Create Customer class with deposit() and withdraw() as synchronized methods. Declare AccountNo, AccName and Balance as Instance Variables inside the class. From the main class, Input the amount for withdraw() operation and if requested amount is not available in existing Balance amount, withdraw() method should be temporarily suspended using wait() method until deposit() method receives the input for amount, to be added in the existing Balance amount and then withdraw() would be completed in a successful manner. Develop the above scenario using Synchronization and Inter-Thread Communication.

**Note :** existing Bank balance amount 10000

CODE:

import java.util.Scanner;

class Customer {

private int accountNo;

private String accName;

private double balance;

public Customer(int accountNo, String accName, double balance) {

this.accountNo = accountNo;

this.accName = accName;

this.balance = balance;

}

// Synchronized method for deposit

public synchronized void deposit(double amount) {

balance += amount;

System.out.println("Amount deposited: " + amount);

System.out.println("New balance: " + balance);

notify(); // Notify waiting threads that deposit operation is completed

}

// Synchronized method for withdraw

public synchronized void withdraw(double amount) {

if (balance < amount) {

System.out.println("Insufficient balance. Waiting for deposit...");

try {

wait(); // Wait for deposit to increase balance

} catch (InterruptedException e) {

e.printStackTrace();

}

}

balance -= amount;

System.out.println("Withdraw operation success, balance amount " + balance);

}

}

public class withdrawOperations {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter amount to withdraw: ");

double withdrawAmount = scanner.nextDouble();

System.out.print("Enter amount to deposit: ");

double depositAmount = scanner.nextDouble();

scanner.close();

Customer customer = new Customer(12345, "John Doe", 10000);

// Simulating withdrawal request

System.out.println("Requesting to withdraw: " + withdrawAmount);

Thread withdrawThread = new Thread(() -> customer.withdraw(withdrawAmount));

withdrawThread.start();

// Simulating deposit after some time

try {

Thread.sleep(2000); // Simulating delay

} catch (InterruptedException e) {

e.printStackTrace();

}

// Deposit some amount to the account

System.out.println("Depositing: " + depositAmount);

customer.deposit(depositAmount);

}

}

OUTPUT:

C:\javap>javac withdrawOperations.java

C:\javap>java withdrawOperations

Enter amount to withdraw: 12000

Enter amount to deposit: 3000

Requesting to withdraw: 12000.0

Insufficient balance. Waiting for deposit...

Depositing: 3000.0

Amount deposited: 3000.0

New balance: 13000.0

Withdraw operation success, balance amount 1000.0

