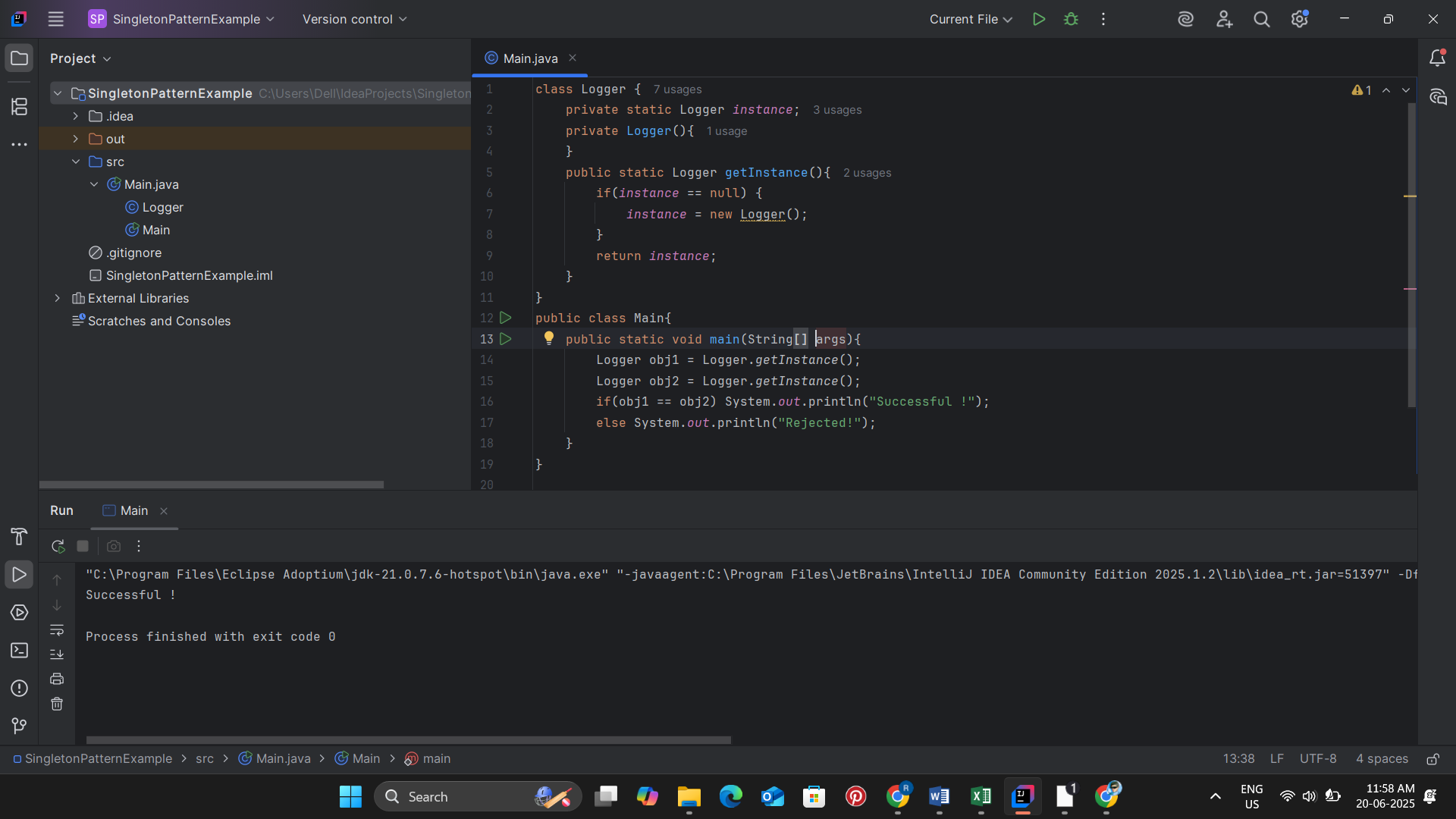
**Exercise 1: Implementing the Singleton Pattern**

**Code:**

class Logger {  
 private static Logger *instance*;  
 private Logger(){  
 }  
 public static Logger getInstance(){  
 if(*instance* == null) {  
 *instance* = new Logger();  
 }  
 return *instance*;  
 }  
}  
public class Main{  
 public static void main(String[] args){  
 Logger obj1 = Logger.*getInstance*();  
 Logger obj2 = Logger.*getInstance*();  
 if(obj1 == obj2) System.*out*.println("Successful !");  
 else System.*out*.println("Rejected!");  
 }  
}

**Output**

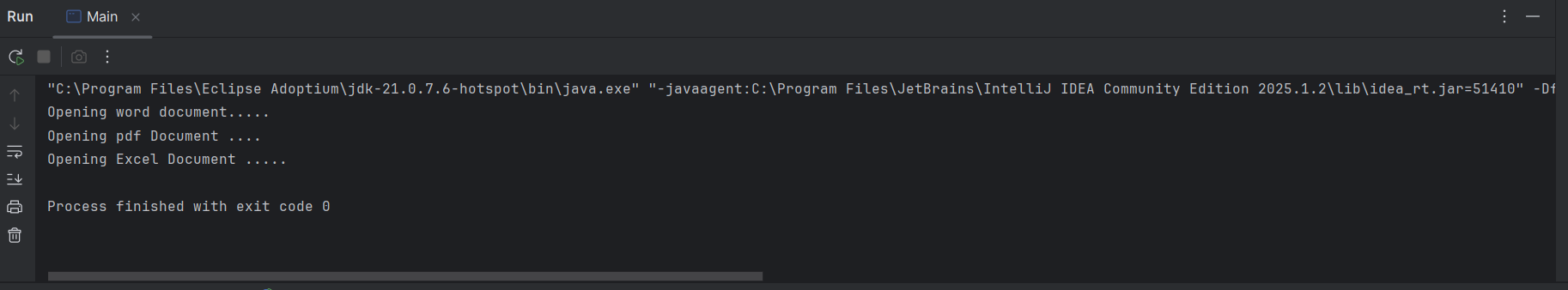


**Exercise 2: Implementing the Factory Method Pattern**

**Code:**

interface Document {  
 void open();  
}  
  
class WordDocument implements Document {  
 public void open(){  
   
 System.*out*.println("Opening word document.....");  
 }  
}  
class PDFDocument implements Document{  
 public void open(){  
 System.*out*.println("Opening pdf Document ....");  
 }  
}  
  
class ExcelDocument implements Document{  
 public void open(){  
 System.*out*.println("Opening 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 Main {  
 public static void main(String[] args){  
 DocumentFactory wordF = new WordDocumentFactory();  
 Document word = wordF.createDocument();  
 word.open();  
  
 DocumentFactory pdfF = new PDFDocumentFactory();  
 Document pdf = pdfF.createDocument();  
 pdf.open();  
  
 DocumentFactory excelF = new ExcelDocumentFactory();  
 Document excel = excelF.createDocument();  
 excel.open();  
 }  
}

**Output:**

****