**SingletonPatternExample**

**Objective:**  
Implement and test the Singleton Design Pattern in Java using a Logger class.

**Steps:**

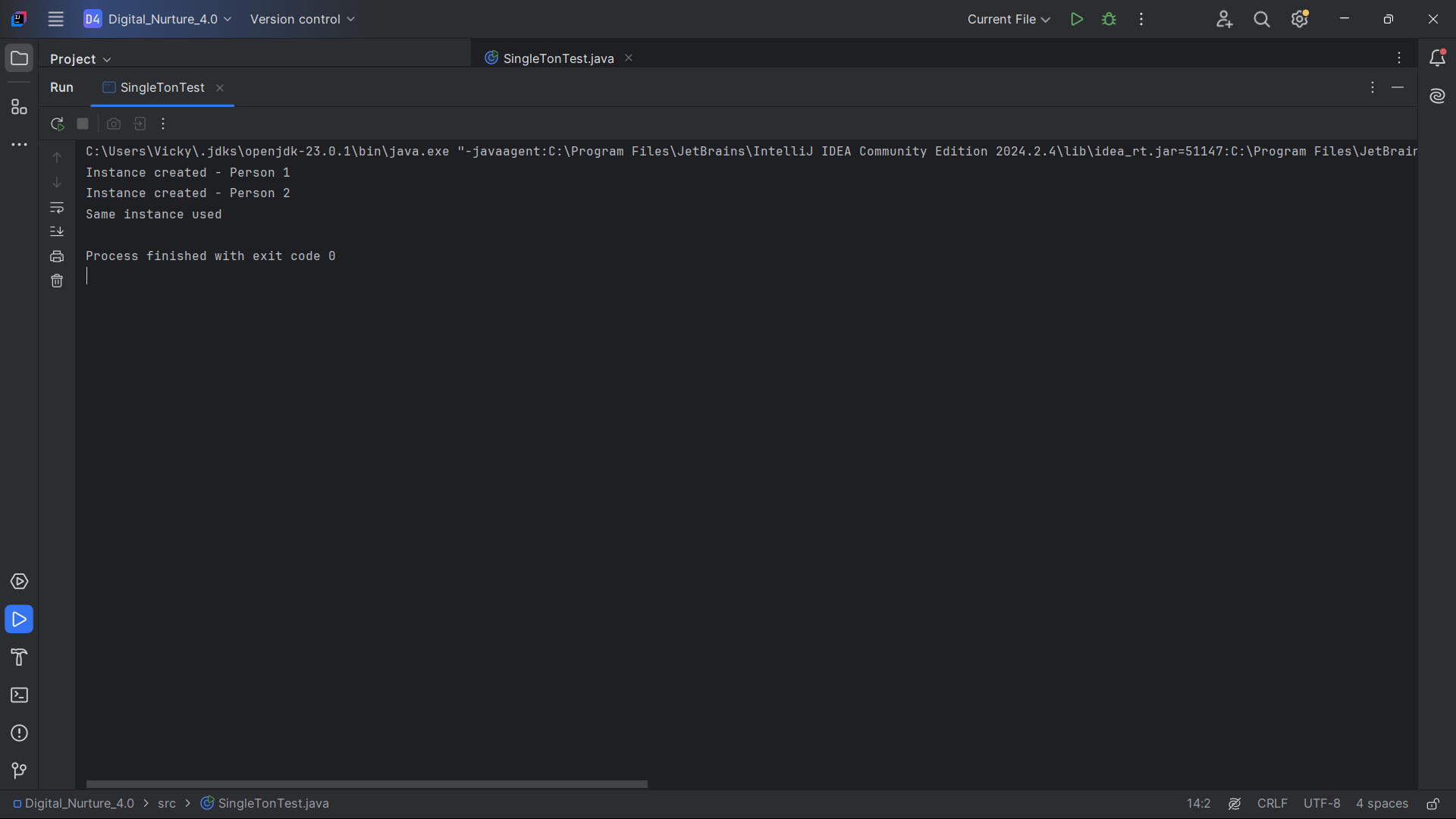
1. **Create a New Java Project:**
   * Name the project **SingletonPatternExample**.
2. **Design the Singleton Class (Logger):**
   * Create a class named Logger.
   * Declare a **private static instance** of the Logger class within itself.
   * Make the **constructor private** to prevent instantiation from outside the class.
   * Implement a **public static method** that returns the single instance of the class.
3. **Implement the Singleton Pattern Logic:**
   * Write the necessary code to ensure the Logger class adheres strictly to the Singleton design principles.
4. **Test the Singleton Implementation:**
   * Create a separate test class.
   * Instantiate the Logger class multiple times through the access method and confirm that all references point to the **same instance**, validating the Singleton behavior.

Source code :

class SingleTon {

private static SingleTon *instance*;  
 private SingleTon() {  
 }  
 public static SingleTon getInstance() {  
 if (*instance* == null) {  
 *instance* = new SingleTon();  
 }  
 return *instance*;  
 }  
 public void statement(String message) {  
 System.*out*.println("Instance created - " + message);  
 }  
}  
public class SingleTonTest {  
 public static void main(String[] args) {  
 SingleTon log1 = SingleTon.*getInstance*();  
 log1.statement("Person 1");  
  
 SingleTon log2 = SingleTon.*getInstance*();  
 log2.statement("Person 2");  
  
 if (log1 == log2) {  
 System.*out*.println("Same instance used");  
 } else {  
 System.*out*.println("Different instances created");  
 }  
 }  
}

Output :



**FactoryMethodPatternDemo**

**Objective:**  
Demonstrate the Factory Method Design Pattern by creating and opening different types of documents using factory classes.

**Steps to Implement:**

**1. Create a New Java Project**

* Create a new Java project and name it **FactoryMethodPatternDemo**.

**2. Define the Document Interface**

* Create an interface named **Doc**.
* Declare a method openDoc() to represent opening a document.

**3. Implement Concrete Document Classes**

* Create the following classes that implement the Doc interface:
  + WordDoc – Implements openDoc() to simulate opening a Word document.
  + PdfDoc – Implements openDoc() to simulate opening a PDF document.
  + ExcelDoc – Implements openDoc() to simulate opening an Excel document.

**4. Implement the Factory Method Pattern**

* **Abstract Factory Class**:
  + Create an abstract class named **DocFactory**.
  + Declare an abstract method createDoc() that returns a Doc type.
* **Concrete Factory Classes**:
  + Create classes WordDocFactory, PdfDocFactory, and ExcelDocFactory that extend DocFactory.
  + Each class overrides the createDoc() method to return the respective document object.

**5. Test the Factory Method Implementation**

* Create a test class named **FactoryMethodPatternDemo**.
* In the main() method:
  + Instantiate the concrete factories.
  + Use them to create different document types via the createDoc() method.
  + Call the openDoc() method on each document to simulate opening it.

Source code :

interface Document {  
 void open();  
}  
class WordDocument implements Document {  
 public void open() {  
 System.*out*.println("Opening a Word document.");  
 }  
}  
  
class PdfDocument implements Document {  
 public void open() {  
 System.*out*.println("Opening a PDF document.");  
 }  
}  
class ExcelDocument implements Document {  
 public void open() {  
 System.*out*.println("Opening an Excel document.");  
 }  
}  
  
abstract class DocumentFactory {  
 public abstract Document createDocument();  
}  
  
class WordDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new WordDocument();  
 }  
}  
  
class PdfDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
}  
  
class ExcelDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}  
public class FactoryMethodPatternDemo {  
 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 :

