Timer Class Worksheet

The Basics

The timer class is great for any time you need to have functionality which triggers at regular amounts of time (or irregular amounts of time with a little math). It's basically like an egg timer for Flash. The timer class works in milliseconds, not seconds, so you'll need to do a little multiplication to convert it. 1,000 milliseconds is one second.

Basic Example:

```
import flash.utils.Timer;
import flash.events.TimerEvent;

// Goes off every 4 seconds. Setting the time here is optional.
var enemyTimer:Timer = new Timer( 4000 );

enemyTimer.addEventListener(TimerEvent.TIMER, spawnEnemy);
enemyTimer.start();

function spawnEnemy(event:TimerEvent):void
{
   trace("move the enemy!");
}
```

Example With Random Delay:

```
import flash.utils.Timer;
import flash.events.TimerEvent;

var enemyTimer:Timer = new Timer();
enemyTimer.addEventListener(TimerEvent.TIMER, spawnEnemy);
enemyTimer.start()

function spawnEnemy(event:TimerEvent):void
{
       trace("move the enemy!");

       // Changes the next delay to a time between 2 and 6 seconds
       enemyTimer.delay = 2000 + Math.random() * 4000;
}
```

Timer Class Worksheet (continued)

The Basics (Continued)

Example Which Only Goes Off 3 Times:

```
import flash.utils.Timer;
import flash.events.TimerEvent;

var enemyTimer:Timer = new Timer();
enemyTimer.addEventListener(TimerEvent.TIMER, spawnEnemy);
enemyTimer.repeatCount = 3;
enemyTimer.delay = 4000;
enemyTimer.start()

function spawnEnemy(event:TimerEvent):void
{
	trace("move the enemy!");
}
```

NOTE: You could set the repeatCount and delay when you create the timer like this:

```
var enemyTimer:Timer = new Timer(4000, 3); // 4 second delay which fires 3
times before calling it quits.
```

Functions of the Timer Class

start()	Makes the timer start running.
stop()	Stops the timer if it is running.
reset()	Stops the timer, if it is running, and sets the currentCount property back to 0, like the reset button of a stopwatch.

Properties of the Timer Class

delay	**(Number)**	The number of milliseconds before the timer is triggered.
repeatCount	**(int)**	The number of times the timer will go off. A value of 0 will go on forever, or until you tell your timer to stop.
currentCount	**(int)**	The number of times the timer has gone off.
running	**(Boolean)**	True if the timer is currently running, false if it's not.