Outline

- Thread review
- Stopping a thread
- java.util.Timer
- Swing threads
  - javax.swing.Timer
- ProgressMonitor
- invokeLater
Homework #3

- Not “getParameters”
- Should be “setParameters”
Thread Review I

- Threads
  - Java objects
  - Independent paths of execution
  - Share memory and code
- To define a thread’s behavior
  - run()
Threads Review II

- Threads only appear to run simultaneously
  - only a single thread executes at a time
  - each thread runs for a time and then is replaced by another

- Priority
  - determines which available thread is allowed to run
Thread Review III

- Java event handling
  - takes place in a single thread
  - other system threads

- Threads may have resource conflict
  - share the processor with sleep() and yield()
  - achieve exclusive use through synchronized methods
  - coordinate using wait() and notify()
Thread Review II
Thread Review IV

- **Lifecycle methods**
  - start = transition to runnable
  - end of run method = transition to dead
  - transition to non-runnable
    - wait ()
    - sleep ()
    - blocked
  - transition back to runnable
    - notify ()
    - end of sleep
    - resource unlocked
Why not stop()

- Dead thread drops its locks
  - synchronized method may be only partially executed
  - corrupt state
Example

```java
public void run ()
{
    while (true)
    {
        ... do something ...
    }
}
```
Alternative I (exit variable)

```java
private boolean m_isRunning;
public synchronized void setIsRunning (boolean newVal)
{   m_isRunning = newVal; }
public synchronized boolean IsRunning ()
{   return m_isRunning; }
public void run ()
{
    setIsRunning(true);
    while (isRunning())
    { ... do something ... }
}
```
To Stop the Thread

thread.setRunning (false)
Requirements

- Inner loop
- Exit variable checked regularly
public void run ()
{
    try
    {
        while (true)
        {
            foo.wait();
            ...
            do something ...
        }
    } catch (InterruptedException e)
    {
        .. clean up ...
    }
}
To Stop the Thread

thread.interrupt();
Requirements

- Thread is in “wait” state in its inner loop
Alternative III (i/o)

public void run ()
{
    try
    {
        while (true)
        {
            byte [] buffer = inputStream.readBytes(4000);
            ... do something ...
        }
    }
    catch (IOException e)
    {
        .. clean up ...
    }
}
To Stop the Thread

inputStream.close();
Requirement

- Thread is waiting for I/O in its inner loop
Three ways to stop

- Rapid inner loop
  - use a loop exit variable
- Wait state
  - call interrupt()
  - use the interrupted exception
- Waiting on I/O
  - close the I/O stream
  - use the IOException
Periodic Action

- **Examples**
  - Check the mail server every 10 minutes
  - Animate something on the screen
  - Autosave

- **Need**
  - a thread that sleeps for a specified period
  - then runs
  - possibly repeats
java.util.Timer

- Timer
  - schedule (TimerTask, delay)
  - schedule (TimerTask, delay, period)

- TimerTask
  - implements Runnable
java.util.Timer Lifecycle

- Timer created
- Task scheduled
  - wait setup in Timer thread
- Time arrives
  - TimerTask run method called
  - (not a separate thread)
- Timer canceled
  - timer.cancel()
javax.swing.Timer

- Timer
  - Timer (delay, ActionListener)

- ActionListener
  - actionPerformed (ActionEvent)
javax.swing.Timer Lifecycle

- ActionListener created
- Timer created
- Timer started
  - timer.start()
  - wait setup (in Timer thread)
- Time arrives
  - actionEvent created and inserted in event queue
  - ActionListener handles event
- Timer canceled
  - timer.stop
Progress Monitoring
Example

- Add progress monitor to ImageBrowser
- What needs to happen?
  - ProgressMonitor dialog must open
  - Monitor must be updated
  - Cancel/Completion must be handled
ProgressMonitor class

- ProgressMonitor(Component parentComponent, Object message, String note, int min, int max)
- setProgress
  - tell the dialog to change progress indication
Example
Note

- Updating must happen in EH thread
  - Swing timer ensures this
- If updating from another thread
  - must place updates into the EH thread
Interaction between threads and UI

- Swing components are not thread-safe
  - methods not synchronized
  - example

- Solution
  - only modify components from with the EH thread
  - after the component has been “realized”
Accessing the EH Thread

- EventQueue.invokeLater (Runnable)
- What happens
  - event inserted into event queue
  - when it comes to top
  - run method of object is called
- Also
  - invokeAndWait(Runnable)
  - not as useful