Practice of Programming using Java
Lecture 8     June 29, 2006. 6-8 pm

Rushikesh K Joshi
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
A Thread is a unit of concurrency.
For example: A thread sends a flow into a tank, whereas, another thread takes out the liquid.
Specifying Threads

Class MyThread extends Thread {

...  
...  
...  
public void run ( ) {.... };  
...  
}

Starting a Thread and the run () method

To start a thread, send start() message to a thread object

For example:

```java
MyThread t = new MyThread();
t.start();
```

The run() method starts executing concurrently while start() returns in the calling thread. When run() completes, the thread terminates
What if two threads invoke methods on a shared object?

```java
int height = 20;
tmp = height;
height = tmp + 1;

T1
```

```java
int height = 20;
tmp = height;
height = tmp - 1;

T2
```
Synchronized Methods

Declare a method as *synchronized* to guarantee mutually exclusive execution on an object.

An invocation of a synchronized method locks the object and another synchronized method cannot execute when the object is locked.

```java
class A {
    private int value;
    public synchronized readVal () {return value;}
    public synchronized incrVal() {value=value+1;}
    ...
}
```
public class PingPong extends Thread {
    private String word;
    private int d;
    public PingPong (String whatToSay, int delayTime) {
        word = whatToSay;
        d = delayTime;
    }
}
public void run() {
    try {
        while (true) {
            System.out.println(word + "\n");
            sleep(d);
        }
    }
    catch (InterruptedException e) {
        System.out.println("exception occurred\n");
        return;  // end this thread
    }
}
public static void main(String[] args) {
    PingPong T1 = new PingPong("Ping", 1000);
    PingPong T2 = new PingPong("Pong", 500);

    System.out.println("main thread starts");
    T1.start();
    T2.start();
    System.out.println("main thread ends");
}
}
Class X {
    
    public f() {
        
        ...  
        synchronized (anObj) {
            .. some code C..
        }
    }
}

Code C executes once anObj is locked
A thread can wait for a condition to occur by calling `wait()`

The thread waits on an object's monitor

A wait automatically unlocks the monitor object

Locking can occur through synchronized methods

Another thread, the owner of the object's monitor can wake up a waiting thread by sending it a `notify()` message

If a thread is executing a synchronized method on the object, synchronized statement on the object, synchronized static method, it becomes owner of that object's monitor.

```java
synchronized void someFunc() {
    while (somecondition) wait();
    ... program logic ..
}
```
Wait and Notify..

wait(long timeout): waits till a notification or till timeout (in ms) expires

wait (long timeoutMs, int timeoutNanoSec): A fine grained wait (ms+ns)

notify(): notifies at most one thread that is waiting

notifyAll(): notifies all threads that are waiting
Some Scheduling Aspects

sleep(long ms) : the executing thread sleeps for given milliseconds

sleep (long ms, int ns) : fine grain sleep

yield () : the executing thread yields so that other thread which is ready to run can run. It generates an explicit context switch between threads
Runnable Interface

Runnable is an interface that exports method:

\[ \text{public void run()} \]

A class can implement the run() method by inheriting the interface, instead of inheriting from class Thread

- possible to use multiple inheritance
- overheads of Thread class are avoided if the class needs only to be runnable

A Thread object can be created from a runnable object