UTA

Java Threads

- Thread Creation
- Thread Synchronization
- Thread States And Scheduling
- Short Demo

UTA

Thread Creation

There are two ways to create a thread in Java:

- Extend the Thread class
- Implement the Runnable interface

The Thread class

```
class A extends Thread {
    public A (String name) { super (name); }
    public void run () {
        System.out.println("My name is " + getName());
    }
}
class B {
    public static void main (String [] args) {
        A a = new A ("mud");
        a.start ();
    }
}
```


The Runnable interface

```
class A extends ... implements Runnable {
    public void run () {
        System.out.println("My name is " + getName () );
    }
}
class B {
    public static void main (String [] args) {
        A a = new A ();
        Thread t = new Thread (a, "mud, too");
        t.start ();
    }
}
```

UTA

Java Threads

- Thread Creation
- Thread Synchronization
- Thread States And Scheduling
- Short Demo

UTA

Lock

Each Java object is implicitly associated with a lock.

To invoke a synchronized method of an object, a thread must obtain the lock associated with this object.

The lock is not released until the execution of the method completes.

The locking mechanism ensures that at any given time, at most one thread can execute any synchronized method of an object.

Important: Lock is per object (NOT per method)!

UTA

wait, notify and notifyAll

The execution of wait on an object causes the current thread to wait until some other thread to call notify or notifyAll.

A thread must own the object lock before it invokes wait on an object. The execution of wait will also release the lock.

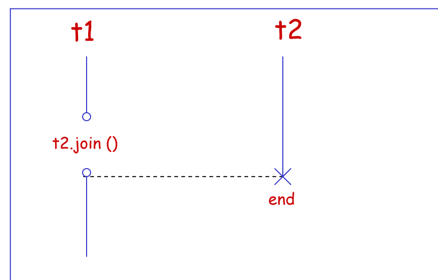
When a waiting thread is notified, it has to compete and reacquire the object lock before it continues execution.

What's the difference between notify and notifyAll?

UTA

join

A thread t1 can wait until another thread t2 to terminate.



UTA

interrupt

interrupt allows one thread to send a signal to another thread.

It will set the thread's interrupt status flag, and will throw a ThreadInterrupted exception if necessary .

The receiver thread can check the status flag or catch the exception, and then take appropriate actions.

UTA

Other Thread methods

Method sleep puts the running thread into sleep, releasing the CPU.

Method suspend suspends the execution of a thread, which can be resumed later by another thread using method resume.

Method stop ends the execution of a thread.

Note that suspend, resume, and stop has been deprecated in Java 2. (For more info, refer to http://java.sun.com/j2se/1.4.1/docs/guide/misc/threadPrimitiveDeprecation.html.)

UTA

Daemon Thread

A daemon thread is used to perform some services (e.g. cleanup) for other threads.

Any thread can be marked as a daemon thread using setDaemon (true).

A program terminates when all its non-daemon threads terminate, meaning that daemon threads die when all non-daemon threads die.

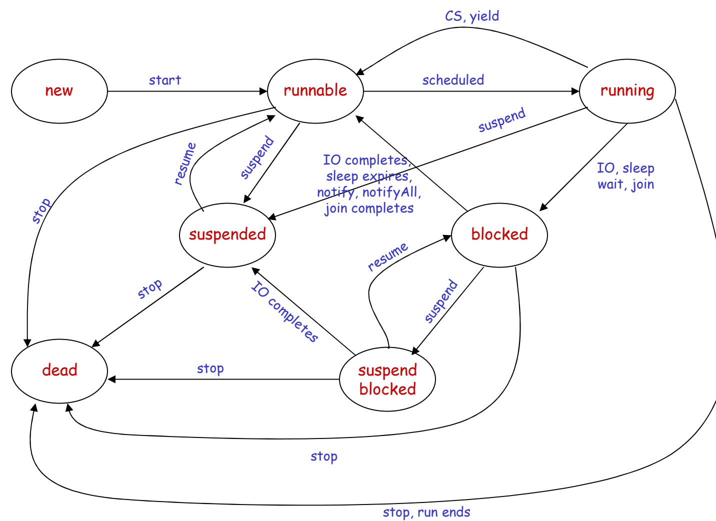
UTA

Java Threads

- Thread Creation
- Thread Synchronization
- Thread States And Scheduling
- Short Demo

UTA

State Transition Diagram



UTA

Scheduling

In general, there are two types of scheduling: non-preemptive scheduling, and preemptive scheduling.

In non-preemptive scheduling, a thread runs until it terminates, stops, blocks, suspends, or yields.

In preemptive scheduling, even if the current thread is still running, a context switch will occur when its time slice is used up.

UTA

Priorities

Each thread has a priority that also affects their scheduling to run.

If a thread of a higher priority enters the runnable set, the currently running thread will be preempted by the new thread.

UTA

Java Threads

- Thread Creation
- Thread Synchronization
- Thread States And Scheduling
- Short Demo