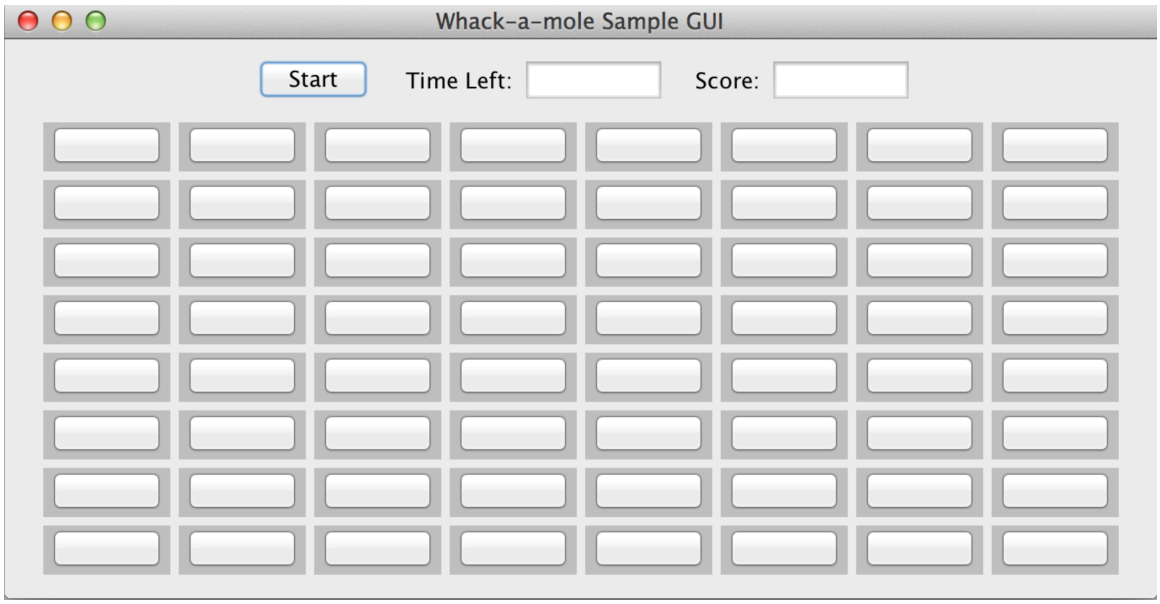
Homework 7: Whack-a-mole Game

Due date: February 25, 2019 (Mon)

For this homework, you will build a Whack-a-mole game using Swing. Your game will allow a player to play the game for 20 seconds before the game stops. When player clicks on a "mole" with its head "up", the player gets a point. Keep in mind that this game would be played by many users at the same time and many games would run concurrently on the same machine.

We are flexible with the layout of your swing GUI, as long as all the components are present and it looks nice. Below is a sketch of the interface we would like you to build. Do not use any tools or plugins to generate the GUI. You must write your own code. Your program must be in a file called `Game.java`.



The GUI which must include a "Start" button, a label and text field for the time, and a label and text field for the score. There must be many buttons, each representing a mole and it's "mole hole". You can choose the number, placement, and content of the buttons, as long as there are at least 10 buttons (representing 10 holes, each hole containing a mole).

Specification

Part I – There must be an "up" configuration for buttons indicating that the mole has popped his head out of his hole. Below we discuss that clicking on an "up" button will give the player a point. Similarly, there must be a "down" configuration for buttons to indicate that the mole's head is not popped up. Also, there will be a "hit" configuration that is displayed when an "up" button is clicked on. Clicking on a "down" button (or a "hit") will not give the player a point or another point. At the start of the game, all buttons are in the "down" configuration.

Part II – When the start button is clicked:

(a) Disable the start button so that it cannot be clicked again until the game is over. (Do this in the actionPerformed() method while you are starting up the game.)

(b) create and start a timer thread. (You are not allowed to use java.util.Timer class in Java. You must write your own Timer following recipes provided to you in class.)

The timer thread must display a countdown clock in seconds from 20 to 0 in the time text field. Time text field should not be editable. The timer thread must also re-enable the start button 5 seconds after the game ends. Roughly these are the steps:

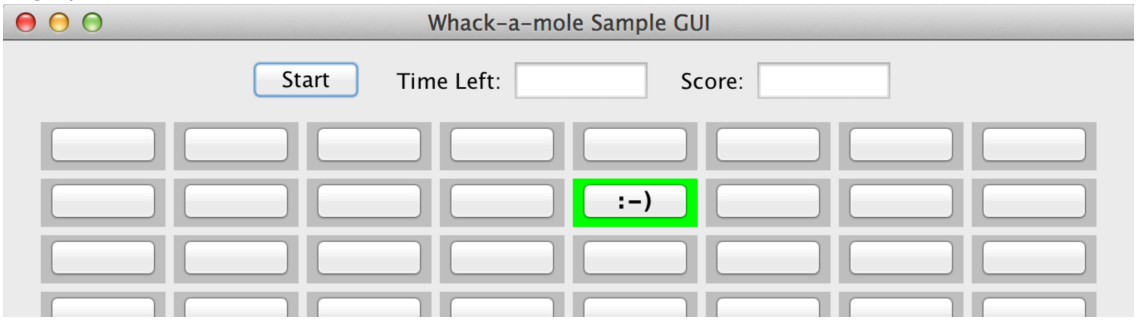
1. Set count to 20
2. Display count
3. While count is greater than zero
    - Sleep for one second
    - Subtract one from count
    - Display count
4. Sleep for five more seconds
5. Enable start button

Part III – Also in the `actionPerformed()` method (after the timer thread is created and started), **create and start one thread for each of the buttons.** Each mole thread should cause its "mole" to pop up by changing the button into the "up" configuration. The mole should pop up for a random amount of time that's more than ½ second and less than 4 seconds. You can fine-tune the "up" time as necessary. (E.g., you could decide ½ second is too short for the minimum up time, so, for example, you might choose to make the "up" time range from 1 to 4 seconds.) Once a mole goes down, the mole must stay down for at least two seconds before popping up again. A mole may not pop up once the timer reaches zero. It's up to you whether to make the all the moles go down when the timer gets to zero. (The Lights program in from Lecture shows an example of generating random integer numbers.)

Important things to keep in mind:

- Only one thread for timer.
- **Each button represents one mole and there must be one mole thread per button.**
- Each mole thread that gets created and started should be in charge of controlling its corresponding button (mole) to pop up and go down. **No other threads should be created and used for this purpose.**

Here's an example of what the mole might look like when it's "up". You can use different colors, text, and/or graphics for the mole buttons.



Part IV – When a mole button is clicked, check to see if the mole is "up" in this mole hole. If an "up" mole button is clicked:

1. Increment the score and display it in the score text field. Score field should not be editable.
2. Change the button to display an indication that the mole has been hit. For example, perhaps the button would then show a frowny face :-(
3. After some delay, the button should return to the "down" configuration. (This can be the remaining time the mole thread would have left the button "up" if were not clicked.)

Note: If the button is clicked multiple times when a particular mole is up, the score should not increment more than once. Also, if the timer is zero, clicking on a button will not increment the score.

Turning in your work

Submit your `Game.java` file itself via AutoLab (https://autolab.andrew.cmu.edu). Note that no zip file needs to be submitted. **No image files are allowed**. Submit just Game.java file only.

Grading

This assignment will primarily be graded by the TAs, after the due date for this assignment. AutoLab will only verify that you have turned in a Java file that compiles and will check your coding style. (Autolab will only show the points for "file compiles", "coding style".)

For this assignment, points will be allocated as follow (total of 100 points):

- File Compiles (on Autolab): 1 point
- Coding Style (on Autolab): 8 points
- @author comment in JavaDoc (on AutoLab): 1 point
- Swing GUI: 30 points
- Actions: 40 points
- Threading: 20 points

After the TA grades your assignment, the total score will be available in the Canvas Gradebook.

Be careful! If you did not follow the specifications provided in this document, there will be heavy point loss even if your program seems to work.