CISC 3120
Design & Implementation of Software Applications I

Lecture #11 – Introduction to Threading

M. Meyer
Course Page: http://www.sci.brooklyn.cuny.edu/~meyer/CISC3120
Content

- The Thread Class
  - sleep()
- Blinking
- Efficiency
- Locking
- Multi-Threading
Multi-Tasking (Multi-Programming)

- Multitasking is the ability of a computer to run multiple programs, or task, at the same time.
  - SingleTasking where one process must entirely finish before another can begin (MS-DOS is primarily a single-tasking environment).
- On a single-processor multitasking system, multiple processes don't actually run at the same time (there's only 1 processor.)
  - Instead the processor switches among the tasks (processes) that are active at any given time.
  - This happens so fast that it appears to the user as though the computer is executing all of the tasks at once.
- Multitasking also allows the computer to make good use of the time it would otherwise spend waiting for I/O devices and user input--that time can be used for some other task that doesn't need I/O at the moment.
Multi-Processing

- Multiprocessing is the coordinated processing of one or more programs by more than one CPU.
- Multiprocessing is a general term that can mean the dynamic assignment of a program to more than one computer, more than one processor.
- In symmetric (or "tightly coupled") multiprocessing, multiple processors share memory and the I/O bus or data path and a single operating system.
- In massively parallel (or "loosely coupled") processing, each processor has its own operating system and memory, but an "interconnect" arrangement of data paths allows messages to be sent between processors.
Multi-Threading

- Multi-threading allows multiple threads to exist within the context of a single process.
  - A thread is a light-weight process.
  - A thread is the smallest sequence of programmed instructions that can be managed independently by an operating system scheduler.

- When multiple threads are within a single larger process (program) they share the process' resources, but are able to execute independently (including pause and terminate).
Multithreading VS. Multiprocessing

- **Multithreading** has advantages
  1. Threads are lightweight compared to processes
  2. Threads share the same address space and therefore can share both data and code
  3. Context switching between threads is usually less expensive than between processes
  4. Cost of thread intercommunication is relatively low that of process intercommunication
  5. Threads allow different tasks to be performed concurrently.
Core Methods of Java

- The following figure shows the methods that are members of the **Object** and **Thread** Class that are useful for multithreading.

<table>
<thead>
<tr>
<th><strong>Object</strong></th>
<th><strong>Thread</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>notify()</td>
<td>sleep()</td>
</tr>
<tr>
<td>notifyAll()</td>
<td>yield()</td>
</tr>
<tr>
<td>wait()</td>
<td></td>
</tr>
</tbody>
</table>
Thread Creation

- There are two ways to create threads in Java;
  1. By extending the Thread class (java.lang.Thread)
  2. Implement the Runnable interface (java.lang.Runnable)
- We will look at both of these methods.
- We will start by extending the Thread class.
- Note: Every program can be considered to reside within its own thread, so when we talk about creating threads, we really mean creating additional threads within a program.
There is a whole lot more…

- But it can wait…

You can do this!