# Multithreading and Synchronization

## What is a thread?

- It's a program counter and a stack
- All threads share the same memory space
  - take turns with the CPU, for a uni-processor
  - run concurrently, on a multiprocessor
- Example: Web browser
  - one thread for I/O
  - one thread for each file being downloaded
  - one thread to render web page
- Running thread might
  - yield, sleep, wait for I/O or **notify**, be pre-empted



## Writing Multi-threaded Code

- Need to control which events can happen simultaneously
  - e.g., update and display methods for a class
- Usually only covered in OS/DB courses
  - so few programmers have lots of training
- Can get inconsistent results or deadlock
  - problems often not easily reproduced
- Easy to get multi-threading, without trying
  - Graphical User Interfaces (GUI's)
  - Remote Method Invocation



## Extending class Thread

- Can build a thread class by extending **java.lang.Thread**
- Must supply a **public void run()** method
- Start a thread by invoking the **start()** method
- When a thread starts, executes **run**()
- When **run()** returns, thread is finished/dead



## Simple thread methods

- **void start()**
- **boolean isAlive()**
- **void setDaemon(boolean on)**
  - if only daemon threads running, VM terminates
- **void setPriority(int newPriority)**
  - thread scheduler might respect priority
- **void join() throws InterruptedException**
  - waits for a thread to die/finish



## Simple static thread methods

- Apply to thread invoking the method
  - **void yield()**
  - **void sleep(long milliseconds) throws InterruptedException**
  - **Thread currentThread()**

## Runnable interface

- Extending **Thread** means can't extend any other class
- Instead **implement Runnable**
  - declares that the class has a **void run()** method
- Can construct a new **Thread**
  - and give it an object of type **Runnable** as an argument to the constructor
  - **Thread(Runnable target)**
  - **Thread(Runnable target, String name)**



## Thread Example

```
public class ThreadDemo implements Runnable {
  public void run() {
    for (int i = 5; i > 0; i--) {
          System.out.println(i);
          try { Thread.sleep(1000);  }
          catch(InterruptedException e) { };  }
    System.out.println("exiting " + Thread.currentThread());
  }
  public static void main(String [] args) {
    Thread t = new Thread(new ThreadDemo(),"Demo Thread");
    System.out.println("main thread: " + Thread.currentThread());
    System.out.println("Thread created: " + t);
    t.start();
    try { Thread.sleep(3000);  }
    catch (InterruptedException e) {};
    System.out.println("exiting " + Thread.currentThread() );
  }
}
```



## InterruptedException

- A number of thread methods throw it
  - really means: wakeUpCall
- **interrupt()** sends a wakeUpCall to a thread
- Won't disturb the thread if it is working
  - but if thread attempts to sleep
  - it will get immediately woken up
- Will also wake up the thread if it is already asleep
- Thrown by **sleep(), join(), wait()**



## Be careful with threads

- Under some implementations of JVM
  - a thread stuck in a loop will *never* yield by itself
- Preemptive scheduling would guarantee it
  - but not supported on all platforms
- Put **yield()** into loops
- I/O has highest priority, so should be able to get time on CPU



## Another thread example

```
class UnSyncTest extends Thread {
        String msg;
        public UnSyncTest(String s) {
                msg = s;  start();  }
        public void run() {
                System.out.println("[" + msg);
                try { Thread.sleep(1000); }
                catch(InterruptedException e) {}
                System.out.println("]");  }
        public static void main(String [] args) {
                new UnSyncTest("Hello");
                new UnSyncTest("UnSynchronized");
                 new UnSyncTest("World");  }
}
```



## Daemon threads

- A thread can be marked as a daemon thread
- By default, thread acquires status of thread that spawned it
- When no threads running except daemons
  - program execution terminates