RIT | Golisano College of Computing and Information Sciences
School of Information

ISTE-121
HW03 - Fun with Timers & Threads

Homework Topics:

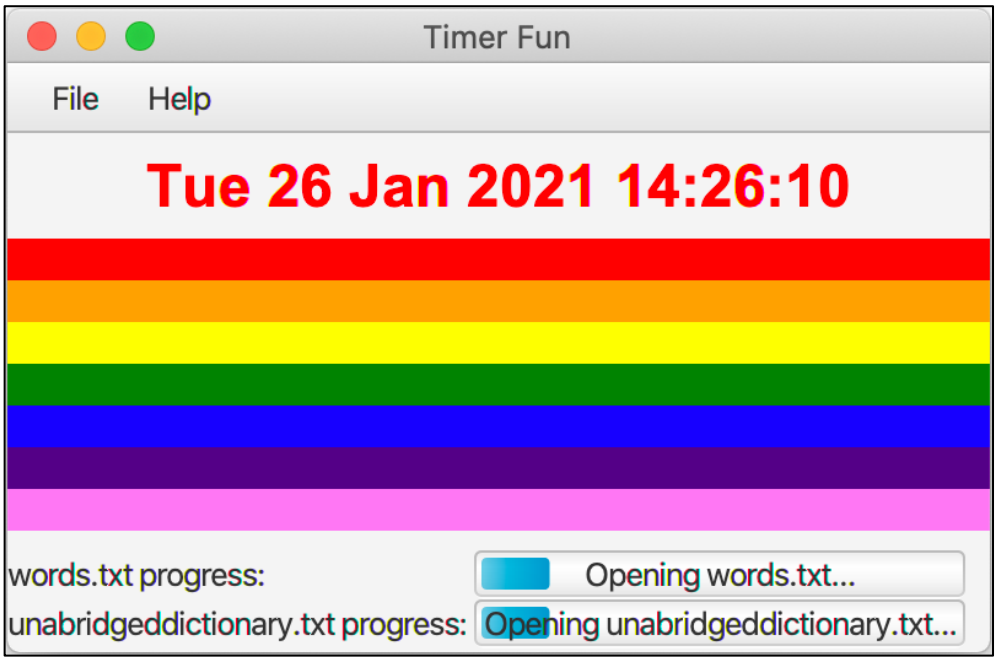
When you have an activity that requires the behavior of threading, you have a choice of which type of threading you want to use. This homework provides practice for two different types of threading behaviors: Threads/Runnable, and the Timer class from the java.util package.

This homework also includes: Changing label fonts via HTML or Java classes, File I/O, Color class, ProgressBar, anonymous inner classes, Date class, SimpleDateFormat class, Thread's sleep() method and possibly arrays.

Problem Statement:

Duplicate the following display and behavior using the two different threading techniques specified in this assignment.

When the program is started it presents (1) the date/time in format shown, (2) a rainbow of colors (ROYGBIV), and (3) at least two progress bars at the bottom of the screen. The different behaviors are controlled by three somewhat different threading controls as specified for each.



Requirements / Specifications:

This is an explanation of how the program is composed:

The three parts are as follows:

1) Date/Time - This starts to display as soon as the program starts.
    a. Current time is updated every second, and does not turn off under any circumstances.
    b. Use the **java.util.Timer** class.
    c. The clock's font and size must be different than the default Label. This can be accomplished by using the Java Font class or HTML tags, your choice.

2) The Rainbow of colors - displays, then rotates colors starting 2 seconds after the program starts.
    a. Colors follow ROYGBIV (Red, Orange, Yellow, Green, Blue, Indigo, Violet), in that order from top to bottom as shown. The colors are Color class constants. Consider:

```
Color[] color = { Color.RED, Color.ORANGE, Color.YELLOW,
   Color.GREEN, Color.BLUE, Color.INDIGO, Color.VIOLET };
```

    I used a Label for each bar, and set its background to the correct color:

```
public static final int NUM_BARS = 7;
Label[] lblColor = new Label[NUM_BARS];
```

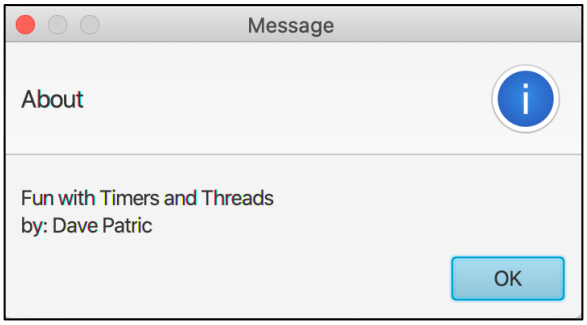
    To set the background, use:

```
lblColor[i] = new Label("");
lblColor[i].setBackground(
   new Background(
      new BackgroundFill(color[i], null, null)));
lblColor[i].setPrefWidth(350);
```

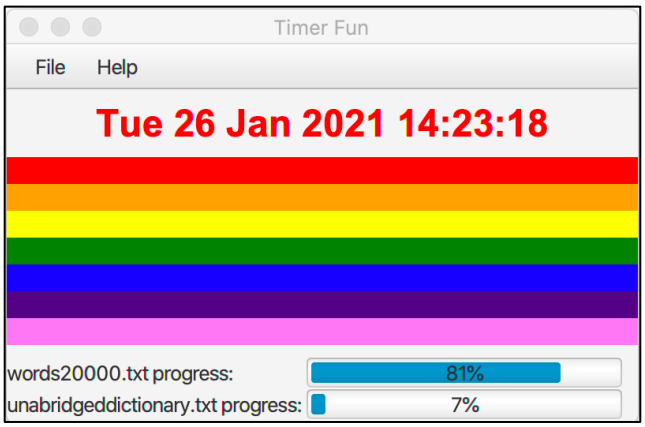
    These seven colors will be showing at all times, in cyclical order.
    b. The colors rotate up every half-second (½ second). This means Red moves to the bottom of the rainbow, and orange is at the top as in OYGBIVR.
    c. Before the cycle starts, wait 2 seconds after the program starts, then start the rainbow cycle moving.
    d. Since there is a wait before movement, then the cycle starts, use the **java.util.Timer** and **TimerTask** classes for the rainbow movement.
    e. Hints:
        i. These are seven Labels (in an array).
        ii. The colors move up. The top color is set to the color below it, etc.

3) Progress bars - reading the files
    a. Several files have been supplied in HW4Downloads.zip.
        i. A words.txt file of 10,000 words and an UnabridgedDictionary.txt of 213,558 words are supplied in HW4Downloads, among others. Both files are to be read in, and the progress bars are advanced to match the progress of reading in the file.
    b. Since this is a complex task, create ONE inner class of either Thread or Runnable, from which you will create the multiple reading threads and progress bar updating.
    c. You must program this as if there were more than two files. Should your program be given 12 files, it would show 12 progress bars, and all would interact similar to as if there were only two files. To add a new file, minimal additional code would be required.
    d. Upon program startup set the progress bar
        i. To display "Opening file…".
        ii. Wait 2-seconds, and then start reading the files. Sleep is OK here.
    e. You must use BufferedReader or Scanner to read the text files one **line** at a time, not one character at a time.
        i. Concatenate the words to a String variable to slow down your Thread. Do NOT use a StringBuffer/StringBuilder as it will be so fast we won't see any progress bar movement. Computers with high memory may also speed up the progress bar, possibly too fast.
    f. Progress bars are to show percentage increases, from 0% to 100%.
    g. The progress bar of the file that finished reading first:
        i. Will show 100%, then wait for a 2-second pause.
        ii. Each other file should show how much of the file was read when the first file finishes.
        iii. For all progress bars, return to the indeterminate scanning back and forth until the program exits. Screen shots on next page.
        iv. Halt the color bar cycling. Clock is only component still changing.
    h. Hints:
        i. Parameters for inner class constructor should be filename and progress bar object.
        ii. To determine progress, use the byte size of the file compared to the total length of strings read.
        iii. If reading stops at 83% or 91% you are missing a concept about reading and counting characters in text files.

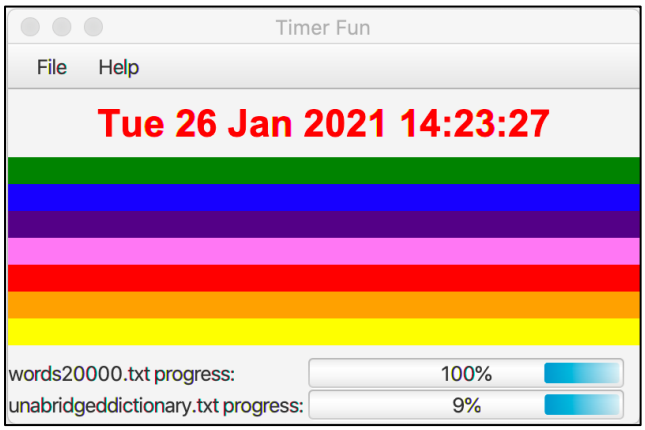
4) Menu's
    a. File → Exit
        i. File has one item under it, Exit. This along with the red [X] has a 2-second wait, then the program exits. The code to wait and System.exit(0) should use the same code for both Exit and [X].
    b. Help → About
        i. The About menu item can be run at any time and as many times as desired during the file reading. The About message could be anything, but should at least look something like this.



5) Overall information:
    a. The program name is TimerFun.
    b. Program this to allow these two files, or several files read in at one time, adding minimal code for progress bars and labels as needed.
    c. Size of display is 350x350, or close to this.
    d. The screen shot on page 1 is actual size, the two displays below are reduced in order to fit in this document.



During execution.



After words20000.txt finished reading. In this example Unabridged.txt only was able to reach 9% before it was halted.
- Color bars are now stopped.
- Clock seconds still advance.

Other requirements:
- Program to run must be called **TimerFun.**
- Zip all ***.java** and all text data files used. Zip the files, **not** the folder containing the files.
- Submit your zip file to the HW03 Assignment folder by the due date.

ISTE-121 HW03 Gradesheet

Timer Fun

Item	Possible points	Earned points
GUI Appearance:		
Clock: (10%)		
Clock at top in different font and size than normal Label, HTML/font-color	4	
Clock shows time in specified format (ex: Tue, 26 Jan 2021 14:23:27)	4	
Clock smoothly changes time every second	2	
Rainbow: (10%)		
Rainbow of 7 colors, ROYGBIV, appear in the middle	3	
Rainbow smoothly rotates colors up the GUI, colors don't skip	4	
Rainbow moves every ½ second	3	
File reading: (25%)		
Two labeled progress bars, show "indeterminate" on startup, with "Opening Words", and "Opening Unabridged" (or similar wording)	5	
Progress bar uses 0%-100%	4	
Word changes ##% faster than Unabridged ##%	4	
Word progress bar stops at 100% for two seconds, before changing	4	
At the same time:		
Progress bars stop with % of file read displayed	3	
Rainbow colors stop progressing, possibly after a delay	5	
Help → About works while *File reading* is advancing the progress bars	5	
Red [X], and File → Exit, have a 2 second wait before exiting program. Same code is used.	5	
Code:		
Clock: (8%)		
Uses java.util.Timer class	10	
Rainbow: (13%)		
Uses java.util.Timer class	5	
Colors are defined using Color	3	
Waits 2 seconds before starting cycling colors	2	
Code is simple, showing programmer understands java.util.Timer	6	

Reading files: Uses ONE inner class of either Thread or Runnable class to create objects One per file being read, which directly correlates to the progress bar. Read file with BufferedReader or Scanner, a line at a time (one char at a time is invalid) Concatenate the words to a String variable (no StringBuffer/StringBuilder) Adjusts read in length of line, so it does not stop at 83% or 91% (not -2)	 5 4 3 3	
Other items: Program named: TimerFun Overall layout of code, logic, and documentation • No, or minimal, duplication of code. • Private methods used where needed to better decompose the code • Documentation of file, class, methods, and functional code segments	 4 5	
Deductions: Late deduction: Attributes were not set private Coding standard violations, etc. Submitted something other than a zip file, or not all code supplied (-5) Other violations:		
Total:	100	

Comments: