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

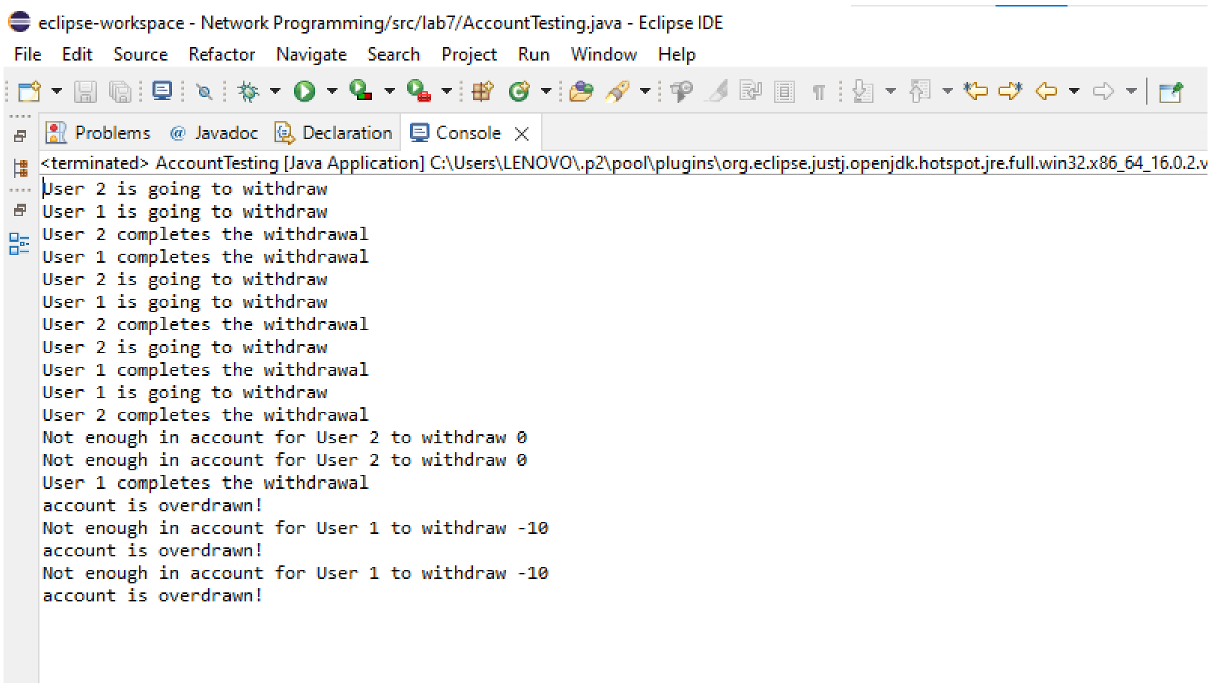
AccountTesting.java

```
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:


eclipse-workspace - Network Programming/src/lab7/AccountTesting.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems
@ Javadoc
Declaration
Console

```
<terminated> AccountTesting [Java Application] C:\Users\LENOVO\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v
User 2 is going to withdraw
User 1 is going to withdraw
User 2 completes the withdrawal
User 1 completes the withdrawal
User 2 is going to withdraw
User 1 is going to withdraw
User 2 completes the withdrawal
User 2 is going to withdraw
User 1 completes the withdrawal
User 1 is going to withdraw
User 2 completes the withdrawal
Not enough in account for User 2 to withdraw 0
Not enough in account for User 2 to withdraw 0
User 1 completes the withdrawal
account is overdrawn!
Not enough in account for User 1 to withdraw -10
account is overdrawn!
Not enough in account for User 1 to withdraw -10
account is overdrawn!
```

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

With Synchronization

Code

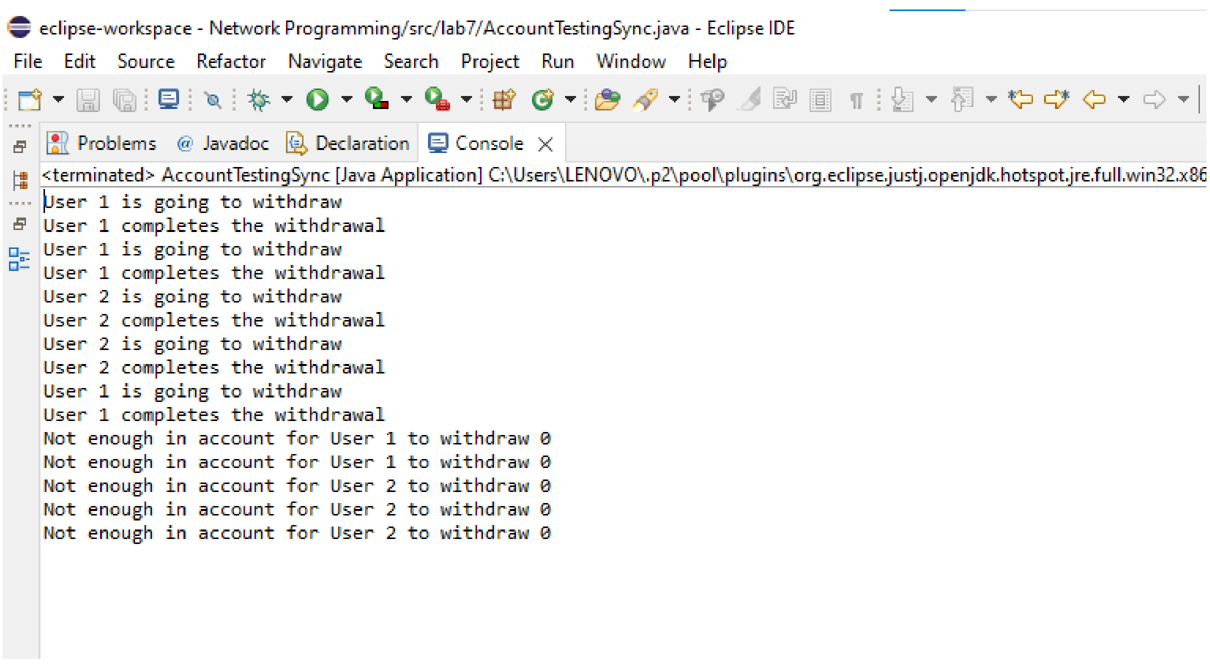
AccountTestingSync.java

```
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



eclipse-workspace - Network Programming/src/lab7/AccountTestingSync.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help
Problems
Javadoc
Declaration
Console

```
<terminated> AccountTestingSync [Java Application] C:\Users\LENOVO\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
User 1 is going to withdraw
User 1 completes the withdrawal
User 1 is going to withdraw
User 1 completes the withdrawal
User 2 is going to withdraw
User 2 completes the withdrawal
User 2 is going to withdraw
User 2 completes the withdrawal
User 1 is going to withdraw
User 1 completes the withdrawal
Not enough in account for User 1 to withdraw 0
Not enough in account for User 1 to withdraw 0
Not enough in account for User 2 to withdraw 0
Not enough in account for User 2 to withdraw 0
Not enough in account for User 2 to withdraw 0
```

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