Java Interthread Communication

Java provide benefits of avoiding thread pooling using inter-thread communication. The `wait()`, `notify()`, and `notifyAll()` methods of Object class are used for this purpose. These method are implemented as **final** methods in Object, so that all classes have them. All the three method can be called only from within a **synchronized** context

- `wait()` tells calling thread to give up monitor and go to sleep until some other thread enters the same monitor and call notify.
- `notify()` wakes up a thread that called wait() on same object.
- `notifyAll()` wakes up all the thread that called wait() on same object.

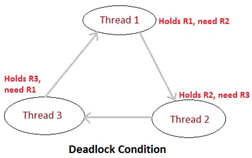
Difference between `wait()` and `sleep()`

wait()	sleep()
called from synchronised block	no such requirement
monitor is released	monitor is not released
gets awake when notify() or notifyAll() method is called.	does not get awake when notify() or notifyAll() method is called
not a static method	static method
wait() is generaly used on condition	sleep() method is simply used to put your thread on sleep.

Thread Pooling

Pooling is usually implemented by loop i.e to check some condition repeatedly. Once condition is true appropriate action is taken. This waste CPU time.

Thread Deadlock in Java



Deadlock Condition

Deadlock is a situation of complete Lock, when no thread can complete its execution because lack of resources. In the above picture, Thread 1 is holding a resource R1, and need another resource R2 to finish execution, but R2 is locked by Thread 2, which needs R3, which in turn is locked by Thread 3. Hence none of them can finish and are stuck in a deadlock.

Example

In this example, multiple threads are accessing same method that leads to deadlock condition. When a thread holds the resource and does not release it then other thread will wait and in deadlock condition wait time is never ending.

```
class Pen{}
class Paper{}

public class Write {

  public static void main(String[] args)
  {
     final Pen pn =new Pen();
     final Paper pr =new Paper();

     Thread t1 = new Thread() {
        public void run()
        {
            synchronized(pn)
            {
                System.out.println("Thread1 is holding Pen");
                try{
                    Thread.sleep(1000);
                }
```

Output:

```
Thread1 is holding Pen
Thread2 is holding Paper
```

← Prev
Next →