Lab 8 – Nim
CSCI 1101B – Spring 2015
Due: April 28, 10 pm

Objective: To gain experience using arrays and for loops.

In this lab, you will write a program to play Nim, a simple game that involves two players taking turns removing stones from one (or several) piles of stones. While the exact rules of the game have many variations, the version you will implement starts with a pile of 10-20 stones (randomly selected), and each player takes turns removing either 1 or 2 stones (but no other amount) from the top of the pile. The player who removes the last stone from the pile wins the game.

Requirements

Write a program that allows the user to play a game of Nim against the computer.

The provided `Events` class in the skeleton project creates a new game every time the mouse enters the window. Figure 1 shows an example of the initial game window. When the mouse exits the window, the canvas is cleared and, upon reentry, another game appears. Note that the `Events` program checks, when the user presses the mouse, to see if they pressed on any of the stones (that's what the `contains` method does), in which case the `executeRound` method is called, which executes a user move followed by a computer move. The `executeRound` and `contains` methods (and the constructor) are described below.

Note that there is a message that says "Your move! Click a stone to remove that stone and all above it." This Text should be used to provide helpful messages throughout the game, e.g. if the user tries to remove too many stones, it should tell them that. Messages are also described below.

Constructor

The `Nim` constructor should create the initial message, generate a random number of stones in the appropriate range, and create the pile of numbered stones. The pile of stones must be implemented as an array of `FramedOval` objects and the stone numbers must be an array of `Text` objects.

ExecuteRound Method

The `executeRound` method should make the specified user-initiated move, followed by a computer move (assuming the user's move didn't end the game).

In the `Events` class, the `contains` method of the `Nim` class is called to determine whether the user clicked on a visible stone, and the `executeRound` method is called only if that is the case. However, the `executeRound` method also needs to know where the user clicked so you can determine *which* stone they clicked in.

Knowing which stone was clicked in allows you to calculate how many stones the user wants to remove, i.e. all the visible stones from the one they clicked in on up. If the user tries to remove too many stones, change the text message to tell them they have tried to remove too many stones and do nothing else. Otherwise, remove the stones by hiding them.

If the user removes the last stone, use the text message to declare them the winner. Otherwise, execute a move for the computer. It does not need to be an "intelligent" move, but simply a random, legal move (i.e. 1 or 2 stones, but not exceeding the number of stones remaining). Remove these stones. If there are stones left, use the text message to inform the user how many stones the computer removed and tell them it's their turn. If there are no more stones left, declare the computer the winner.

Contains Method

The `contains` method returns `true` if the `Location` object passed in to it is inside a visible stone; otherwise, `false`. A useful method here is the `isHidden` method that can be called on a `FramedOval` and returns `true` if that `FramedOval` is hidden; `false` otherwise.

Submitting Your Work

Submit your project on Blackboard in the usual way. Don't forget to name your project correctly, fill in documentation, and double check the logic and organization of your code.

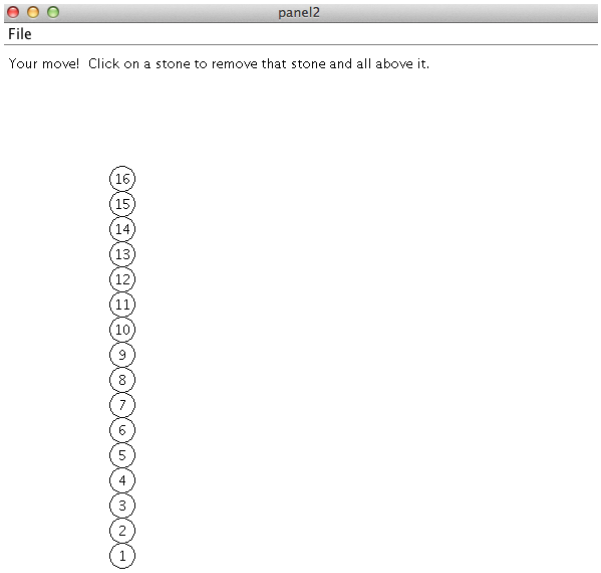

Figure 1: An initial Nim game with 16 stones.