**LAB # 5**

**Inter-Thread Communication:**

**OBJECTIVE**

Develop an inter-thread user communication program by using synchronization.

**Synchronization in Multi-Threading:**

Synchronization is a process of handling resource accessibility by multiple thread requests. The main purpose of synchronization is to avoid thread interference. At times when more than one thread tries to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called synchronization. The synchronization keyword in java creates a block of code referred to as critical section.

**Why we need Synchronization?**

If we do not use synchronization, and let two or more threads access a shared resource at the same time, it will lead to distorted results.

Consider an example, suppose we have two different threads T1 and T2, T1 starts execution and save certain values in a file temporary.txt which will be used to calculate some result when T1 returns. Meanwhile, T2 starts and before T1 returns, T2 change the values saved by T1 in the file temporary.txt (temporary.txt is the shared resource). Now obviously T1 will return wrong result.

To prevent such problems, synchronization was introduced. With synchronization in above case, once T1 starts using temporary.txt file, this file will be locked (LOCK mode), and no other thread will be able to access or modify it until T1 returns.

**Types of Thread Synchronization:**

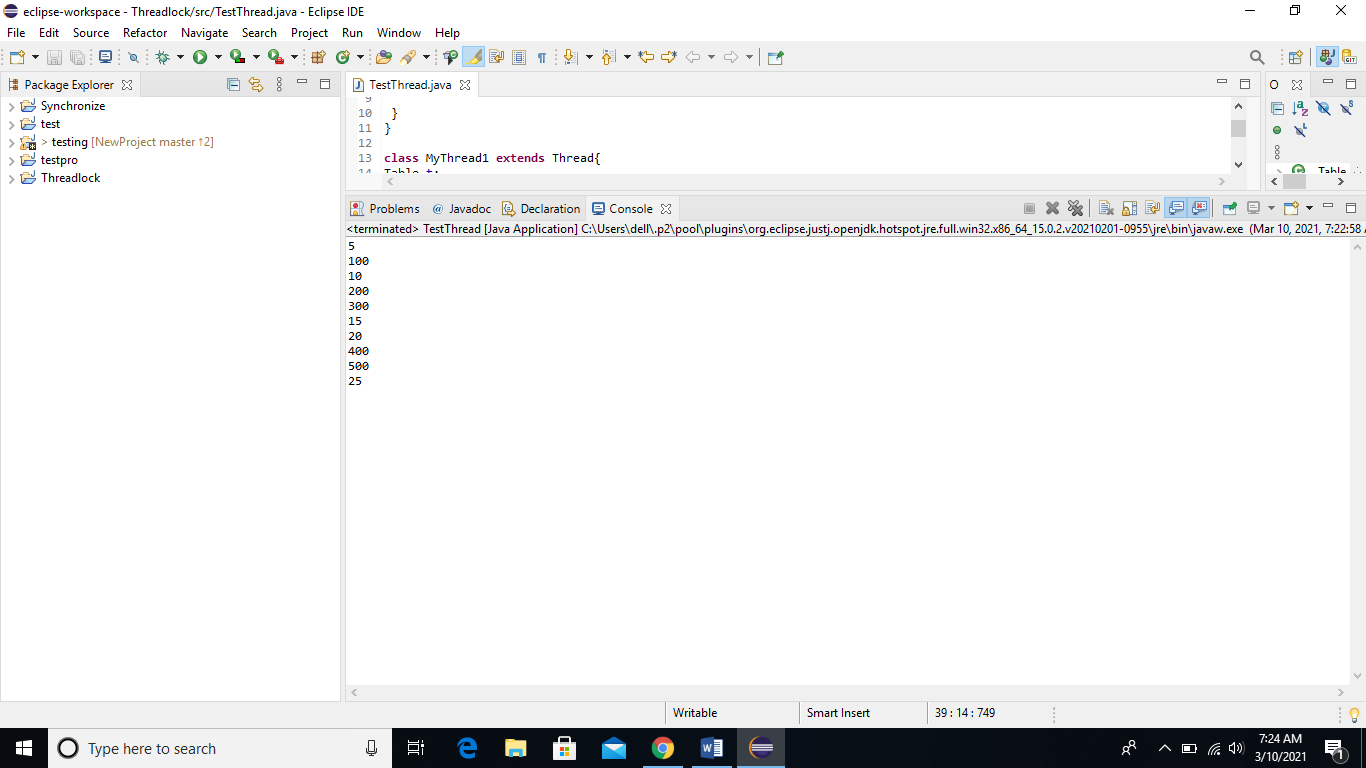
There are two types of thread synchronization mutual exclusive and inter-thread communication.

* 1. Synchronized method.
  2. Synchronized block.

**Example with no Synchronization**

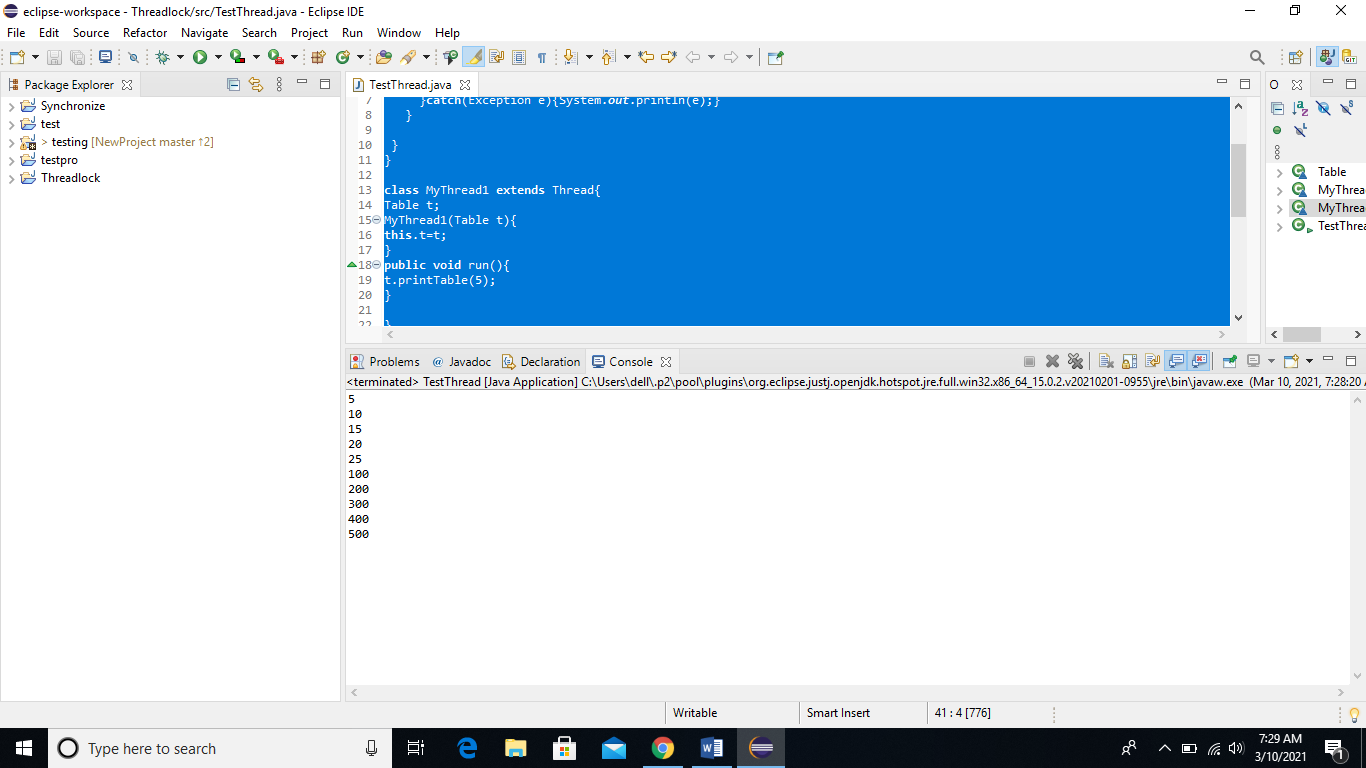
In this example, we are not using synchronization and creating multiple threads that are accessing display method and produce the random output.

|  |
| --- |
| **class** Table{  **void** printTable(**int** n){//method not synchronized  **for**(**int** i=1;i<=5;i++){  System.***out***.println(n\*i);  **try**{  Thread.*sleep*(400);  }**catch**(Exception e){System.***out***.println(e);}  }    }  }    **class** MyThread1 **extends** Thread{  Table t;  MyThread1(Table t){  **this**.t=t;  }  **public** **void** run(){  t.printTable(5);  }    }  **class** MyThread2 **extends** Thread{  Table t;  MyThread2(Table t){  **this**.t=t;  }  **public** **void** run(){  t.printTable(100);  }  }    **class** TestThread{  **public** **static** **void** main(String args[]){  Table obj = **new** Table();//only one object  MyThread1 t1=**new** MyThread1(obj);  MyThread2 t2=**new** MyThread2(obj);  t1.start();  t2.start();  }  } |



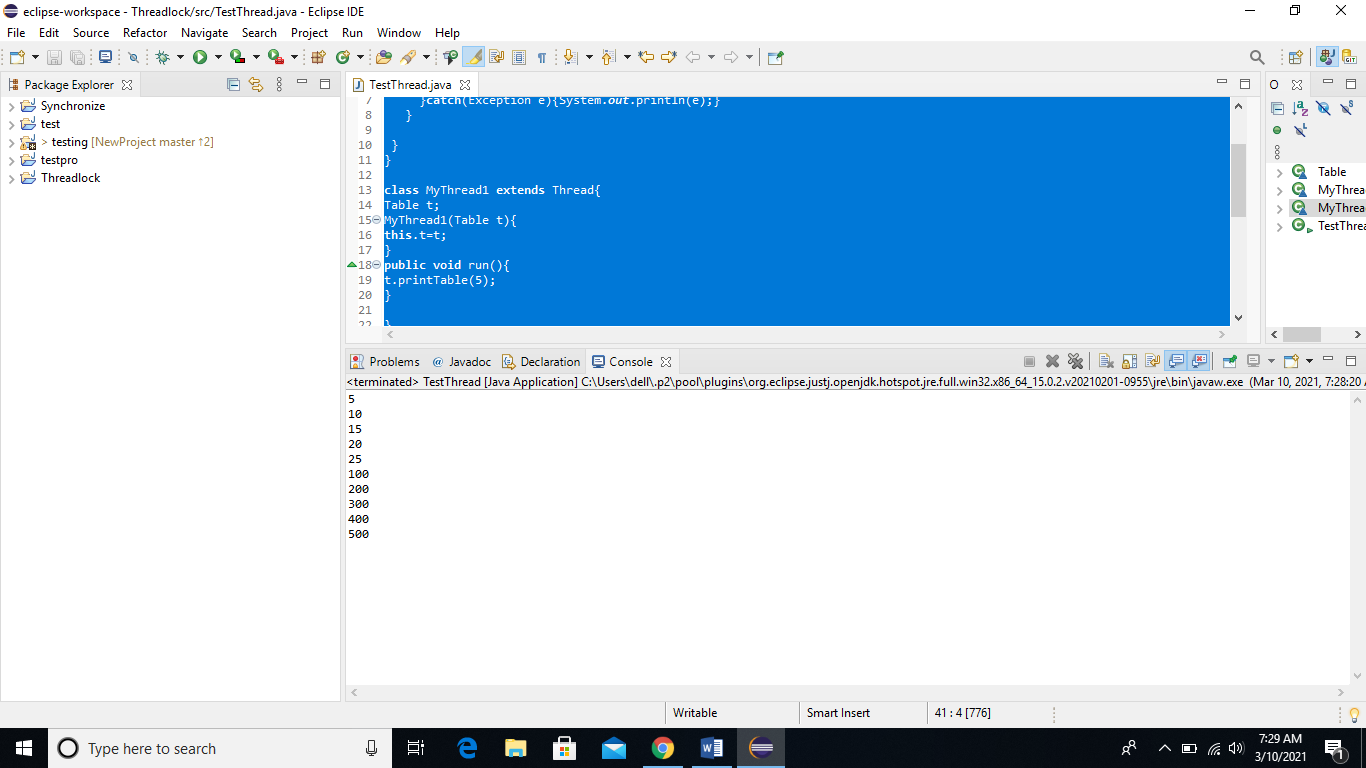
**Solution with Synchronization Method:**

|  |
| --- |
| **class** Table{  **synchronized** **void** printTable(**int** n){//synchronized method  **for**(**int** i=1;i<=5;i++){  System.***out***.println(n\*i);  **try**{  Thread.*sleep*(400);  }**catch**(Exception e){System.***out***.println(e);}  }    }  }    **class** MyThread1 **extends** Thread{  Table t;  MyThread1(Table t){  **this**.t=t;  }  **public** **void** run(){  t.printTable(5);  }    }  **class** MyThread2 **extends** Thread{  Table t;  MyThread2(Table t){  **this**.t=t;  }  **public** **void** run(){  t.printTable(100);  }  }    **public** **class** TestThread{  **public** **static** **void** main(String args[]){  Table obj = **new** Table();//only one object  MyThread1 t1=**new** MyThread1(obj);  MyThread2 t2=**new** MyThread2(obj);  t1.start();  t2.start();  }  } |



**Solution with Synchronization Block:**

|  |
| --- |
| **class** Table{    **void** printTable(**int** n){  **synchronized**(**this**){//synchronized block  **for**(**int** i=1;i<=5;i++){  System.***out***.println(n\*i);  **try**{  Thread.*sleep*(400);  }**catch**(Exception e){System.***out***.println(e);}  }  }  }//end of the method  }    **class** MyThread1 **extends** Thread{  Table t;  MyThread1(Table t){  **this**.t=t;  }  **public** **void** run(){  t.printTable(5);  }    }  **class** MyThread2 **extends** Thread{  Table t;  MyThread2(Table t){  **this**.t=t;  }  **public** **void** run(){  t.printTable(100);  }  }    **public** **class** TestThread{  **public** **static** **void** main(String args[]){  Table obj = **new** Table();//only one object  MyThread1 t1=**new** MyThread1(obj);  MyThread2 t2=**new** MyThread2(obj);  t1.start();  t2.start();  }  } |



**Example of inter thread communication in java:**

Inter-thread communication or Co-operation is all about allowing synchronized threads to communicate with each other.

Cooperation (Inter-thread communication) is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed. It is implemented by following methods of Object class:

* wait()
* notify()
* notifyAll()

**1) wait() method**

Causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

**2) notify() method**

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. Syntax:

public final void notify()

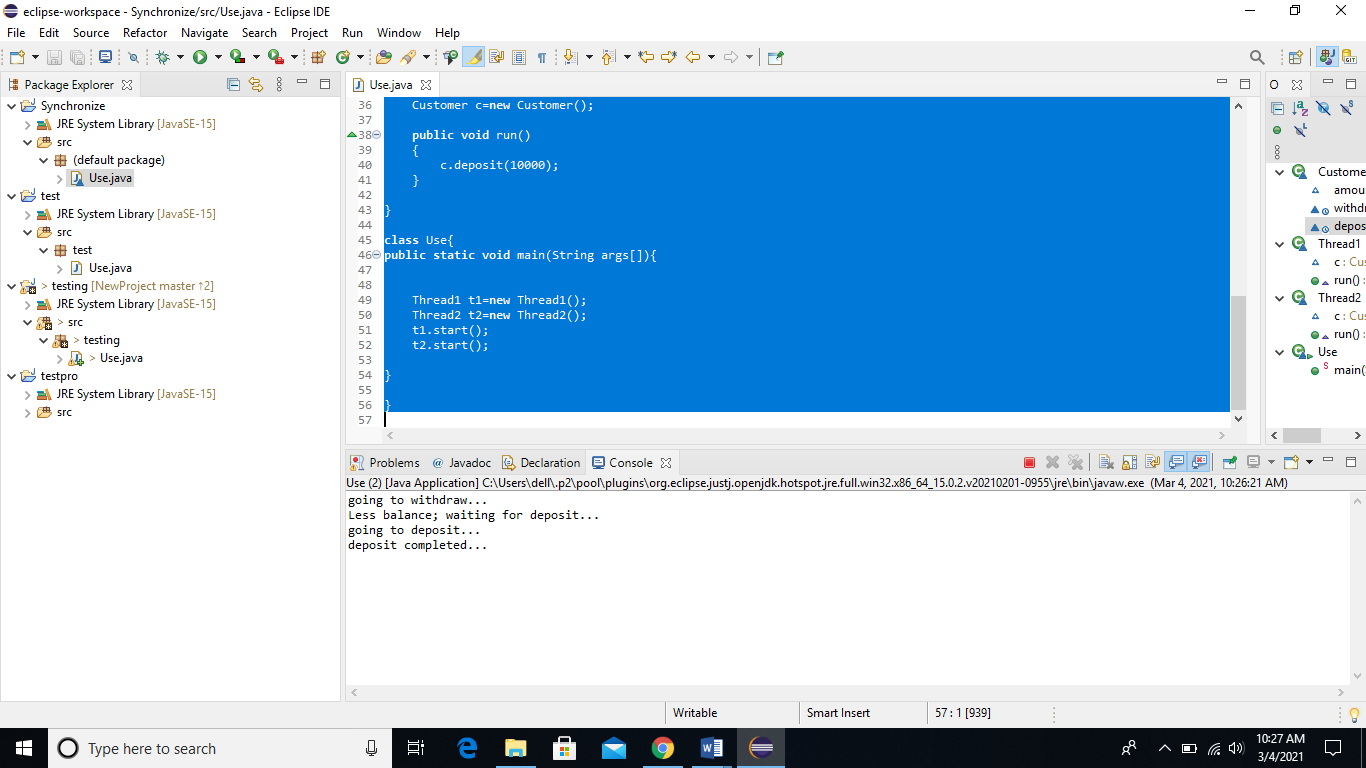
**3) notifyAll() method**

Wakes up all threads that are waiting on this object's monitor. Syntax:

public final void notifyAll()

Let's see the simple example of inter thread communication.

|  |
| --- |
| **class** Customer{  **int** amount=10000;    **synchronized** **void** withdraw(**int** amount){  System.***out***.println("going to withdraw...");    **if**(**this**.amount<amount){  System.***out***.println("Less balance; waiting for deposit...");  **try**{wait();}**catch**(Exception e){}  }  **this**.amount-=amount;  System.***out***.println("withdraw completed...");  }    **synchronized** **void** deposit(**int** amount){  System.***out***.println("going to deposit...");  **this**.amount+=amount;  System.***out***.println("deposit completed... ");  notify();  }  }  **class** Thread1 **extends** Thread  {  Customer c=**new** Customer();    **public** **void** run()  {  c.withdraw(15000);  }  }  **class** Thread2 **extends** Thread  {  Customer c=**new** Customer();    **public** **void** run()  {  c.deposit(10000);  }  }  **class** Use{  **public** **static** **void** main(String args[]){    Thread1 t1=**new** Thread1();  Thread2 t2=**new** Thread2();  t1.start();  t2.start();    }  } |



Lab Task:

1. Design a simple program of concurrency by implementing the scenario of two account holders in a joint bank account. (Hint: Total amount will be 50000, if ‘user A’ wants to withdraw 45,000 and ‘user B’ wants to withdraw 20,000) Apply mechanism of synchronization e.g. Block or Method for handling accessibility of multi-threads:
2. Create an inter thread communication program of printer job by implementing two threads, one for calculating the remaining pages in printer tray and other one will print the pages that are pending on queue. (Hint: If total pages are 10 and user sends job for 15 pages than print thread will be on wait and will be notified once available pages are equal or greater than printing pages).