Multitasking

What is multitasking?
Why do we need multitasking?
- A long running process should not block all other processes
- Fully utilize the resources of a computer
  - CPUs, graphic card, hard drives etc.

Multitasking within a Process – Threads

Regular process
```
{  
    // do task 1
    
    // do task 2
}
```

Process with two threads
```
{  
    // do task 1
    // do task 2
}
```

Creating A Thread

Subclass Thread class
- [http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Thread.html](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Thread.html)

Implement Runnable interface
- [http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Runnable.html](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Runnable.html)

Thread Example

- A program performs two tasks
  - Calculate Fibonacci(n)
  - Download a web page
- Without thread: ThreadTest1.java
- With thread: ThreadTest2.java
Subclass Thread Class

- class Foobar extends Thread
- Override run() method
- Thread newThread = new Foobar();

Implement Runnable Interface

- class Foobar implements Runnable
- Implement run() method
- Thread newThread = new Thread(new Foobar())

How do we choose between these two approaches??

Run a Thread

- start() in the Thread class
- start() is non-blocking

Without join()

Collaboration between Processes/Threads

- Processes
  - Do not share address space
  - Collaborate through message passing
- Threads
  - Share address space
  - Collaborate through shared memory (usually faster than message passing)
Life Cycle of a Thread

- **New Thread**
- **Running (Runnable)**
- **Not Runnable**
- **Dead**

```
start
yield
run method terminates
```

Scheduling

- What happens in the running/Runnable state?
- Scheduling – pick a thread from the Runnable threads and run it
  - Time slicing
  - JVM default: *Fixed Priority Scheduling*

Fixed Priority Scheduling

- Threads with higher priority are run first
- Threads with the same priority are run in a round-robin manner.
- Threads with lower priority are only run when high priority threads are either dead or not Runnable.
- Preemptive – current thread may be stopped if there’s a thread with higher priority is Runnable

Runnable → Not Runnable

- `sleep()` method is invoked
- `wait()` method is invoked
- Blocked on I/O

Not Runnable → Runnable

- Sleep time expires
- `notify()` or `notifyAll()` method is invoked
- I/O is completed

Producer/Consumer Example

- A *producer* thread writes 0, 1, 2,..., 9 into a buffer
- A *consumer* thread reads from the buffer
- If two threads are perfectly synchronized, the consumer thread should read 0, 1, 2, 3,..., 9, but ...
From Non-synchronized to Synchronized

- `Thread.sleep(1000)` — just to make things more interesting
- `wait()` and `notify()`
- `synchronized`

Beyond Basics

- High-level Thread API
  - Timer and SwingWorker
- Semaphores, locks, conditions
- Scheduling
- Deadlock and starvation

- *So take CS440*