Java Threads

- A thread is computer code being executed.
- More than one thread can be executed simultaneously (actually interleaved).
  - The code for the threads can be the **same**, or different.
  - Each thread has it's own state, **sort of**.
  - Threads can **share variables**, and modify the variables they share.
- Programs with > 1 thread are called "concurrent programs".

Timing of Threads

- Threads don't progress in lock-step fashion.
- One may be started and another stopped in an **unpredictable** fashion by the operating system.
- This behavior is called **asynchronous**.

Similar Idea: Processes

- A process is also code in execution.
- Typically processes don't share variables, although limited sharing is possible.
- Multiple processes is common in, e.g. UNIX.
- Processes are "heavy weight", threads are "light weight".
- Weight refers to the cost of switching the processor from one to another.

Why are Threads Useful?

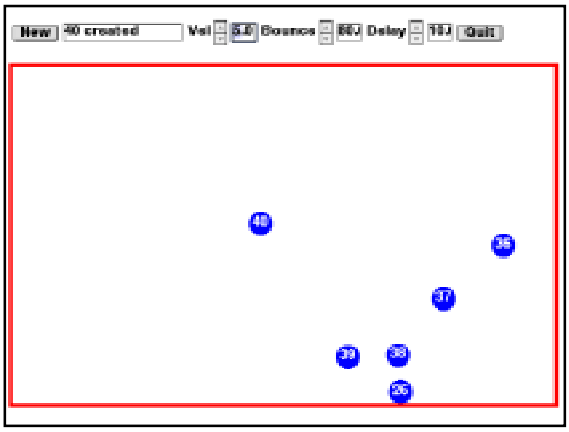
- May wish to have multiple activities going on at once.
- Don't want one activity's waiting (e.g. for an event) to stop the other activities.
- This is only doable with threads (or processes).

Thread Example

- On thread is a **computational** one, that occasionally needs to wait for input from the outside, say from an input stream of characters.
- Another thread may be a **graphical user interface**, responding to mouse events.
- We don't want **waiting** for input to hold up the graphics, or waiting for a click to hold up the computational thread.

Thread Example

- Bouncing Balls Example
- Each ball is run by a separate thread.
- Each thread can, in principal, be **suspended** and started independently of the others.
- If a ball is "clicked" in mid-air, it will suspend, and resume if clicked a second time.



Two Ways to Have Threads in Java

- extends Thread
  - Thread is a base class with threading capability.
- implements Runnable
  - Runnable is an interface that requires method
    - void run()
- The latter is preferred, because it does not take away your ability to inherit from another class (multiple inheritance is not allowed in Java).

Using "implements Runnable"

- The class that implements Runnable still needs to contain a Thread.
- This Thread is what controls starting and stopping.

Partial Ball Thread Code

```
class Ball implements Runnable
{
Thread myThread;          // this ball's thread
double x, y;              // this ball's coordinates
String myNumber;          // ball's number as a string
boolean suspended;        // whether thread is suspended

public void run()
    {
    myThread = new Thread(this);   // make thread
    while( true )
      {
      move();                      // move the ball
      myThread.sleep(delayMs);     // sleep
      }
    }
...
}
```

Cautions about Threads

- Reasoning about concurrent programs is inherently more difficult than reasoning about sequential ones.
- They can exhibit **non-deterministic** behavior, when variables are shared among threads.

Non-Determinism

Suppose x == 1 initially.