Design principles & Patterns

Exercise 1: Implementing the Singleton Pattern

Code:

App.java

```
package singletonPatternExample;


public class App {

    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 are same instance");
        }
        else{
            System.out.println("Both are different instance");
        }
    }
}
```

Logger.java

```
package singletonPatternExample;


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(message);
   }
}
```

Output

```
ns-and-Principles> javac SingletonPatternExample\App.java SingletonPatternExample\Logger.java
PS E:\Cognizant-Java FSE\Week 1\code\Module-1-Design-Patterns-and-Principles> java SingletonPatternExample.App
Logger Initialized
First log message
Second log message
Both are same instance
PS E:\Cognizant-Javjavac SingletonPatternExample/*.java
```

Exercise 2: Implementing the Factory Method Pattern

Code:

Document.java

```java
package factoryMethodPatternExample;

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

ExcelDocument.java

```java
package factoryMethodPatternExample;

public class ExcelDocument implements Document {
    @Override
    public void createDocument() {
        System.out.println("Creating an Excel Document");
    }

    @Override
    public void open() {
        System.out.println("Opening an Excel Document");
    }
}
```

ExcelDocumentFactory.java

```java
package factoryMethodPatternExample;

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

PdfDocument.java

```java
package factoryMethodPatternExample;

public class PdfDocument implements Document {
    @Override
    public void createDocument() {
        System.out.println("Creating a PDF Document");
    }

    @Override
    public void open() {
        System.out.println("Opening a PDF Document");
    }
}
```

PdfDocumentFactory.java

```java
package factoryMethodPatternExample;

public interface PdfDocumentFactory {
public void createDocument();

}
```

WordDocument.java

```java
package factoryMethodPatternExample;

public class WordDocument implements Document {
    @Override
    public void createDocument() {
        System.out.println("Creating a Word Document");
    }

    @Override
    public void open() {
        System.out.println("Opening a Word Document");
    }
}
```

WordDocumentFactory.java

```java
package factoryMethodPatternExample;

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

Test.java

```java
package factoryMethodPatternExample;

public class Test {
    public static void main(String args[]){
        ExcelDocument excel=new ExcelDocument();
        excel.createDocument();
        PdfDocument pdf=new PdfDocument();
        pdf.createDocument();
        WordDocument word=new WordDocument();
        word.createDocument();
    }
}
```

Output:

```
PS E:\Cognizant-Java FSE\Week 1\Code\Module-1-Design-Patterns-and-Principles> java factoryMethodPatternExample.Test
Creating an Excel Document
Opening an Excel Document
Creating a PDF Document
Opening a PDF Document
Creating a Word Document
Opening a Word Document
PS E:\Cognizant-Java FSE\Week 1\Code\Module-1-Design-Patterns-and-Principles>
```