**Exercise 1: Implementing the Singleton Pattern**

**PROGRAM:**

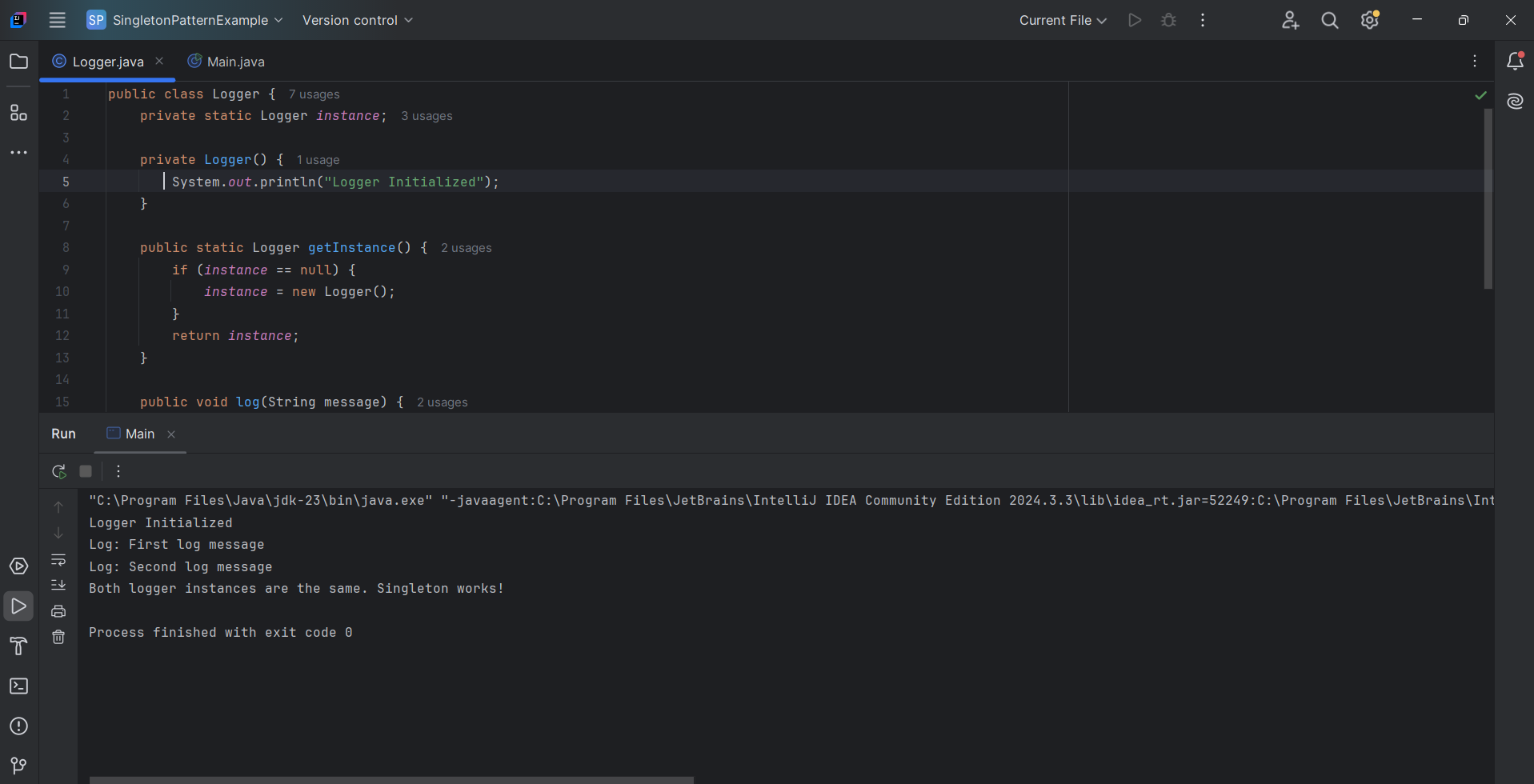
**Logger.java**

public 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("Log: " + message);  
 }  
}

**Main.java:**

public class Main {  
 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 logger instances are the same. Singleton works!");  
 } else {  
 System.*out*.println("Different instances. Singleton failed!");  
 }  
 }  
}

**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.

Steps:

1. Create a New Java Project:
   * Create a new Java project named FactoryMethodPatternExample.
2. Define Document Classes:
   * Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.
3. Create Concrete Document Classes:
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. Implement the Factory Method:
   * Create an abstract class DocumentFactory with a method createDocument().
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the createDocument() method.
5. Test the Factory Method Implementation:
   * Create a test class to demonstrate the creation of different document types using the factory method.

**PROGRAM:**

**1.Document.java**

package com.example.factory;  
  
public interface Document {  
 void open();  
}

**2.** **WordDocument.java**

package com.example.factory;  
public class WordDocument implements Document {  
 @Override  
 public void open() {  
 System.*out*.println("Opening a Word document...");  
 }  
}

**3.PdfDocument.java**

package com.example.factory;  
public class PdfDocument implements Document {  
 @Override  
 public void open() {  
 System.*out*.println("Opening a PDF document...");  
 }  
}

**4. ExcelDocument.java**

package com.example.factory;  
public class ExcelDocument implements Document {  
 @Override  
 public void open() {  
 System.*out*.println("Opening an Excel document...");  
 }  
}

**5. DocumentFactory.java**

package com.example.factory;  
public abstract class DocumentFactory {  
 public abstract Document createDocument();  
}

**6. WordDocumentFactory.java**

package com.example.factory;  
public class WordDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new WordDocument();  
 }  
}

**7.PdfDocumentFactory..java**

package com.example.factory;  
public class PdfDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
}

**8.ExcelDocumentFactory.java**

package com.example.factory;  
public class ExcelDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}

**9.FactoryTest.java**

package com.example.factory;  
  
public class FactoryTest {  
 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();  
 }  
  
}

**output:**

