WEEK:1

EXERCISE:1-Singleton Pattern

Logger.java

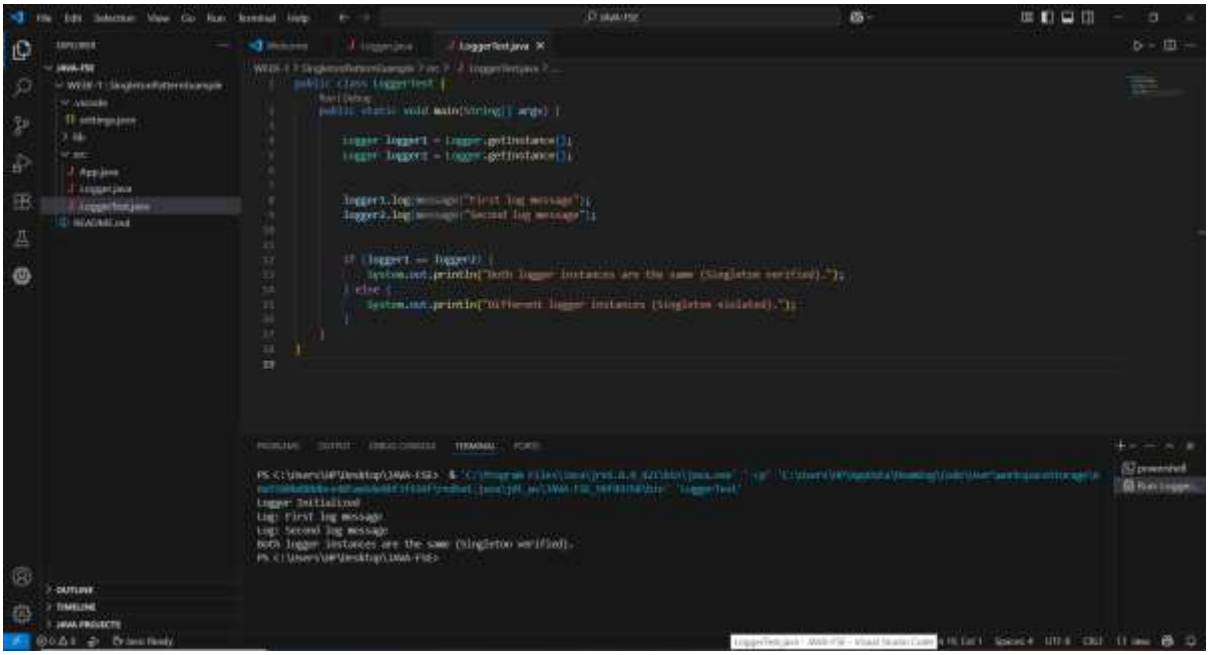
```
public class Logger {
   private static Logger instance;
   private Logger() {
      System.out.println("Logger Initialized");
   }
      public static Logger getInstance() {
      if (instance == null) {
         instance = new Logger();
      }
      return instance;
   }
   public void log(String message) {
      System.out.println("Log: " + message);
   }
}
```

LoggerTest.java

```
public class LoggerTest {
   public static void main(String[] args) {
      Logger logger1 = Logger.getInstance();
      Logger logger2 = Logger.getInstance();
      logger1.log("First log message");
      logger2.log("Second log message");
      if (logger1 == logger2) {
```

```
        System.out.println("Both logger instances are the same (Singleton verified).");
    } else {
        System.out.println("Different logger instances (Singleton violated).");
    }
  }
}
```

OUTPUT:



EXCERSICE:2-FACTORY METHOD PATTERN

Document.java

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

DocumentFactory.java

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

```
}
```

DocumentTest.java

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

        DocumentFactory pdfFactory = new PdfFactory();
        Document pdfDoc = pdfFactory.createDocument();
        pdfDoc.open();
        DocumentFactory excelFactory = new ExcelFactory();
        Document excelDoc = excelFactory.createDocument();
        excelDoc.open();
    }
}
```

ExcelDocument.java

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

ExcelFactory.java

```
public class ExcelFactory extends DocumentFactory {
    public Document createDocument() {
        return new ExcelDocument();
```

```
    }
}
```

PdfDocument.java

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

PdfFactory.java

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

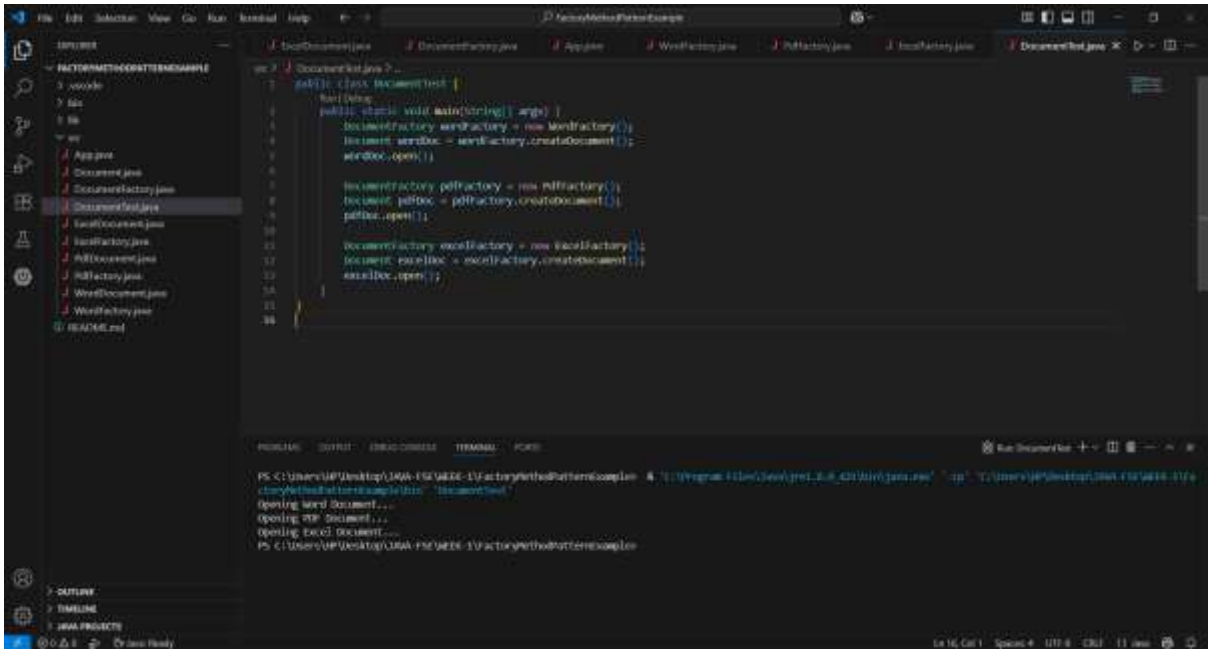
WordDocument.java

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

WordFactory.java

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

OUTPUT:



EXERCISE:3-Ecommerce Platform Search Engine

Product.java

```
package model;

public class Product {

    int productId;

    public 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 "[" + productId + ", " + productName + ", " + category + "]";
    }
}
```

SearchEngine.java

```
package search;
import model.Product;
import java.util.Arrays;
import java.util.Comparator;
public class SearchEngine {
    public static Product linearSearch(Product[] products, String targetName) {
        for (Product p : products) {
            if (p.productName.equalsIgnoreCase(targetName)) {
                return p;
            }
        }
        return null;
    }
    public static Product binarySearch(Product[] products, String targetName) {
        Arrays.sort(products, Comparator.comparing(p ->
p.productName.toLowerCase()));
        int left = 0, right = products.length - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            int cmp =
products[mid].productName.compareToIgnoreCase(targetName);
            if (cmp == 0) return products[mid];
            else if (cmp < 0) left = mid + 1;
```

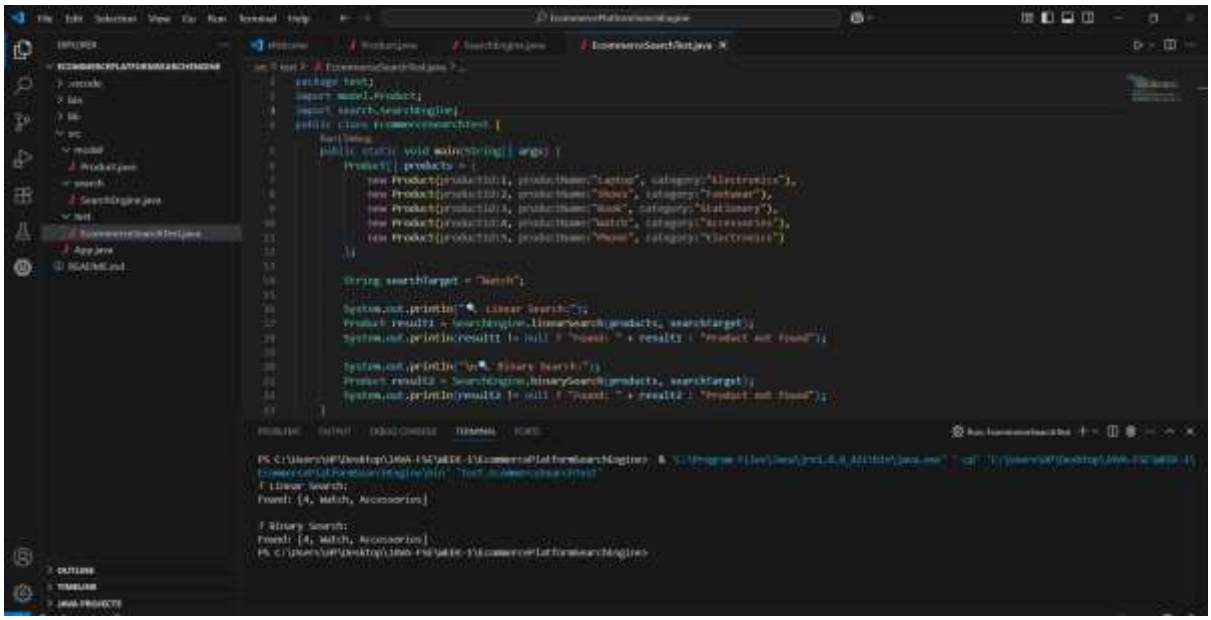
```
            else right = mid - 1;
        }
        return null;
    }
}
```

EcommerceSearchTest.java

```
package test;
import model.Product;
import search.SearchEngine;
public class EcommerceSearchTest {
    public static void main(String[] args) {
        Product[] products = {
            new Product(1, "Laptop", "Electronics"),
            new Product(2, "Shoes", "Footwear"),
            new Product(3, "Book", "Stationery"),
            new Product(4, "Watch", "Accessories"),
            new Product(5, "Phone", "Electronics")
        };
        String searchTarget = "Watch";
        System.out.println(" Linear Search:");
        Product result1 = SearchEngine.linearSearch(products, searchTarget);
        System.out.println(result1 != null ? "Found: " + result1 : "Product not
found");
        System.out.println(" Binary Search:");
        Product result2 = SearchEngine.binarySearch(products, searchTarget);
        System.out.println(result2 != null ? "Found: " + result2 : "Product not
found");
```

```
    }
}
```

OUTPUT:



EXERCISE:4-FINANCIAL FORECASTING

FinancialForecast.java

```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Hello, World!");
    }
}
```

ForecastTest.java

```
package test;
import forecast.FinancialForecast;
public class ForecastTest {
    public static void main(String[] args) {
```

```
        double pv = 10000;
        double rate = 0.07;
        int years = 5;
        System.out.printf("Recursive: Future value after %d years = %.2f\n", years,
            FinancialForecast.forecast(pv, rate, years));
        System.out.printf("Iterative: Future value after %d years = %.2f\n", years,
            FinancialForecast.forecastIterative(pv, rate, years));
    }
}
```

OUTPUT: