## Class SimpleTimer

java.lang.Object

public class **SimpleTimer**extends java.lang.Object

A simple timer class that allows you to keep track of how much time has passed between events. You use this class by creating a timer as a member field in your actor (or whatever):

private SimpleTimer timer = new SimpleTimer();

Then when you want to start the timer (for example, when a shot is fired), you call the mark() method:

timer.mark();

Thereafter, you can use the millisElapsed() method to find out how long it's been since mark() was called (in milliseconds, i.e. thousandths of a second). So if you want to only allow the player to fire a shot every second, you could write:

if (timer.millisElapsed() > 1000 && Greenfoot.isKeyDown("space"))  
 {  
 // Code here for firing a new shot  
 timer.mark(); // Reset the timer  
 }

**Version:** 1.0 **Author:** Neil Brown

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**SimpleTimer**](http://docs.google.com/SimpleTimer.html#SimpleTimer())() |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**mark**](http://docs.google.com/SimpleTimer.html#mark())()            Marks the current time. |
| int | [**millisElapsed**](http://docs.google.com/SimpleTimer.html#millisElapsed())()            Returns the amount of milliseconds that have elapsed since mark() was last called. |

|  |
| --- |
| **Methods inherited from class** |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### SimpleTimer

public **SimpleTimer**()

|  |
| --- |
| **Method Detail** |

### mark

public void **mark**()

Marks the current time. You can then in future call millisElapsed() to find out the elapsed milliseconds since this mark() call was made. A second mark() call will reset the mark, and millisElapsed() will start increasing from zero again.

### millisElapsed

public int **millisElapsed**()

Returns the amount of milliseconds that have elapsed since mark() was last called. This timer runs irrespective of Greenfoot's act() cycle, so if you call it many times during the same Greenfoot frame, you may well get different answers.