**Exercise 1: Implementing the Singleton Pattern**

**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);

}

}

**SingletonTest.java:**

public class SingletonTest {

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 logger1 and logger2 are the same instance.");

} else {

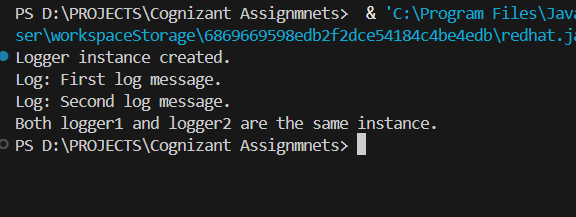
System.out.println("Different instances! Singleton failed.");

}

}

}

**Output:**



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

Code:

Create a Document interface which all document types will implement.

// Document.java

public interface Document {

void open();

}

**Create Concrete Document Classes**

These classes **implement the Document interface**.

// 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.");

}

}

**Implement the Factory Method**

First, create an **abstract factory class**:

// DocumentFactory.java

public abstract class DocumentFactory {

public abstract Document createDocument();

}

Then, create a **concrete factory** for each type of document:

// WordDocumentFactory.java

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

// PdfDocumentFactory.java

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

// ExcelDocumentFactory.java

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

**Test the Factory Method Implementation**

This test class shows how to use the **factory method pattern** to create different types of documents **without changing the client code**.

// FactoryMethodTest.java

public class FactoryMethodTest {

public static void main(String[] args) {

// Create Word document using factory

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

// Create PDF document using factory

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

// Create Excel document using factory

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

Output:

