Topics

- Thread Deadlock?
- Inter Thread Communication

Thread Deadlock

- When two threads have a circular dependence on a pair of synchronized objects.
- Declaring methods synchronized ensures that threads executes them fully not partially. [Sometimes Not Enough to Ensure that a Program Runs Correctly]
- Assume two thread instances say 'T1' and 'T2' of class 'Producer' are adding string type greetings say "Hello" and "Welcome" into the same instance 'queue' of class 'Queue'. Assume all methods of the class 'Queue' are synchronized. The code that 'T1' and 'T2' use to add greeting into 'queue' is shown below

```
// run() Method of 'Producer' class
if(!queue.isFull())
{
    queue.add(…….); // Note add method is synchronized
    i++;
}
```

Thread Deadlock …

- In order to overcome this issue, the condition check [if(!queue.isFull())] has to be moved inside the add method

```
public synchronized void add(Object obj)
{
   while(queue.isFull())   // Note use of if is dangerous
       wait for consumer thread to vacate some space
}
```

This May Lead to a Deadlock Situation, If Currently Executing Thread Enters WAITING State Without Releasing the Lock

Inter Thread Communication

- **Methods Used for Inter Thread Communication**

1. ***final void wait()*** → Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.
2. ***final void wait(long timeout)*** → Causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.
3. ***final void notify()*** → Wakes up a single thread that is waiting on this object's monitor.
4. ***final void notifyAll()*** → Wakes up all threads that are waiting on this object's monitor.

Inter Thread Communication …

```
public class Queue
{
            private Object[ ]  elements;
            private int head;
            private int tail;
            private int size;
            // Constructor Method
            public Queue(int capacity)       {            }
            public synchronized Object removeFirst() throws interruptedException
            {
                        while(size() == 0) wait();       // wait(100); Timed Wait
                        ……………………
                        ……………………
                        notifyAll();
            } // End of Method
            public synchronized void add(Object anotherObject) throws interruptedException
            {
                        while(size == elements.length) wait();    // wait(100); Timed Wait
                        ……………………
                        ……………………
                        notifyAll();
            } // End of Method
}// End of class
```

Thank You