

HANDS-ON-EXERCISE
MODULE I- Designs patterns and principles

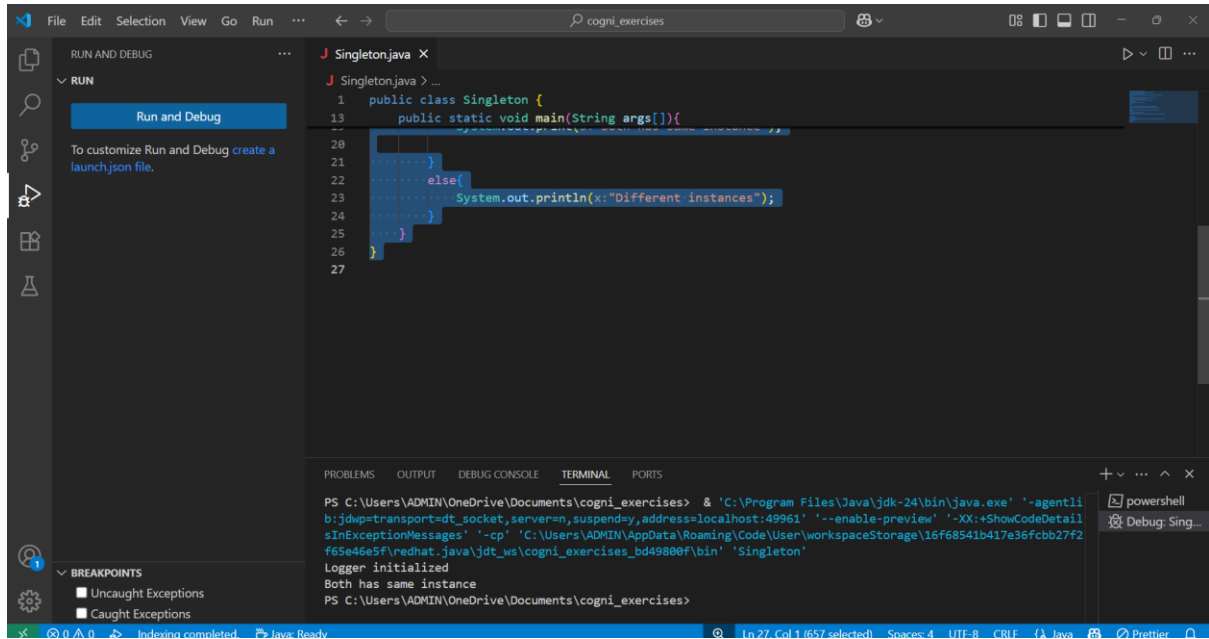
EX.NO.01

SINGLETON_PATTERN

CODE:

```
public class Singleton {  
    private static Singleton instance;  
    private Singleton(){  
        System.out.println("Logger initialized");  
    }  
    public static Singleton getInstance(){  
        if(instance==null){  
            instance=new Singleton();  
        }  
        return instance;  
    }  
  
    public static void main(String args[]){  
        Singleton log1=Singleton.getInstance();  
  
        Singleton log2=Singleton.getInstance();  
  
        if(log1==log2){  
            System.out.print("Both has same instance");  
  
        }  
        else{  
            System.out.println("Different instances");  
        }  
    }  
}
```

OUTPUT:



The screenshot shows an IDE window with a Java file named `Singleton.java`. The code defines a Singleton class with a static `main` method. The `main` method calls `Singleton.getInstance()` and prints the result. The IDE's `Run and Debug` panel is open, showing the `Run` button. The `Terminal` panel at the bottom shows the command prompt output:

```
PS C:\Users\ADMIN\OneDrive\Documents\cogni_exercises> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:49961' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ADMIN\AppData\Roaming\Code\User\workspaceStorage\16f68541b417e36fcb27f2f65e46e5f\redhat.java\jdt_ws\cogni_exercises_bd49800f\bin' 'Singleton'
Logger initialized
Both has same instance
PS C:\Users\ADMIN\OneDrive\Documents\cogni_exercises>
```

EX.NO.02

FACTORY_METHOD_PATTERN

CODE:

```
public class FactoryMethodPattern {  
  
    interface Document {  
  
        void open();  
    }  
  
    static class WordDoc implements Document {  
  
        public void open() {  
            System.out.println("Word_Documet");  
        }  
    }  
  
    static class PdfDoc implements Document {  
  
        public void open() {  
            System.out.println("PDF_Documet");  
        }  
    }  
  
    static class ExlDoc implements Document {  
  
        public void open() {  
            System.out.println("Excel_Documet");  
        }  
    }  
}
```

```

    }
}

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

static class WordFactory extends DocumentFactory{
    public Document createDocument(){
        return new WordDoc();
    }
}

static class PdfFactory extends DocumentFactory{
    public Document createDocument(){
        return new PdfDoc();
    }
}

static class ExcelFactory extends DocumentFactory{
    public Document createDocument(){
        return new ExlDoc();
    }
}

public static void main(String[] args) {
    DocumentFactory word=new WordFactory();
    Document doc1=word.createDocument();
    doc1.open();

    DocumentFactory pdf=new PdfFactory();
    Document doc2=pdf.createDocument();
}

```

```
doc2.open();
```

```
DocumentFactory exl=new ExcelFactory();
```

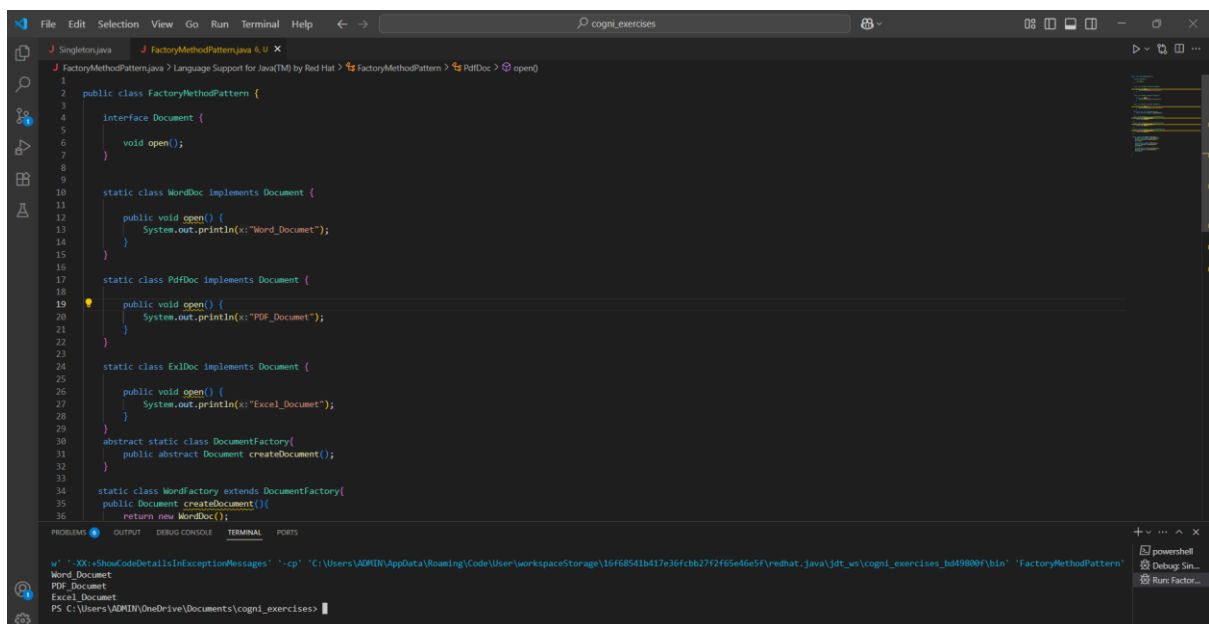
```
Document doc3=exl.createDocument();
```

```
doc3.open();
```

```
}
```

```
}
```

OUTPUT:



The screenshot shows an IDE window titled 'cogni_exercises' with a Java file 'FactoryMethodPattern.java'. The code defines an interface 'Document' with an 'open()' method. It then implements this interface with three static classes: 'WordDoc', 'PdfDoc', and 'ExcelDoc'. Each class has an 'open()' method that prints its name. An abstract class 'DocumentFactory' defines a 'createDocument()' method. A concrete class 'WordFactory' extends 'DocumentFactory' and overrides 'createDocument()' to return a new 'WordDoc' object. The terminal output shows the execution of the program, resulting in three lines of output: 'Word_Documet', 'PDF_Documet', and 'Excel_Documet'.

```
1 public class FactoryMethodPattern {
2
3     interface Document {
4
5         void open();
6     }
7
8
9     static class WordDoc implements Document {
10
11         public void open() {
12             System.out.println("Word_Documet");
13         }
14     }
15
16     static class PdfDoc implements Document {
17
18         public void open() {
19             System.out.println("PDF_Documet");
20         }
21     }
22
23     static class ExcelDoc implements Document {
24
25         public void open() {
26             System.out.println("Excel_Documet");
27         }
28     }
29
30     abstract static class DocumentFactory {
31         public abstract Document createDocument();
32     }
33
34     static class WordFactory extends DocumentFactory {
35         public Document createDocument() {
36             return new WordDoc();
37         }
38     }
39 }
```

Word_Documet
PDF_Documet
Excel_Documet