<http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90>

## [notifyAll 과 wait 사용시 주의점.](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90)

[JAVA](http://everysw.tistory.com/category/JAVA) 2013.07.16 22:39

자바에서 동기화 방법으로 synchronized 키워드와 notifyAll 그리고 wait 함수를 제공한다는 것은

많은 사람들이 잘 안다.

간단히 정리하면.

synchronized 키워드는 shared resource 에 대한 상호 배타적으로 접근할 수 있도록 지정할 때 사용하며,

wait 함수와 notifyAll 함수는 synchronized 키워드 만으로는 해결이 힘든 Producer / Consumer 시나리오와 같은 곳

에서 사용한다.

쓰레드가 wait  함수를 호출하게 되면 현재 가지고 있는 Lock 을 Release 하고 대기 상태에 빠지며,

쓰레드가 notifyAll 함수를 호출하게 되면 wait 함수를 호출하여 대기 상태에 빠져 있는 모든 쓰레드들을 깨우게 된다.

다만 많은 사람들이 잘못 이해하고 있는 부분이,

notifyAll 함수를 호출하게 되면 대기 상태에 있던 쓰레드들이 깨어나면서 Lock 도 획득하게 된다고 알고 있는 것이다.

notifyAll 함수는 대기 상태에 있던 쓰레드들을 깨울 뿐이지 깨어난 쓰레드들이 Lock 획득을 하도록 만들어 주진 않는다.

**즉 중요한 것은 notifyAll 함수 호출로 깨어난 쓰레드들은 synchorized 키워드 블록 안에 있으므로 같이**

**깨어난 다른 쓰레드들과 Lock 획득을 위해 경쟁하게 되며 그 중 한 쓰레드만이 Lock을 획득하게 된다는**

**것이다.**

그리고 이와는 별도로 wait, notifyAll 함수를 사용하면서 실수하기 쉬운 부분이 또 있다.

이 경우는 설명하기 위하여 문제가 발생하는 Producer / Consumer 예제를 만들어 보았다.

간단히 예제를 설명하면 Producer Thread 한 개와 Consumer Thread 가 여러개 있고 Producer Thread 가

주기적으로 저축을 하면 Consumer Thread 들이 Balance 가 0이 아닌 이상 계속 돈을 빼가는 시나리오다.

이때 Shared Resource 는 Balance 객체이며 여러 쓰레드가 동시 접근하는 것을 고려하여 synchronized 키워드를

사용하였다.

[view source](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90#viewSource)[print](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90#printSource)[?](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90#about)

public class Balance {

/\*

\* mBalance 는 Shared Resouce 이므로 mBalance 를 수정하는

\* withdraw 함수와 deposit 함수를 synchronized 키워드로 지정하였다.

\* 즉 withdraw 함수와 deposit 함수는 각각 atomic 하게 실행되며,

\* 한 쓰레드가 withdraw 함수를 실행하는 동안 다른 쓰레드가 deposit 함수를 호출하거나

\* withdraw 함수를 호출할 수 없다.

\*/

private int mBalance = 0;

Balance(int balance) {

mBalance = balance;

}

synchronized void withdraw(int value) {

try {

// if 문을 사용하여 현재 mBalance 가 0 인지 확인한다.

if(mBalance == 0)

wait(); // mBalance 가 0이면 Lock 을 릴리즈하고 대기 상태에 빠진다.

mBalance -= value;

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Withdraw. Current Balance : " + mBalance);

}

synchronized void deposit(int value) {

mBalance += value;

// wait 을 호출하여 대기 상태에 빠져 있는 모든 쓰레드들을 깨운다.

notifyAll();

System.out.println("Deposit. Current Balance : " + mBalance);

}

}

public class Producer implements Runnable {

private Balance mBalance;

Producer(Balance balance) {

mBalance = balance;

}

@Override

public void run() {

// 100번 루프를 돌면서 100씩 저금한다.

for(int i=0; i<100; i++) {

mBalance.deposit(100);

try {

Thread.sleep(1);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class Consumer implements Runnable {

private Balance mBalance;

public Consumer(Balance balance) {

mBalance = balance;

}

@Override

public void run() {

while(true) {

// 무한 루프를 돌면서 100씩 인출한다.

mBalance.withdraw(100);

}

}

}

public class Main {

public static void main(String[] args) {

Balance balance = new Balance(0);

Producer producer = new Producer(balance);

Consumer consumer = new Consumer(balance);

// Producer 쓰레드는 1개, Consumer 쓰레드는 10개를 만들어서 시작시킨다.

new Thread(producer).start();

for(int i=0;i<10;i++) {

new Thread(consumer).start();

}

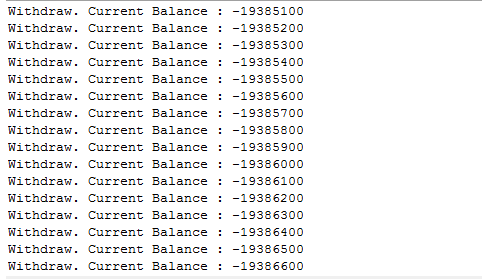
}

}

적절히 synchronized 와 wait 그리고 notifyAll 함수를 사용한 것처럼 보이지만,

위 코드는 심각한 문제를 가지고 있다.

일단 실행 해 보면 결과는 다음과 같다.



Balance 는 0 이하로 값이 떨어지면 안되는데 Withdraw 함수가 계속 호출되면서 값이 0이하로 계속 떨어지고 있다.

withdraw 함수 안에서 분명히 현재 balance 값이 0일때는 쓰레드를 wait 하게 만들어 놓았는데,

왜 그럴까? 코드를 다시 한번 보자.

[view source](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90#viewSource)[print](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90#printSource)[?](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90#about)

synchronized void withdraw(int value) {

try {

// if 문을 사용하여 현재 mBalance 가 0 인지 확인한다.

if(mBalance == 0)

wait(); // mBalance 가 0이면 Lock 을 릴리즈하고 대기 상태에 빠진다.

mBalance -= value;

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Withdraw. Current Balance : " + mBalance);

}

문제가 생기는 시나리오를 보자.

|  |
| --- |
| 0. 현재 balance 는 100  1. consumer #1 쓰레드가 스케쥴링을 받아서 withdraw 함수를 호출한다.  2. consumer #1 쓰레드가 100만큼을 withdraw 해간다.  3. balance 는 0이 된다.  4. consumer #2 쓰레드가 스케쥴링 받아서 withdraw 함수를 호출한다.  5. if 문에서 balance 를 체크하고 0 이므로 wait 함수를 호출하여 Lock 을 릴리즈 하고 대기 상태에 빠진다.  6. consumer #3 쓰레드가 스케쥴링 받아서 withdraw 함수를 호출한다.  7. if 문에서 balance 를 체크하고 0 이므로 wait 함수를 호출하여 Lock 을 릴리즈 하고 대기 상태에 빠진다.  8. producer 쓰레드가 스케쥴링을 받아서 deposit 함수를 호출한다.  9. producer 쓰레드가 100만큼을 deposit 하고 notifyAll 함수를 호출하여 대기 상태에 빠져 있던 consumer #2, consumer# 3  쓰레드를 깨운다.  10. consumer #2, #3 쓰레드는 깨어났지만 현재 synchronized 함수 안에 있으므로 #2와 #3 쓰레드가 Lock 획득을 위해 경쟁한다.  11. consumer #2 쓰레드가 Lock 을 획득한다.  12. consumer #2 쓰레드가 balance 에서 100만큼을 withdraw 해간다.  13. balance 는 0이 된다.  **14. consumer #3 쓰레드가 Lock 을 획득한다. 그리고 이어서 바로 balance 에서 100만큼을 withdraw 해간다.**  **15. balance 는 음수가 된다.** |

**문제는 14번 단계에서 발생한다.**

**consumer #3 쓰레드가 Lock 을 획득한 후  다시 한번 balance 가 0인지를 체크하여야 하는데 하지 않고**

**바로 balance 에서 값을 빼버린 것이다.**

즉 10번 단계에서 consumer #3 쓰레드는 consumer #2 쓰레드와의 Lock 획득 경쟁에서 져서 다음번 Lock 획득을

기다리고 있을 뿐 Lock을 획득하게 되면 wait 함수 이후로 실행이 되게 되는 것이다.

문제의 원인은 알았다.

이런 경우 해결은 어떻게 해야 할까?

**간단하다. wait 에서 깨어나고 나서 다시 한번 balance 가 0인지를 체크하면 된다.**

정상적인 withdraw 함수는 다음과 같다.

synchronized void withdraw(int value) {

try {

// while 문을 사용하여 현재 mBalance 가 0 인지 확인한다.

while(mBalance == 0)

wait(); // mBalance 가 0이면 Lock 을 릴리즈하고 대기 상태에 빠진다.

mBalance -= value;

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Withdraw. Current Balance : " + mBalance);

}

즉 위 코드처럼 while 문을 사용하여 wait 함수에서 깨어난 후에 while 조건문 안에서 다시 한번 balance 값을 체크하는 것이다.

인터넷을 보다보면 synchronized 키워드와 wait / notifyAll 함수 자체에 대한 사용법에 대해서는 설명해 놓는 곳은

많지만 위 경우와 같은 문제 시나리오를 알려주는 곳은 많지 않다.

그러나 wait / notifyAll 함수를 사용하면서 개발자가 쉽게 실수 할 가능성이 높은 부분이랄까...

한번 쯤은 생각해 보고 넘어갈 문제이다.

[신고](http://everysw.tistory.com/toolbar/popup/abuseReport/?entryId=7)

#### '[JAVA](http://everysw.tistory.com/category/JAVA)' 카테고리의 다른 글

|  |  |
| --- | --- |
| [**왜 Application 의 GUI Component 는 여러 쓰레드에서 동시 접근이 안될까?**](http://everysw.tistory.com/entry/%EC%99%9C-%EB%8C%80%EB%B6%80%EB%B6%84%EC%9D%98-GUI-Framework-%EC%97%90%EC%84%9C%EB%8A%94-%ED%95%9C-%EC%93%B0%EB%A0%88%EB%93%9C%EB%A7%8C%EC%9D%B4-GUI-Component-%EC%97%90-%EC%A0%91%EA%B7%BC%EC%9D%B4-%EA%B0%80%EB%8A%A5%ED%95%A0%EA%B9%8C)**(1)** | 2013.07.22 |
| [**notifyAll 과 wait 사용시 주의점.**](http://everysw.tistory.com/entry/notifyAll-%EA%B3%BC-wait-%EC%82%AC%EC%9A%A9%EC%8B%9C-%EC%A3%BC%EC%9D%98%EC%A0%90)**(0)** | 2013.07.16 |