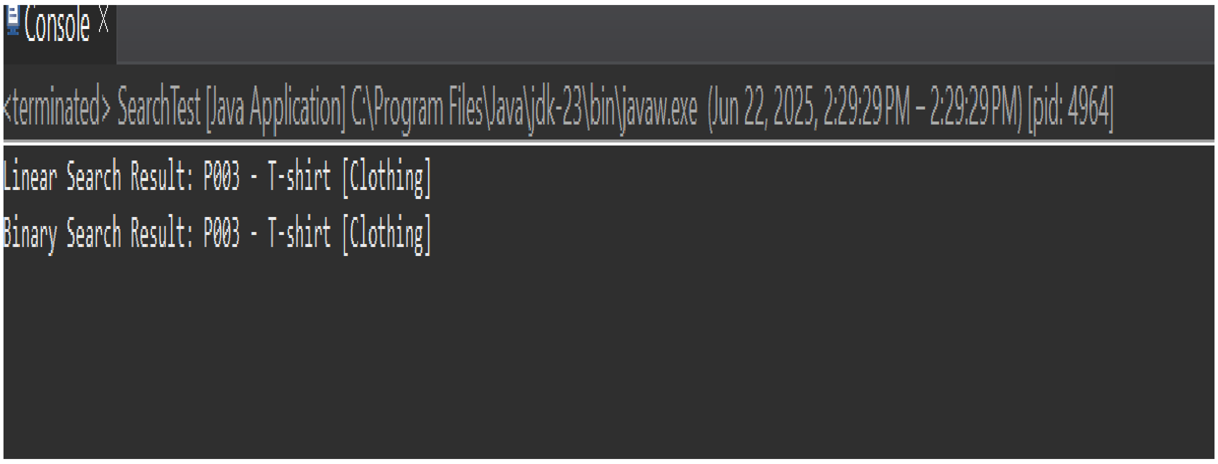
Algorithms Data Structures

Exercise 2: E-commerce Platform Search Function

```
public class Product {
    private String productId;
    private String productName;
    private String category;

    public Product(String productId, String productName, String
category) {
        this.productId = productId;
        this.productName = productName;
        this.category = category;
    }

    public String getProductName() {
        return productName;
    }

    public String getProductId() {
        return productId;
    }

    public String getCategory() {
        return category;
    }

    @Override
    public String toString() {
        return productId + " - " + productName + " [" + category + "]";
    }
}


public class SearchService {

    public static Product linearSearch(Product[] products, String
targetName) {
        for (Product product : products) {
```

```
        if(product.getProductName().equalsIgnoreCase(targetName))
        {
            return product;
          }
        }
        return null;
    }

    public static Product binarySearch(Product[] products, String
targetName) {
        int left = 0;
        int right = products.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int cmp =
products[mid].getProductName().compareToIgnoreCase(targetName);

            if (cmp == 0) return products[mid];
            if (cmp < 0) left = mid + 1;
            else right = mid - 1;
        }
        return null;
    }
}


public class SearchTest {
    public static void main(String[] args) {
        Product[] products = {
            new Product("P001", "iPhone", "Electronics"),
            new Product("P002", "MacBook", "Electronics"),
            new Product("P003", "T-shirt", "Clothing"),
            new Product("P004", "Shoes", "Footwear"),
            new Product("P005", "Headphones", "Electronics")
        };

        Product result1 = SearchService.linearSearch(products, "T-
shirt");
        System.out.println("Linear Search Result: " + result1);
        Arrays.sort(products, (a, b) ->
a.getProductName().compareToIgnoreCase(b.getProductName()));


        Product result2 = SearchService.binarySearch(products, "T-
shirt");
        System.out.println("Binary Search Result: " + result2);
    }
}
```

- **OUTPUT:**

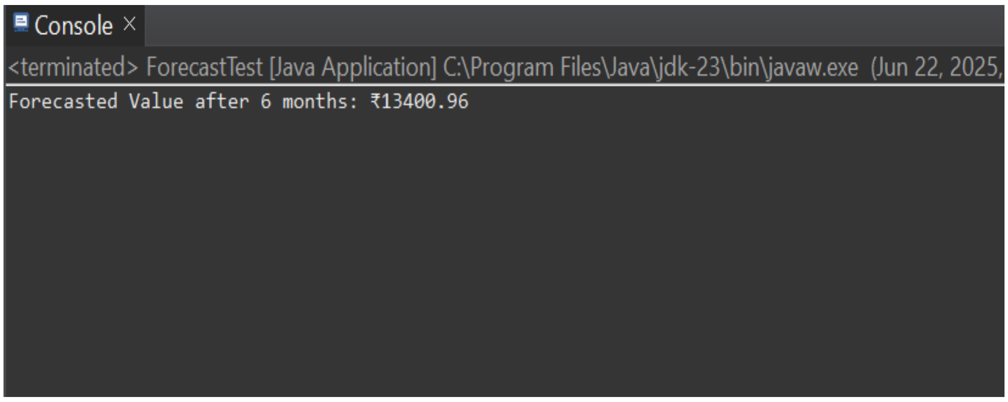


```
<terminated> SearchTest [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (Jun 22, 2025, 2:29:29 PM – 2:29:29 PM) [pid: 4964]
Linear Search Result: P003 - T-shirt [Clothing]
Binary Search Result: P003 - T-shirt [Clothing]
```

Exercise 7: Financial Forecasting

```
public class FinancialForecast {
   public static double forecastValue(double currentValue,
double growthRate, int months) {
      if (months == 0) {
         return currentValue;
      }
      return forecastValue(currentValue, growthRate, months - 1)
* (1 + growthRate);
   }
}

public class ForecastTest {
   public static void main(String[] args) {
      double initialValue = 10000.0;
      double monthlyGrowthRate = 0.05;
      int forecastMonths = 6;

      double futureValue =
FinancialForecast.forecastValue(initialValue,
monthlyGrowthRate, forecastMonths);
      System.out.printf("Forecasted Value after %d months:
₹%.2f%n", forecastMonths, futureValue);
   }
}
```

- **OUTPUT:**



```
Console ×
<terminated> ForecastTest [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (Jun 22, 2025,
Forecasted Value after 6 months: ₹13400.96
```