**Advanced Java: Multi-threading Part 10 - Re-entrant Locks**

https://www.youtube.com/watch?v=fjMTaVykOpc&index=10&list=PLBB24CFB073F1048E

**2nd class – secondclass.java:**

**import** java.util.Scanner;

**import** java.util.concurrent.locks.Condition;

**import** java.util.concurrent.locks.Lock;

**import** java.util.concurrent.locks.ReentrantLock;

**public** **class** secondclass{

**private** **int** count = 0;

**private** Lock lock = **new** ReentrantLock();

**private** Condition cond = lock.newCondition();

**public** **void** increment(){

**for**(**int** i=0;i<10000;i++){

count++;

}

}

**public** **void** firstThread() **throws** InterruptedException{

lock.lock();

System.*out*.println("Waiting...");

cond.await();

System.*out*.println("Woken up!");

**try**{

increment();

} **finally** {

lock.unlock();

}

}

**public** **void** secondThread() **throws** InterruptedException{

Thread.*sleep*(1000);;

lock.lock();

System.*out*.println("Press the return key!");

**new** Scanner(System.*in*).nextLine();

System.*out*.println("Got return key!");

cond.signal();

**try**{

increment();

} **finally** {

lock.unlock();

}

}

**public** **void** finished(){

System.*out*.println("Count is: "+count);

}

}

**1st class – apples.java:**

**public** **class** apples {

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

**final** secondclass second = **new** secondclass();

Thread t1 = **new** Thread(**new** Runnable(){

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

**try** {

second.firstThread();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

});

Thread t2 = **new** Thread(**new** Runnable(){

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

**try** {

second.secondThread();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

});

t1.start();

t2.start();

t1.join();

t2.join();

second.finished();

}

}

**Result:**

Waiting...

Press the return key!

Got return key!

Woken up!

Count is: 20000

**Important notes:**

* Like *wait()* and *notify(),* *await()* and *signal()* interacts with each other. But one difference is that java.util.concurrent.locks.Condition; was imported.
* In case the *increment()* method might get interrupted, the programmer used **try** and **finally** to avoid that scenario – he still made it so the lock would unlock.