Design Patterns Hands-on Report

# Exercise 1: Singleton Pattern – Logger Utility

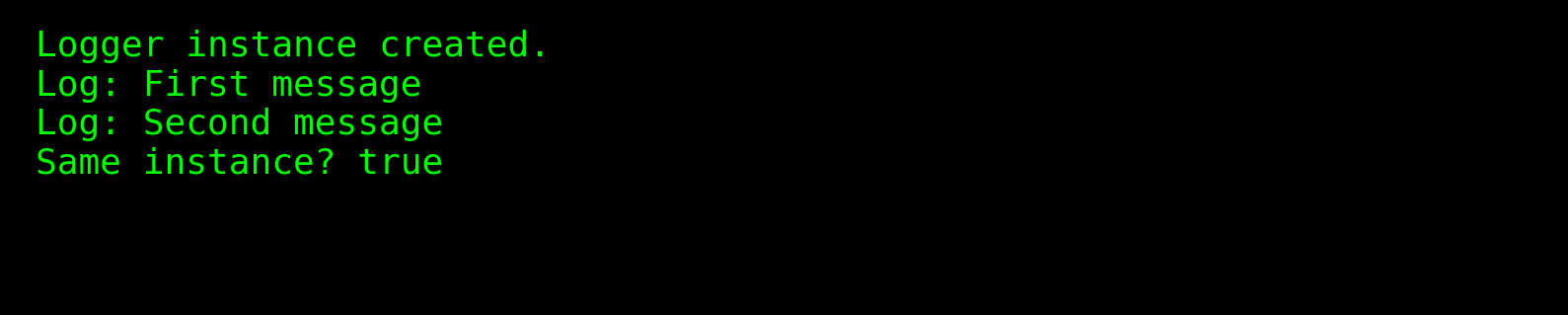
This implementation ensures that the Logger class has only one instance used throughout the application, maintaining consistent logging.

## Logger.java

public class Logger {  
 private static Logger instance;  
  
 private Logger() {  
 System.out.println("Logger instance created.");  
 }  
  
 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 message");  
  
 Logger logger2 = Logger.getInstance();  
 logger2.log("Second message");  
  
 System.out.println("Same instance? " + (logger1 == logger2));  
 }  
}

→ Screenshot of output :

# Exercise 2: Factory Method Pattern – Document Creation System

This pattern is used to instantiate different types of documents (Word, PDF, Excel) through a factory method, promoting flexibility and loose coupling.

## Document.java

public interface Document {  
 void open();  
}

## WordDocument.java

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

## PdfDocument.java

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

## ExcelDocument.java

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

## DocumentFactory.java

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

## WordFactory.java

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

## PdfFactory.java

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

## ExcelFactory.java

public class ExcelFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}

## Main.java

public class Main {  
 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();  
 }  
}

→ Screenshot of output :  
