**Exercise 1: Singleton Pattern**

**Logger.java**

package com.singletonpatternexample;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class Logger {

private static volatile Logger instance;

private final DateTimeFormatter dtFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

private Logger() {}

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

public void info(String message) {

log("INFO", message);

}

public void warn(String message) {

log("WARN", message);

}

public void error(String message) {

log("ERROR", message);

}

private void log(String level, String message) {

String timestamp = LocalDateTime.now().format(dtFormatter);

System.out.printf("%s [%s] - %s%n", timestamp, level, message);

}

}

**Main.java**

package com.singletonpatternexample;

public class Main {

public static void main(String[] args) {

Logger loggerA = Logger.getInstance();

Logger loggerB = Logger.getInstance();

System.out.println("Logger A hashCode: " + loggerA.hashCode());

System.out.println("Logger B hashCode: " + loggerB.hashCode());

System.out.println("Same instance? " + (loggerA == loggerB));

loggerA.info("Application started");

loggerB.warn("This warning still comes from the same logger");

loggerA.error("An error occurred!");

Runnable task = () -> {

Logger threadLogger = Logger.getInstance();

threadLogger.info("Logging from thread " + Thread.currentThread().getName());

};

Thread t1 = new Thread(task, "Worker-1");

Thread t2 = new Thread(task, "Worker-2");

t1.start();

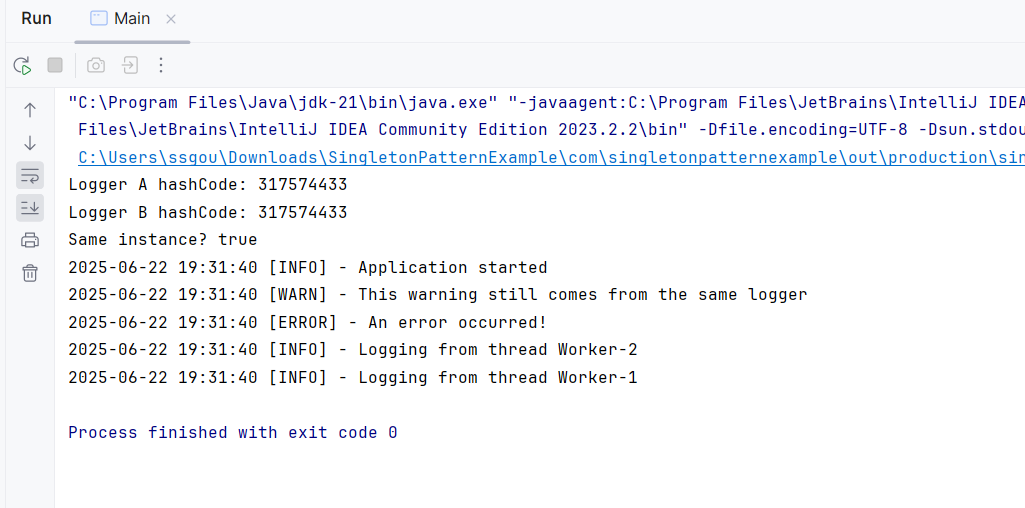
t2.start();

}

}

**Directory Structure**

****

**Output**

**Exercise 2: Factory Method**

**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();  
 }  
}

**WordDocumentFactory.java**

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

**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...");  
 }  
}

**PdfDocumentFactory.java**

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

**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();  
 }  
}

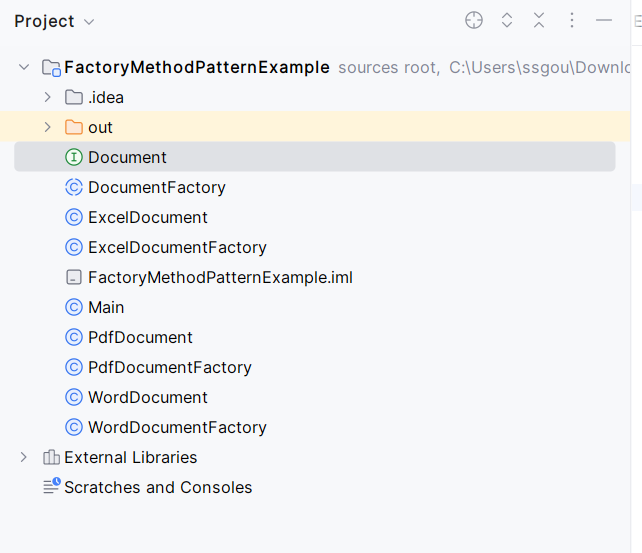
**Documentfactory.java**

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

**Document.java**

public interface Document {  
 void open();  
}

**Directory Structure:**



**Output:**