Alice Lab 5

The purpose of this lab is to further study the use of lists, variables, and parameters. We will build a game called "Whack A Mole". The idea of the game is that a random mole pops his head up and then goes back down over and over. The player tries to click on the mole while he is popped up, and if the player manages to click on the mole while it is up, the player's score increases by one. The moles stop moving when the player's score reaches 12.

1. Add a "Whack A Mole Booth" from the Amusement Park gallery into an empty world. Make the bopper and the mole (which are sub-objects of the booth) **not** visible as part of the setup. Position the booth so that it looks something like this:

2. Add 12 moles to the world, putting one into each of the holes in the booth so that they are not visible at the beginning of the program. The copy button should be very useful. In fact, in the picture you see above, there really are 12 moles pushed down in the holes; they are just below being visible. Create a list that belongs to the booth object that contains the 12 moles.
3. Add a method of the booth called "popMole" that uses an object parameter representing which mole is supposed to pop up and down. Inside this method, make the object passed in move up 0.2 meters in 0.25 seconds, then wait 0.3 seconds, then make the object passed in move down 0.2 meters in 0.25 seconds. Test this method by calling it from myFirstMethod and passing it any of the moles you wish. You may have to play with the distance the parameter moves up and down, depending on how deep you placed the moles.
4. Add a variable to the booth object that will keep track of the player's score. Give it an initial value of zero. Then in myFirstMethod, add a while loop that continues as long as that score is less than 12. Inside the loop, call the popMole method of the booth. Drag in the list of moles as the value of the argument needed. You will see an option appear to allow you to use a random element from the list. Choose that option. When you run the game, you should see random moles popping up and down. If necessary, make adjustments to the initial positions of the moles or the booth.

5. Add an event that occurs when the player clicks on anything. Don't fill in any of the slots yet. Create another method of the booth called "clicked" that uses an object parameter called "x". Inside this method, add a loop that goes through the list of moles in order. Inside this loop, add an "If" statement that tests to see if the item of the list the loop is currently working on is equal to the argument x. If it is, increment the score and have the argument object play a pop sound. Go back and fill in the slot for the mouse event with the "clicked" method, using as an argument the object under the mouse cursor. Run the program and play the game. You have to be fairly quick to catch the moles. If necessary, slow the moles down so that you can hit them when you click. If you leave the score variable showing on the left side of the screen when you run the program, you can watch the score change as you play.
6. We now want to make a mole turn red and stop popping up when he is clicked. To do this, add a new method to the booth object called "remove" that uses an object parameter for the mole that was clicked. This method needs a local variable to keep track of where in the list you are, so that you will be able to remove a mole from the list when appropriate. Initialize the local variable to zero. Inside the method, add a loop that goes through the list of moles in order. Inside the loop, add an "If" that tests to see if the element of the list that you are currently working on is equal to the value of the argument. If they are equal, remove from the list the element in the position (index) indicated by the local variable. Be sure to increment the local variable inside the loop. Go back to the "clicked" method of the booth and inside the "If" that is inside the loop, add a statement that makes the parameter turn red in zero seconds and then removes it from the list. Now when you run the game, you should only be able to hit a mole once. You may find that occasionally you will hit a mole, it will turn red, but it will pop up once or twice more. That can happen because of concurrent timing issues. You should still only be able to hit it once.
7. Turn in your completed program on the shared drive by Wednesday.