EECS 1720: Threads

Yves Lesperance
09/02/2017

Multithreading

• When multiple threads/processes run concurrently/in parallel
• Often timesliced, but may run in parallel on a multiprocessor
• Many applications

Multithreading basics

• Package tasks to be run by threads in classes that implement the Runnable interface
• Put code performing task in the run() method
• Create a thread and use it to execute the task
The Class Thread

- To create a thread:
  ```java
  Thread t = new Thread(runnableObject);
  ```
- Or
  ```java
  Thread t = new Thread();
  ```
- To start running it:
  ```java
  t.start();
  ```
- To make it go to sleep:
  ```java
  Thread.sleep(milliseconds);
  ```

The Runnable Interface

- Packages some task/code to be run by a thread
- Must implement a method
  ```java
  public void run()
  ```
  that contains the code/task to run
- An object that implements the Runnable interface can be passed to a Thread constructor

Beyond the Basics

- Threads can be interrupted
- Need to ensure mutual exclusion when state of data structures is updated; can use locks for this
- Need to ensure that deadlock (or deadly embrace) cannot occur
- Deadlock occur when a thread acquires a lock and then must wait for another thread to do some work before proceeding, but where the second thread needs the lock to proceed
- May need to synchronize threads, e.g. producer and consumer threads with a shared buffer