

# LCS-ES Group Meeting #9

## Exercise 1: Drawing 202

This week's exercise is similar in concept to the Car Class exercise from a few weeks ago. However, instead of each person playing the role of a variable as we run through the code, each person will play the role of a method. This exercise is designed to reinforce your understanding of how values are passed and returned through method calls.

One person will play the role of each of the methods `writeText()`, `drawLine()`, `drawOval()`, `drawTriangle()`, `drawRectangle()`, and `knockOnDoor()`. One person will also play the role of the `trickOrTreater` instance of the class `Halloween`. This person will be responsible for the methods `setColor()`, `setTrickVsTreat()`, `getHousesLeft()`, and `getTrickVsTreat()`. Your peer leaders will be playing the role of the main method, as well as returning values for each of the `Math.random()` calls.

**Part (a):** Before we begin, take some time and look over the structure of the code. Pay attention to where methods are called from and where the parameter values come from. Also fill in the missing lines of code (the `Math.random()` calls).

**Part (b):** Now let's run through the code. Each person will be given a copy of their method and whatever other information their method has access to. A new sheet of graph paper will be given for each call to the method `knockOnDoor()`, which will be used for all of the draw methods.

**Part (c):** Having completed the exercise, are there any improvements to the code that could be implemented? What methods would you add to improve the clarity and structure of the code? What could you remove without changing the function of the code? What could you simplify?

## Exercise 2: Tic Tac Toe

**Part (a):** If you have never played tic-tac-toe, play a game or two with a friend who knows how to play.

**Part (b):** The Blackboard site contains a partial implementation of a tic-tac-toe game that uses the Model-View-Controller pattern. With a Java applet the controller is built into Java, so we do not have to worry about it. The view is provided by class `ttt` from the Blackboard site (while this code uses some library methods you may not have seen, you should feel comfortable reading the code). Finally, the model is where that actual data is kept and is the part of the code you will be working on.

Your model must handle two messages:

- 1) from the controller: `boolean playerPickedSquare(int square)`, and
- 2) from the view: `char markInThisSquare(int square)`.

The method `playerPickedSquare`, is called when the current player (the model must keep track of whose turn it is) selects a given square. The squares are numbered

```
0  1  2
3  4  5
6  7  8
```

This method updates the contents, of the chosen square with the correct symbol and then checks to see if there is a winner (or if it's a cats game). Assuming the game is not over, the current turn is changed to the other player. The method `playerPickedSquare` returns true if the game is over.

The method `markInThisSquare` returns the current mark in the given square, which is a blank, an X, or an O.

Working in pairs, download and then complete the code. Then see if you can beat your partner at a game of tic-tac-toe :)

