Design Patterns and Principles

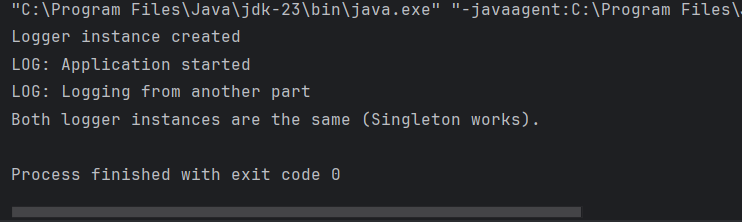
Exercise 1: Implementing the Singleton Pattern:

1. Code:

public class Logger {  
 // Step 1: Private static instance of Logger  
 private static Logger *instance*;  
  
 // Step 2: Private constructor  
 private Logger() {  
 System.*out*.println("Logger instance created");  
 }  
  
 // Step 3: Public static method to get the instance  
 public static Logger getInstance() {  
 if (*instance* == null) {  
 *instance* = new Logger(); // Create instance if it doesn't exist  
 }  
 return *instance*;  
 }  
  
 // Sample method to demonstrate functionality  
 public void log(String message) {  
 System.*out*.println("LOG: " + message);  
 }  
}

public class Main {  
 public static void main(String[] args) {  
  
 Logger logger1 = Logger.*getInstance*();  
 Logger logger2 = Logger.*getInstance*();  
  
  
 logger1.log("Application started");  
 logger2.log("Logging from another part");  
  
  
 if (logger1 == logger2) {  
 System.*out*.println("Both logger instances are the same (Singleton works).");  
 } else {  
 System.*out*.println("Different logger instances (Singleton failed).");  
 }  
 }  
}

2. Output:



Exercise 2: Implementing the Factory Method:

1. Code:

public interface Document {  
 void open();  
}

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

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

public class ExcelDocumentFactory extends DocumentFactory{  
 @Override  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}

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

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

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

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

2.Output:

