Learning Activity: Timer

Back

In this lab, you will create a simple timer application.

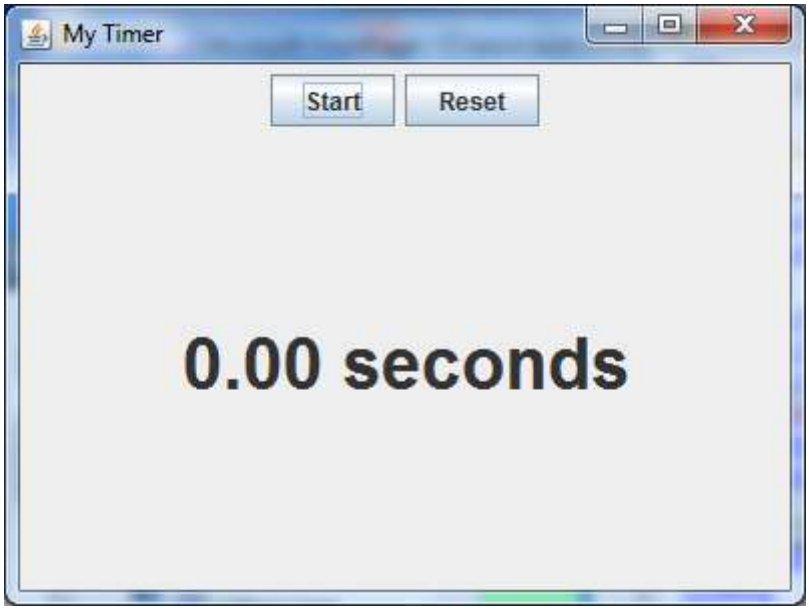
Educational Objectives

- Practice using events
- Practice using threads
  So, if you other (non-thread) timing technologies, you will not have completed the goal of this LA.

Instructions

Initial configuration

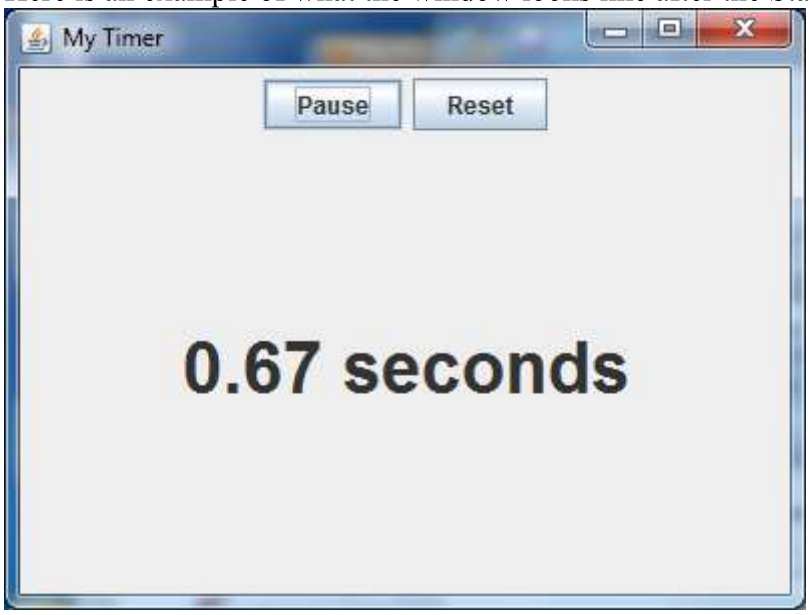
- Create a Java application named **csc143.timer.Timer**.
- The application shall create a window with two buttons and a label.
- The first button shall be labeled "Start".
- The second buttons shall be labeled "Reset".
- The label shall display the elapsed time, in 0.01 second increments.
- The text in the label should be centered vertically and horizontally.
- The label shall be easy to read. This example uses 36-point, bold Arial as the font.
  (Hint: see java.awt.Font)
- The initial window looks like this:



Clicking the Start button

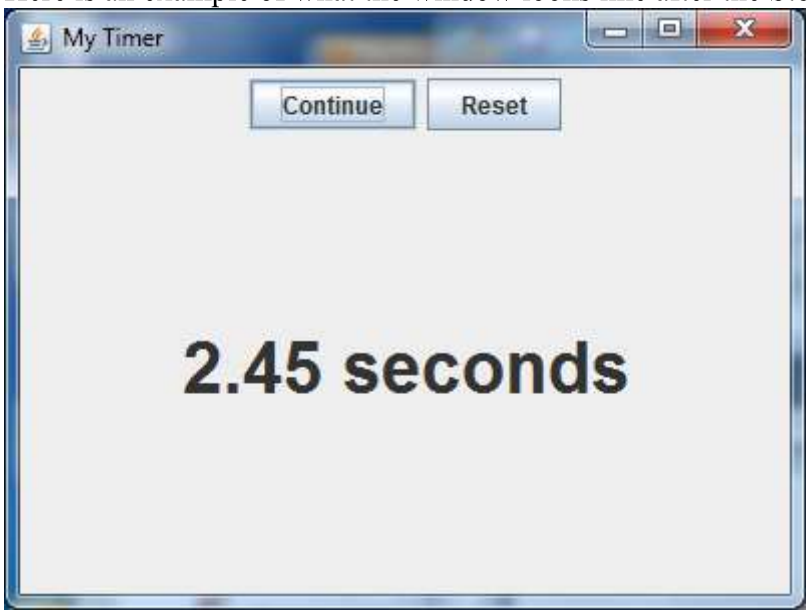
- Clicking the Start button, starts the timer.
- The button caption changes to "Pause".
- The label updates each 0.01 of a second to reflect elapsed time.

- Here is an example of what the window looks like after the Start button is clicked:



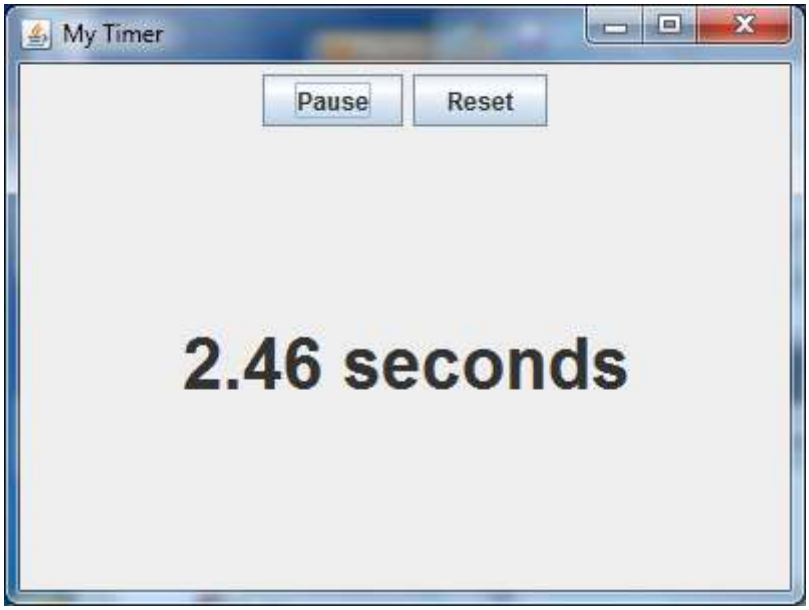
Clicking the Pause button

- Clicking the Pause button, stops the timer.
- The button caption changes to "Continue".
- The output label stops counting elapsed time.
- Here is an example of what the window looks like after the Stop button is clicked:



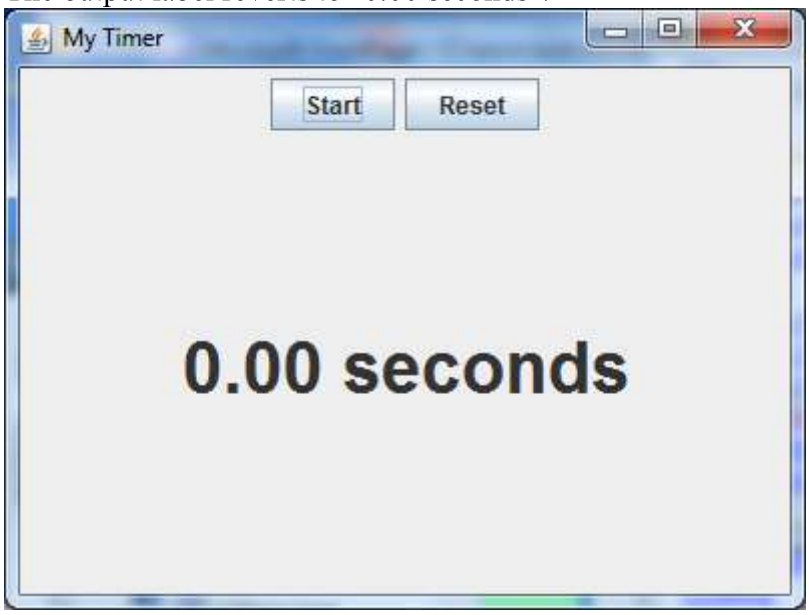
Clicking the Continue Button

- Clicking the Continue button, restarts the timer.
- The button caption changes to "Pause".
- The output label resumes counting elapsed time, *adding* to the elapsed time.
  That is, the next update will indicate 0.01 seconds after the time displayed during pause.



Clicking the Reset Button

- Clicking the Reset button, returns the timer to the initial configuration.
- If the timer is running, it is stopped.
- The caption of the first button changes to "Start".
- The output label reverts to "0.00 seconds".



- (optional) This implementation also sets focus to the Start button.

Notes

- Use java.lang.Thread rather than java.util.Timer to implement this lab.
- You may assume that the Thread.sleep is accurate enough for this implementation. That is, you can assume that every call the Thread.sleep(10) is 10 milliseconds long. So, after calling Thread.sleep(10), the display can be updated by 0.01 seconds.

Submission

- Runnable jar file for the Timer, including source code.
  Source code should be compilable as extracted from the jar.

Evaluation Criteria

Check Level

- The submitted source code shall compile and run without errors.
- The application shall conform to the written specifications above.
  - Two buttons, with appropriate actions
  - One label with centered, easy to read text displaying elapsed time.
  - Timer application displays elapsed time as described above.
- The source code shall include JavaDoc comments for all public resources.
- There shall be reasonable commenting and formatting within the code.

Plus Level

There is an inherent inaccuracy in the output because Thread.sleep allows the thread to be suspended for longer than the argument time. Continue to use Thread.sleep to control the timing of the update of the output, but see how you can increase the accuracy of the timer. For example, how can you keep the elapsed time displayed accurate, even if the time the thread sleeps were 12 milliseconds, rather than 10.

To test your solution, see what happens if the call is changed to:

```
Thread.sleep(50);
```

Remember to set the call back to 10 milliseconds before submitting the code.

Back