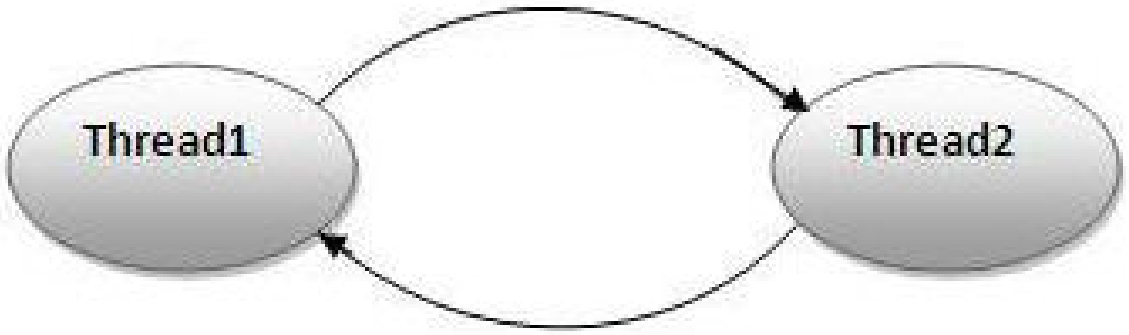
Programming Language II
CSE-215

Prof. Dr. Mohammad Abu Yousuf

yousuf@juniv.edu

Thread dead lock

- Deadlock in java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.



```
public class TestDeadlockExample1 {
  public static void main(String[] args) {
    final String resource1 = “Md. Rahim";
    final String resource2 = “Md. Karim";
    // t1 tries to lock resource1 then resource2
    Thread t1 = new Thread() {
      public void run() {
        synchronized (resource1) {
          System.out.println("Thread 1: locked res
ource 1");

          try { Thread.sleep(100);} catch (Exceptio
n e) {}

          synchronized (resource2) {
           System.out.println("Thread 1: locked res
ource 2");
          }
        }
      }
    };                          // 1
```

```
// t2 tries to lock resource2 then resource1
    Thread t2 = new Thread() {
      public void run() {
        synchronized (resource2) {
          System.out.println("Thread 2: locked reso
urce 2");

          try { Thread.sleep(100);} catch (Exception
 e) {}

          synchronized (resource1) {
           System.out.println("Thread 2: locked res
ource 1");
          }
        }
      }
    };
    t1.start();
    t2.start();
  }
}                               //2
```

Output: Thread 1: locked resource 1
Thread 2: locked resource 2

Thread interrupt

- An *interrupt* is an indication to a thread that it should stop what it is doing and do something else. It's up to the programmer to decide exactly how a thread responds to an interrupt, but it is very common for the thread to terminate.

Thread interrupt

- If any thread is in sleeping or waiting state (i.e. sleep() or wait() is invoked), calling the interrupt() method on the thread, breaks out the sleeping or waiting state throwing InterruptedException.

- If the thread is not in the sleeping or waiting state, calling the interrupt() method performs normal behaviour and doesn't interrupt the thread but sets the interrupt flag to true.

Thread interrupt

- **Example :** Suppose there are two threads and If one of the threads is blocked in an invocation of the wait(), wait(long), or wait(long, int) methods of the Object class, or of the join(), join(long), join(long, int), sleep(long), or sleep(long, int), methods of this class, then its interrupt status will be cleared and it will receive an InterruptedException, which gives the chance to another thread to execute the corresponding run() method of another thread which results into high performance and reduces the waiting time of the threads.

Thread interrupt

The 3 methods provided by the Thread class for interrupting a thread

- **public void interrupt()**
- **public static boolean interrupted()**
- **public boolean isInterrupted()**

```
class ICT extends Thread {
   public void run()
   {
      try {
         Thread.sleep(2000);
         System.out.println(“ICTICT");
      }
      catch (InterruptedException e) {
         throw new RuntimeException("Thread " +   "interrupted");
      }
   }
   public static void main(String args[])
   {
      ICT t1 = new ICT();
      t1.start();
      try {
         t1.interrupt();
      }
      catch (Exception e) {
         System.out.println("Exception handled");
      }
   }
}
```

```
// Example1: Java Program to illustrate the
// concept of interrupt() method
// while a thread stops working
```

```
Output:
Exception in thread "Thread-0"
java.lang.RuntimeException: Thread interrupted
```

```
class TestInterruptingThread1 extends Thread{
public void run(){
try{
Thread.sleep(1000);
System.out.println("task");
}catch(InterruptedException e){
throw new RuntimeException("Thread interrupte
d..."+e);
}
}
public static void main(String args[]){
TestInterruptingThread1 t1=new TestInterrupting
Thread1();
t1.start();
try{
t1.interrupt();
}catch(Exception e){System.out.println("Exception
 handled "+e);}
}
}
```

Example 2 of interrupting a thread that stops working

In this example, after interrupting the thread, we are propagating it, so it will stop working.

In the program, after interrupting currently executing thread, we are throwing a new exception in the catch block so it will stop working.

Output:

```
Exception in thread-0
java.lang.RuntimeException: Thread interrupted...
java.lang.InterruptedException: sleep interrupted at
A.run(A.java:7)
```

```
class TestInterruptingThread2 extends Thre
ad{
public void run(){
try{
Thread.sleep(1000);
System.out.println("task");
}catch(InterruptedException e){
System.out.println("Exception handled "+e);

}
System.out.println("thread is running...");
}
public static void main(String args[]){
TestInterruptingThread2 t1=new TestInterru
ptingThread2();
t1.start();
t1.interrupt();
}
}
```

Example of interrupting a thread that doesn't stop working

In this example, after interrupting the thread, we handle the exception, so it will break out the sleeping but will not stop working.

Output:

```
Exception handled
java.lang.InterruptedException: sleep interrupted
thread is running...
```

```
class TestInterruptingThread3 exten
ds Thread{
public void run(){
for(int i=1;i<=5;i++)
System.out.println(i);
}
public static void main(String args[])
{
TestInterruptingThread3 t1=new Tes
tInterruptingThread3();
t1.start();
t1.interrupt();
}
}
```

Output:1
2
3
4
5

Example of interrupting thread that behaves normally

If thread is not in sleeping or waiting state, calling the interrupt() method sets the interrupted flag to true that can be used to stop the thread by the java programmer later.

What about isInterrupted and interrupted method?

```
public class TestInterruptingThread4 extends Thread{
public void run(){
for(int i=1;i<=2;i++){
if(Thread.interrupted()){
System.out.println("code for interrupted thread");
}
else{
System.out.println("code for normal thread");
}
}//end of for loop
}
public static void main(String args[]){

TestInterruptingThread4 t1=new TestInterruptingThread4();
TestInterruptingThread4 t2=new TestInterruptingThread4();

t1.start();
t1.interrupt();
t2.start();
}
}
```

The isInterrupted() method returns the interrupted flag either true or false. The static interrupted() method returns the interrupted flag afterthat it sets the flag to false if it is true.

Output:
Code for interrupted thread
code for normal thread
code for normal thread
code for normal thread

Thank you