**CSE3011 NETWORK PROGRAMMING**

**LAB EXPERIMENT 7**

NAME – B PRATYUSH

REGISTRATION NUMBER – 19BCN7114

LAB SLOT – L1+L2

FACULTY – PROF. MUNEESWARI

**Experiment Description: Multithreading and Synchronization**

**Without Synchronization**

**Code**

**AccountTesting.java**

**package** lab7;

**public** **class** AccountTesting **implements** Runnable

{

**private** Account acct = **new** Account();

**public** **static** **void** main(String[] args) {

AccountTesting r = **new** AccountTesting();

Thread one = **new** Thread(r);

Thread two = **new** Thread(r);

one.setName("User 1");

two.setName("User 2");

one.start();

two.start();

}

**public** **void** run() {

**for** (**int** x = 0; x < 5; x++)

{

makeWithdrawal(10);

**if** (acct.getBalance() < 0)

{

System.***out***.println("account is overdrawn!");

}

}

}

**private** **void** makeWithdrawal(**int** amt)

{

**if** (acct.getBalance() >= amt)

{

System.***out***.println(Thread.*currentThread*().getName() + " is going to withdraw");

**try**

{

Thread.*sleep*(100);

} **catch** (InterruptedException ex)

{

}

acct.withdraw(amt);

System.***out***.println(Thread.*currentThread*().getName() + " completes the withdrawal");

}

**else**

{

System.***out***.println("Not enough in account for " + Thread.*currentThread*().getName() + " to withdraw " + acct.getBalance());

}

}

}

**class** Account

{

**private** **int** balance = 50;

**public** **int** getBalance()

{

**return** balance;

}

**public** **void** withdraw(**int** amount)

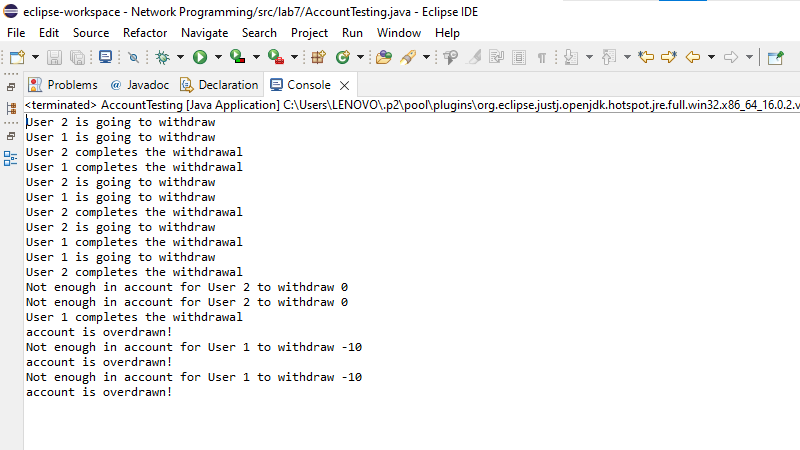
{

balance = balance - amount;

}

}

**Output:**

****

**Here both the users are performing operations parallel which could cause collisions in withdrawal!**

**With Synchronization**

**Code**

**AccountTestingSync.java**

**package** lab7;

**public** **class** AccountTestingSync **implements** Runnable

{

**private** Acct acct = **new** Acct();

**public** **static** **void** main(String[] args) {

AccountTestingSync r = **new** AccountTestingSync();

Thread one = **new** Thread(r);

Thread two = **new** Thread(r);

one.setName("User 1");

two.setName("User 2");

one.start();

two.start();

}

**public** **void** run() {

**for** (**int** x = 0; x < 5; x++)

{

makeWithdrawal(10);

**if** (acct.getBalance() < 0)

{

System.***out***.println("account is overdrawn!");

}

}

}

**private** **void** makeWithdrawal(**int** amt)

{

**synchronized**(**this**) {

**if** (acct.getBalance() >= amt)

{

System.***out***.println(Thread.*currentThread*().getName() + " is going to withdraw");

**try**

{

Thread.*sleep*(100);

} **catch** (InterruptedException ex)

{

}

acct.withdraw(amt);

System.***out***.println(Thread.*currentThread*().getName() + " completes the withdrawal");

}

**else**

{

System.***out***.println("Not enough in account for " + Thread.*currentThread*().getName() + " to withdraw " + acct.getBalance());

}

}

}

}

**class** Acct

{

**private** **int** balance = 50;

**public** **int** getBalance()

{

**return** balance;

}

**public** **void** withdraw(**int** amount)

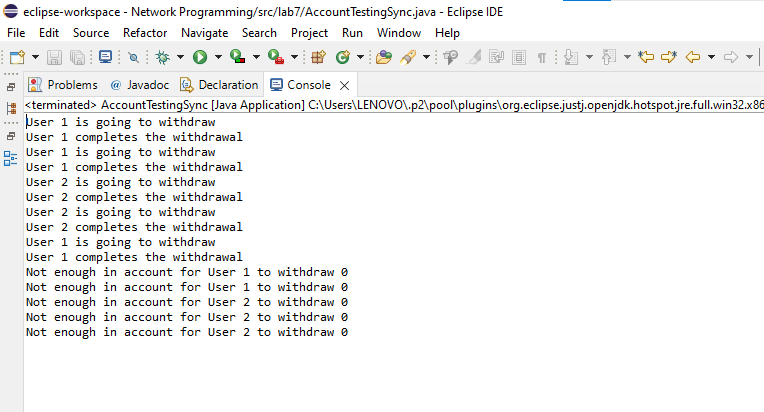
{

balance = balance - amount;

}

}

**Output**

****

**Here one user is obtaining lock, executing the operation and releasing the lock for the next user.**