**Exercise 1: Implementing the Singleton Pattern**

CODE

public class Logger {

private static Logger instance;

private Logger(){

System.out.println("Logger initialised");

}

public static Logger get() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void call(String message){

System.out.println("call :"+message);

}

}

public class test {

public static void main(String[] args) {

Logger temp1=Logger.get();

temp1.call("first call done");

Logger temp2=Logger.get();

temp2.call("second call done");

if(temp1==temp2){

System.out.println("Both instances are the same hence SINGLETON works");

}

else{System.out.println("SINGLETON does not works");}

}

}

OUTPUT

A black screen with white text

AI-generated content may be incorrect.

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

CODE

public interface Document {

void open();

}

public class word implements Document {

public void open(){

System.out.println("opening word document");

}

}

public class pdf implements Document {

public void open(){

System.out.println("opening pdf document");

}

}

public class excel implements Document{

public void open(){

System.out.println("opening excel document");

}

}

abstract class Factory {

abstract Document createDocument();

}

public class wordfactory extends Factory {

Document createDocument(){

return new word();

}

}

public class pdffactory extends Factory{

Document createDocument(){

return new pdf();

}

}

public class excelfactory extends Factory {

Document createDocument(){

return new excel();

}

}

public class main{

public static void main(String[] args) {

Factory wordfile=new wordfactory();

Document wordDoc=wordfile.createDocument();

wordDoc.open();

Factory pdffile=new pdffactory();

Document pdfDoc=pdffile.createDocument();

pdfDoc.open();

Factory excelfile=new excelfactory();

Document excelDoc=excelfile.createDocument();

excelDoc.open();

}

}

OUTPUT

