**WEEK – 1 : DESIGN PRINCIPLES AND PATTERN**

**EXERCISE 1 : IMPLEMENTING THE SINGLETON PATTERN**

Logger.java:

package com.example.bankapp;

public class Logger {

private static Logger *instance*;

private Logger() {

System.***out***.println("Logger initialized.");

}

public static Logger getInstance() {

if (*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void log(String message) {

System.***out***.println("[BANK LOG] " + message);

}

}

BankAccount.java:

package com.example.bankapp;

public class BankAccount {

private String accountHolder;

private double balance;

public BankAccount(String name, double initialBalance) {

this.accountHolder = name;

this.balance = initialBalance;

Logger.*getInstance*().log("Account created for " + name + " with balance ₹" + balance);

}

public void deposit(double amount) {

balance += amount;

Logger.*getInstance*().log(accountHolder + " deposited ₹" + amount + ". New balance: ₹" + balance);

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

Logger.*getInstance*().log(accountHolder + " withdrew ₹" + amount + ". Remaining balance: ₹" + balance);

} else {

Logger.*getInstance*().log(accountHolder + " attempted to withdraw ₹" + amount + " but has insufficient funds.");

}

}

}

Main.java:

package com.example.bankapp;

public class Main {

public static void main(String[] args) {

BankAccount account1 = new BankAccount("Stella", 5000);

account1.deposit(1500);

account1.withdraw(2000);

BankAccount account2 = new BankAccount("Alex", 3000);

account2.withdraw(3500);

Logger logger1 = Logger.*getInstance*();

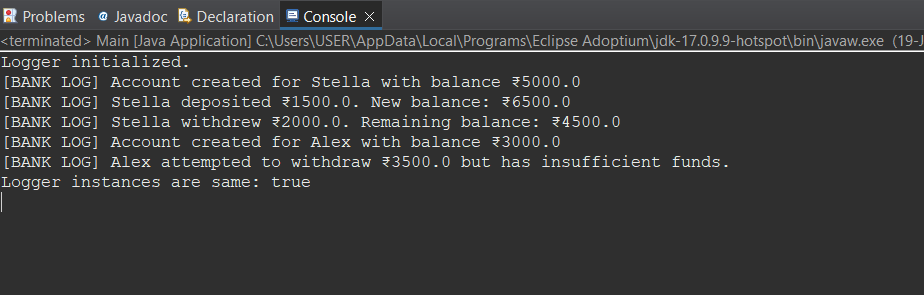
Logger logger2 = Logger.*getInstance*();

System.***out***.println("Logger instances are same: " + (logger1 == logger2));

}

}

**OUTPUT:**



**EXERCISE 2: IMPLEMENTING THE FACTORY METHOD PATTERN**

Document.java:

package com.example.factory;

public interface Document {

void open();

}

WordDocument.java:

package com.example.factory;

public class WordDocument implements Document {

*@Override*

public void open() {

System.***out***.println("Opening a Word document...");

}

}

WordFactory.java:

package com.example.factory;

public class WordFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

return new WordDocument();

}

}

PdfDocument.java:

package com.example.factory;

public class PdfDocument implements Document {

*@Override*

public void open() {

System.***out***.println("Opening a PDF document...");

}

}

PdfFactory.java:

package com.example.factory;

public class PdfFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

return new PdfDocument();

}

}

ExcelDocument.java:

package com.example.factory;

public class ExcelDocument implements Document {

*@Override*

public void open() {

System.***out***.println("Opening an Excel document...");

}

}

ExcelFactory.java:

package com.example.factory;

public class ExcelFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

return new ExcelDocument();

}

}

DocumentFactory.java:

package com.example.factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

Main.java:

package com.example.factory;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.println("Which document would you like to open? (word/pdf/excel): ");

String input = scanner.nextLine().toLowerCase();

DocumentFactory factory;

switch (input) {

case "word":

factory = new WordFactory();

break;

case "pdf":

factory = new PdfFactory();

break;

case "excel":

factory = new ExcelFactory();

break;

default:

System.***out***.println("Invalid input. Please enter 'word', 'pdf', or 'excel'.");

scanner.close();

return;

}

Document doc = factory.createDocument();

doc.open();

scanner.close();

}

}

**OUTPUT:**

