

## Threads

11-21-2002

---

---

---

---

---

---

---

---

## Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?

---

---

---

---

---

---

---

---

## Motivation

- There are many times when we want programs to effectively be able to do two or more things at once. Structuring programs to model this type of behavior in a single thread of control can be quite difficult. It also doesn't allow us to take advantage of multiple processors that might be present.
- Multithreading deals with this. It gives us multiple execution streams in a shared memory space.

---

---

---

---

---

---

---

---

## The Thread Class

- Most modern languages have the ability to do multithreading. It is easier in some than in others.
- Java provides a simple way of doing this with the `java.lang.Thread` class.
- This class can also be useful even if you aren't using multiple threads because it has static methods that will impact the behavior of the current thread.

---

---

---

---

---

---

---

---

## Spawning Threads

- To start a new thread, you simply need to create a `Thread` object and pass it an instance of a `java.lang.Runnable` object.
- `Runnable` is an interface with one method in it, `run`. When the thread object's start method is called, the other thread becomes active, and it will begin execution at the `run` method of the `Runnable` object. Control returns to the original thread.

---

---

---

---

---

---

---

---

## Complexities of Threads

- There are some problems with using threads and they arise from the fact that you never know when one will release control and another one will get it.
- The most serious types of problems with this are when two threads are effectively trying to access the same memory at the same time. As such, we need a new construct to prevent the wrong thread from taking control at certain times.

---

---

---

---

---

---

---

---

## Thread Priority

- When a thread releases control and it is time to pick a new thread to execute, the highest priority thread will get control next.
- You can set the priority of threads with the `setPriority` method of `Thread`.
- You can force a thread to release control before it stops by calling the `sleep` or `yield` methods.

---

---

---

---

---

---

---

---

## Synchronized Methods

- The primary way that you can prevent problems with sections of code trying to access the same memory at once is through the use of the `synchronized` keyword.
- When this keyword is put in front of a function, it puts a lock on the monitor for that object/class that prevents other synchronized methods from executing.

---

---

---

---

---

---

---

---

## wait/notify

- The other way to control threads is with the `wait` and `notify` methods. One thread can wait when it needs something to happen. When that thing happens, the other method needs to call the original threads `notify`.
- This is more efficient than having a tight loop that just checks a flag variable.

---

---

---

---

---

---

---

---

## Timers

- If you just want something to happen at regular intervals, you can use objects like `java.util.Timer` or `javax.swing.Timer`.
- Each of these objects operates slightly differently. If what you are doing will interact with Swing objects the second is recommended. Otherwise the first should be more generally used.

---

---

---

---

---

---

---

## Minute Essay

- Why do we need to be able to synchronize threads?
- Read chapter 14 for next class.

---

---

---

---

---

---

---