COGNIZANT DIGITAL NURTURE MANDATORY PROGRAM
DESIGN PATTERNS AND PRINCIPLES

Exercise 1: Implementing the Singleton Pattern
Scenario:
You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

CODE :

```
class Logger{
    private static Logger instance;
    private Logger(){
        System.out.println("Instance is created");
    }
    public static Logger getInstance(){
        if(instance == null){
            instance = new Logger();
        }
        return instance;
    }
}

class LoggerSafe{
    private LoggerSafe(){
        System.out.println("Instance Created");
    }

    private static class safeInstanceCreator{
        public static final LoggerSafe INSTANCE = new
LoggerSafe();
    }

    public static LoggerSafe getInstance(){
        return safeInstanceCreator.INSTANCE;
    }
}

public class Test
{
public static void main(String[] args) {
     System.out.println("Hello World");
     //unsafe version for multithread operation
     Logger obj1 = Logger.getInstance();
     Logger obj2 = Logger.getInstance();

     if(obj2 == obj1)
     System.out.println("Same instance");
     else
     System.out.println("Different instances");

        // saffer version
        LoggerSafe obj3 = LoggerSafe.getInstance();
        LoggerSafe obj4 = LoggerSafe.getInstance();
```

```
        if(obj2 == obj1)
      System.out.println("Same safe instance");
      else
      System.out.println("Different safe instances");
}
}
```

OUTPUT:

Exercise 2: Implementing the Factory Method Pattern
Scenario:
You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

CODE :

```
public class Main{
    public static void main(String[] args){
        System.out.println("Hello world");

        WordDocumentFactory wdf = new
WordDocumentFactory();
        Document wordDoc = wdf.createDocument();
        wordDoc.open();

        PDFDocumentFactory wdf1 = new PDFDocumentFactory();
        Document PDFDoc = wdf1.createDocument();
        PDFDoc.open();

        ExcelDocumentFactory wdf2 = new
ExcelDocumentFactory();
        Document excelDoc = wdf2.createDocument();
        excelDoc.open();
    }
}

interface Document{
    public void open();
}

class WordDocument implements Document{
```

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

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

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

OUTPUT: