WEEK-1

1) **Exercise 1: Implementing the Singleton Pattern**

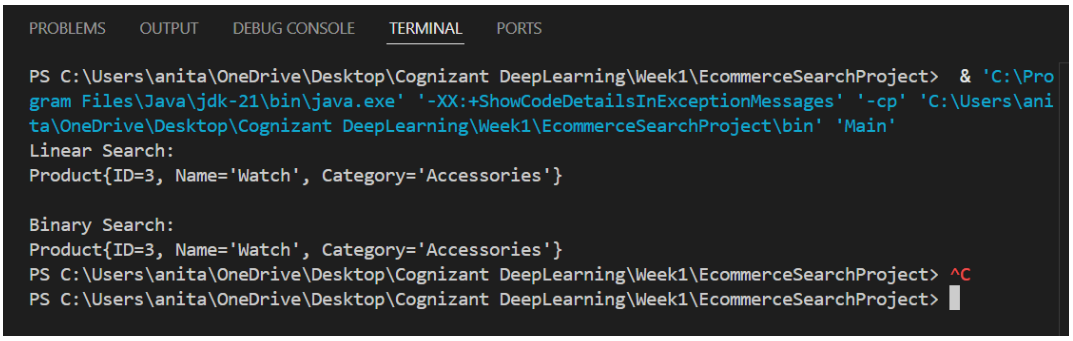
Logger.java

```
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);
   }}
```

Main.java

```
public class Main {
   public static void main(String[] args) {
      Logger logger1 = Logger.getInstance();
      logger1.log("First log message");
      Logger logger2 = Logger.getInstance();
      logger2.log("Second log message");
      if (logger1 == logger2) {
         System.out.println("Both logger instances are the same (Singleton works!)");
      } else {
         System.out.println("Different logger instances (Singleton failed)");
      }}}
```

OUTPUT:



```
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject>  & 'C:\Pro
gram Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ani
ta\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject\bin' 'Main'
Linear Search:
Product{ID=3, Name='Watch', Category='Accessories'}

Binary Search:
Product{ID=3, Name='Watch', Category='Accessories'}
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject> ^C
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject>
```

2) **Exercise 2: Implementing the Factory Method Pattern**

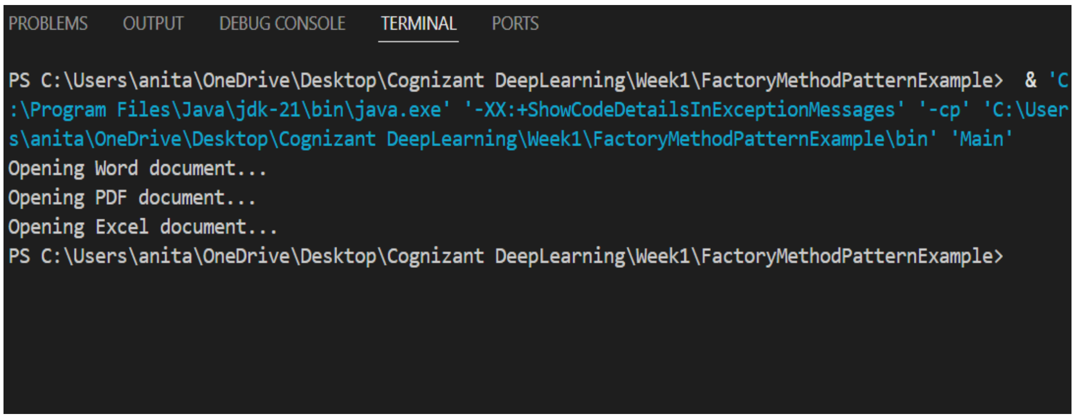
Document.java
```
public interface Document {
   void open();
}
```
DocumentFactory.java
```
public abstract class DocumentFactory {
   public abstract Document createDocument();
}
```
ExcelDocument.java
```
public class ExcelDocument implements Document {
   @Override
   public void open() {
      System.out.println("Opening Excel document...");
   }
}
```
ExcelDocumentFactory.java
```
public class ExcelDocumentFactory extends DocumentFactory {
   @Override
   public Document createDocument() {
      return new ExcelDocument();
   }}
```
PdfDocument.java
```
public class PdfDocument implements Document {
   @Override
   public void open() {
      System.out.println("Opening PDF document...");
   }}
```
PdfDocumentFactory.java
```
public class PdfDocumentFactory extends DocumentFactory {
   @Override
   public Document createDocument() {
      return new PdfDocument();
   }}
```
WordDocument.java
```
public class WordDocument implements Document {
   @Override
   public void open() {
      System.out.println("Opening Word document...");
   }}
```

WordDocumentFactory.java

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

Main.java

```
public class Main {
  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();
  }}
```



```
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\FactoryMethodPatternExample>  & 'C
:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\User
s\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\FactoryMethodPatternExample\bin' 'Main'
Opening Word document...
Opening PDF document...
Opening Excel document...
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\FactoryMethodPatternExample>
```

3) Exercise 2: E-commerce Platform Search Function

Main.java

```
public class Main {

  public static void main(String[] args) {

    Product[] products = {

      new Product(1, "Laptop", "Electronics"),

      new Product(2, "Shoes", "Footwear"),

      new Product(3, "Watch", "Accessories"),
      new Product(4, "Phone", "Electronics")
    };

    System.out.println("Linear Search:");
    Product found1 = Search.linearSearch(products, "Watch");
    System.out.println(found1 != null ? found1 : "Product not found");

    System.out.println("\nBinary Search:");
    Product found2 = Search.binarySearch(products, "Watch");
    System.out.println(found2 != null ? found2 : "Product not found");
  }}
```

Product.java

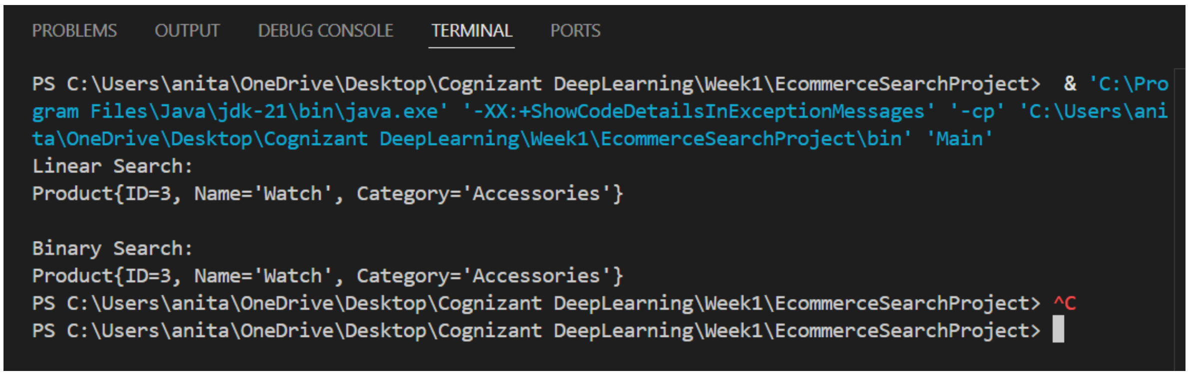
```
public class Product {
    private int productId;
    private String productName;
    private String category;
    public Product(int productId, String productName, String category) {
        this.productId = productId;
        this.productName = productName;
        this.category = category;
    }
    public int getProductId() {
        return productId;
    }
    public String getProductName() {
        return productName;
    }
    public String getCategory() {
        return category;
```

```
  }
  @Override
  public String toString() {
    return "Product{" +
        "ID=" + productId +
        ", Name='" + productName + '\'' +
        ", Category='" + category + '\'' +
        '}'; }}
```

Search.java

```
import java.util.Arrays;
import java.util.Comparator;
public class Search {
  public static Product linearSearch(Product[] products, String name) {
    for (Product p : products) {
      if (p.getProductName().equalsIgnoreCase(name)) {
        return p;
      }}
    return null;
  }
  public static Product binarySearch(Product[] products, String name) {
    Arrays.sort(products, Comparator.comparing(Product::getProductName));
    int left = 0;
    int right = products.length - 1;
     while (left <= right) {
      int mid = (left + right) / 2;
      int result = name.compareToIgnoreCase(products[mid].getProductName());
      if (result == 0) return products[mid];
      else if (result < 0) right = mid - 1;
      else left = mid + 1;
```

```
        }
        return null;
    }}
```

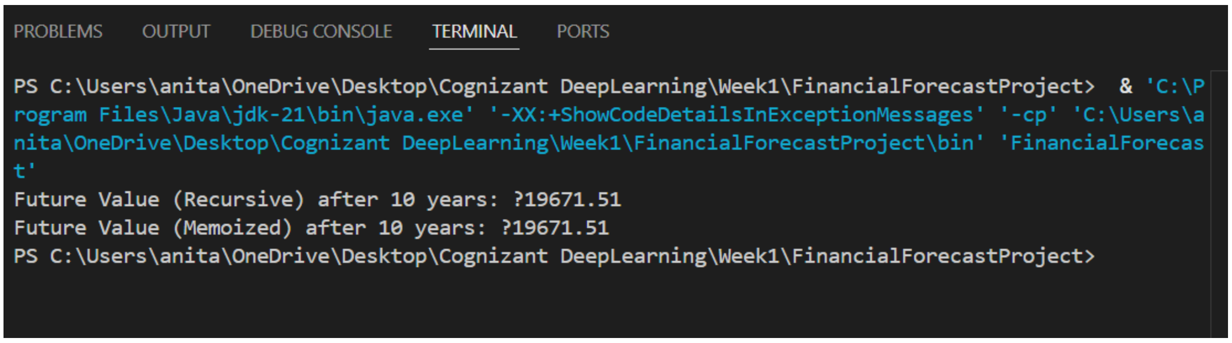


```
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject>  & 'C:\Pro
gram Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ani
ta\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject\bin' 'Main'
Linear Search:
Product{ID=3, Name='Watch', Category='Accessories'}

Binary Search:
Product{ID=3, Name='Watch', Category='Accessories'}
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject> ^C
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\EcommerceSearchProject>
```

4) Exercise 7: Financial Forecasting

Financial Forecasting.java

```
public class FinancialForecast {


    // Recursive method to calculate future value
    public static double futureValueRecursive(double principal, double rate, int years) {
        if (years == 0) {
            return principal;
        }
        return futureValueRecursive(principal, rate, years - 1) * (1 + rate);
    }


    // Memoized version to optimize
    public static double futureValueMemo(double principal, double rate, int years,
double[] memo) {
        if (years == 0) return principal;
        if (memo[years] != 0) return memo[years];
        memo[years] = futureValueMemo(principal, rate, years - 1, memo) * (1 + rate);
        return memo[years];
    }

    public static void main(String[] args) {
        double principal = 10000; // Initial amount
        double rate = 0.07;       // Annual growth rate (7%)
        int years = 10;           // Number of years to forecast

        // Using simple recursion
        double result = futureValueRecursive(principal, rate, years);
        System.out.printf("Future Value (Recursive) after %d years: ₹%.2f\n", years,
result);

        // Using memoization
        double[] memo = new double[years + 1];
        double optimizedResult = futureValueMemo(principal, rate, years, memo);
        System.out.printf("Future Value (Memoized) after %d years: ₹%.2f\n", years,
optimizedResult);
    }
}
```



```
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\FinancialForecastProject>  & 'C:\P
rogram Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\a
nita\OneDrive\Desktop\Cognizant DeepLearning\Week1\FinancialForecastProject\bin' 'FinancialForecas
t'
Future Value (Recursive) after 10 years: ?19671.51
Future Value (Memoized) after 10 years: ?19671.51
PS C:\Users\anita\OneDrive\Desktop\Cognizant DeepLearning\Week1\FinancialForecastProject>
```