Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-seven)

7.4: Multithreaded Programs

How would you write an arcade-style game program in Java? In a typical video game, dozens of little green aliens and spaceships fly around the screen independently. Each object on the screen is spinning, wiggling, or otherwise changing; and each object is changing position according to some gravitational equation or artificial intelligence algorithm.

Or let's say you want to write a Web browser in Java, like the HotJava browser. A web browser has to do many things at once: download data, parse and format HTML (hypertext markup language), process the user's mouse clicks and key presses, and draw animated hourglasses or spinning planets to let the user know time is passing. How would you write a program that seemingly can do ten things at once?

Java makes this kind of *multiprocessing* easy. In Java, to create an independently executing subprocess within an application, you merely need to create an object of type Thread, handing the Thread an object that describes the work to be done, and call the start( ) method on the Thread object:

```
class Spaceship implements Runnable
{
    // Other Spaceship methods
    public void run()
    {
       while (true)
       {
          move();
          animate();
       }
    }
}

...

// create two spaceships and set them going

Thread t1 = new Thread(new SpaceShip()); t1.start();
Thread t2 = new Thread(new SpaceShip()); t2.start();
```

The Runnable interface contains one method, run( ), as implemented in class Spaceship. The Thread class's start( ) method does whatever operating-system-dependent magic is needed to start a new thread of execution (called a light-weight process, LWP, on some systems). The end result of the code above is that two loops run simultaneously in the program: two different Spaceships are alternately move( )ing and animate( )ing. Each thread will run forever, or at least until run( ) returns or we call the Thread stop( ) method. It really is that easy!

By the way, instead of creating a separate Runnable class, you could also make Spaceship extend Thread, and supply the same run( ) method:

```
class Spaceship extends Thread
{
    // Other Spaceship methods
    public void run()
    {
       while (true)
       {
          move();
          animate();
       }
    }
}

...

// create two spaceships and set them going

Spaceship s1 = new SpaceShip(); s1.start();
Spaceship s2 = new SpaceShip(); s2.start();
```

This works too, but it's not as flexible as the first version using Runnable. If a Spaceship inherits from Thread, it could not also inherit from AnimatedObject, for example (an imaginary class supplying some useful animation code). This also violates our precept about not inheriting from concrete classes. It's a classic example of misusing inheritance to mean something other than "is-a."