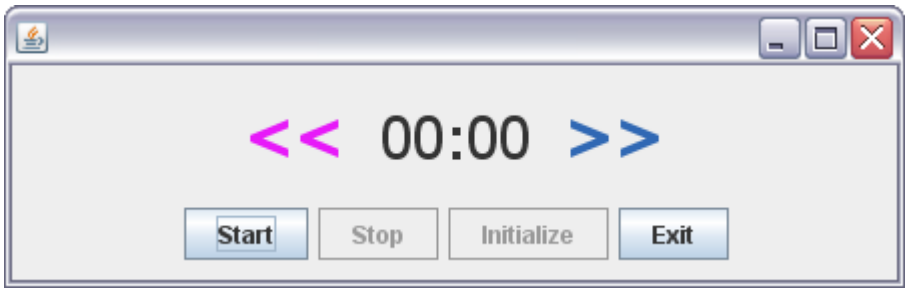
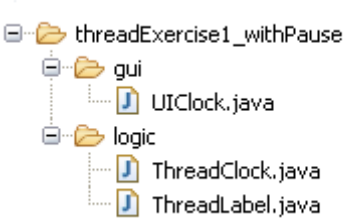
TU2: EXERCISE 2

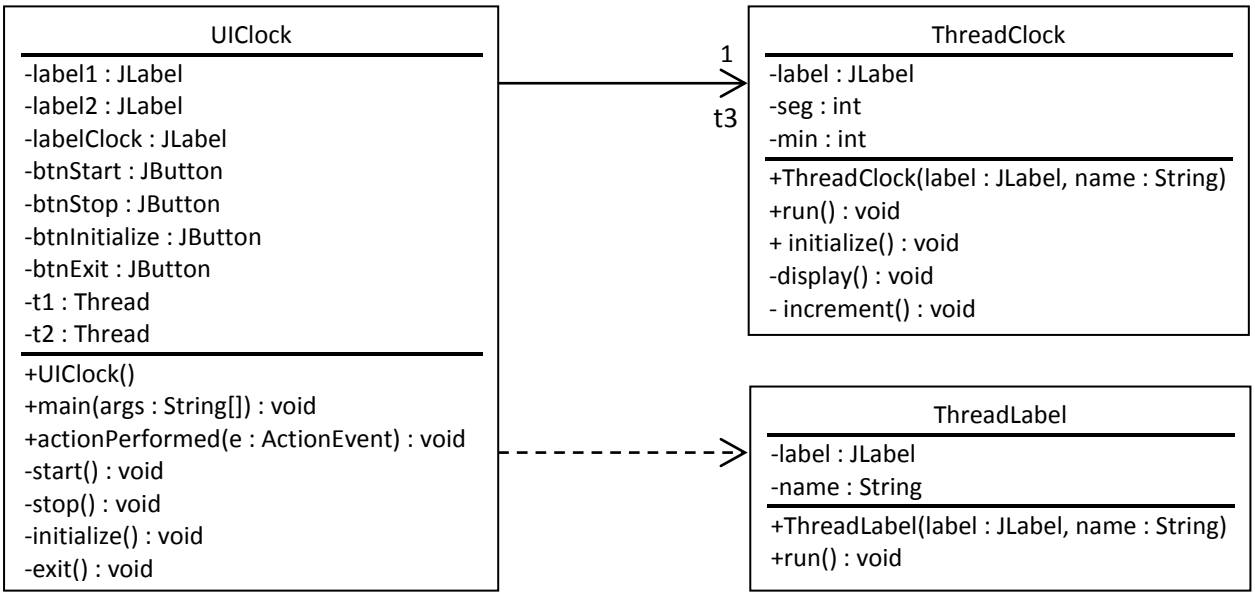
Given the following user interface:



Given the following packages and classes:



Given the following class diagram:



Application's main thread will start 2 threads **ThreadLabel** (implements Runnable type) to manage the labels ('label1' << and 'label2' >>) that change their color randomly every 2 seconds. These threads have the minimum priority assigned. They will be executed infinitely, until the user presses 'Exit' button.

There are 5 buttons, which manage the clock. This clock is just another label, 'labelClock'. This label is managed by the thread **ThreadClock** (extends Thread type). This thread will be executed with a normal priority. Each second the clock must be incremented, so the label's text must be updated. Take into account that if the clock arrives to 60 seconds, the minutes must be incremented, and if minutes and seconds both arrive to 60 the clock must be initialized to zero.

- Start button: starts the thread.

- Stop button: stops the thread and resets the label to 00:00.
- Initialize button: resets the label to 00:00, without interfering with the execution of the thread.
- Exit button: exits the application. All threads must be stopped.

While the clock is stopped, only the buttons 'Start' and 'Exit' will be enabled. While the clock is running, all buttons except 'Start' will be enabled. All threads must have a name assigned and must display a message with their name when they are stopped ('Thread *nameOfThread* has finished').

OPTIONAL

Pause/Resume button: pauses the thread execution. Its name changes to 'Resume' while paused, while running or stopped 'Pause'. While the clock is paused, only buttons 'Resume' and 'Exit' will be enabled.