# CSE 223 Programming Assignment #4 - DOTS!

May 20, 2020

## Introduction

For this project, you're going to write a computer version of the game of Dots (if you're not familiar with dots, see the description below). You will write this in Java, using the Swing package. I encourage you to work in Eclipse, as this is how we are going over this in class. Whatever environment you develop in, **your code must work as a standalone set of classes**.

The program will allow two people to play dots on the computer. They will take turns using the mouse to select lines to draw The program will handle drawing the lines, completing boxes, adding initials, keeping track of whose turn it is, keeping track of the score, and knowing when the game is over and what the outcome was.

(For PA5, we may extend this game to allow two people to play across a network connection...)

## DOTS

The game of Dots is played with two people. You being by making a 2D grid of dots (typically 8x8) on paper. You then take turns connecting pairs of dots by drawing a line (across or up/down, not diagonally). If your line completes a box, then:

1. you put your initial in the box; and
2. you get to go again.

Otherwise, it is the other player's turn. Play continues until all boxes are completed. Whoever has the most initials on the board wins.

## Group Project

PA4 may be done alone, or in a a group of 2 or 3 people. The choice of who to work with is up to you, but no group may be larger than 3 people. For group work, I strongly recommend each group member work on the entire project, and use the group aspect to help each other with bugs and issues, working out how to implement things, comparing approaches, etc. In other words, don't say "Person A does the graphics, person B does the gameplay": each person is responsible for knowing how to do the entire assignment.

If working in a group, either have only one person make their GITLab repo available to me, or make sure all repos are consistent. I will grade the first repo I encounter, and if it's not the latest version of your group's code, you may lose points needlessly.

Make sure every group member's name is included in the top of every .java file. This is necessary to receive credit.

## Requirements

Your program should begin by drawing a grid of dots. Your board should be 9 dots x 9 dots (8x8 boxes). Each box should be 50x50 pixels. These should not be hardwired though: make them variables so that they can be easily changed.

The game should allow two players to enter their names in text fields, and to hit a START button to begin play. The game should keep track of whose turn it is, and display that in some way. Clicking the mouse inside the grid area should draw the closest line (if this line is already drawn report this to the user and let them try again).

The game should notice if a box (or more than one box) has been completed, and if so, it should add the current user's initial to the box(es), and leave it as that player's turn; otherwise, play should switch to the other player.

Each player's score should be reported and updated in a graphical element (a JLabel for example). If all boxes are completed, the game should note this and report who won (or if there was a tie).

There should also be a RESTART button (you can re-use the START button by changing its label) which will completely reset gameplay to the beginning.

## Setup/Suggestions

I have coded this in the following way, which I think makes the task somewhat easier. Other approaches are possible of course, but I believe they will be more complex. Implement this PA as you like, but I strongly suggest the following setup:

### General

It's helpful to think of gameplay differently from the usual procedural programming we've mostly done. This is event-driven code, meaning mostly nothing is happening until an event (a mouse click) occurs. When that event occurs, you can do whatever you like, but then you need to return and let the system go back to waiting for more events.

It's also helpful to remember that you don't get to paint things whenever you like: all painting happens inside your paint() method. All you can do is setup variables to control what paint() will do, and then call repaint to request that paint gets called.

### Ask Questions!

We're using a lot of new ideas in here. I will explain everything in class, but that doesn't mean it's going to make sense! Expect to be confused about some things, and ask questions to help clarify the concepts. It may take several rounds of questions and answers for some of this to click (no pun intended!). Concentrate on one thing at a time: if you understand the pieces, then putting them together will be more natural and make better sense.

### Board

Make a class named Box, which describes a single box in the playing area. Your board is a 2-D array of Boxes. Each Box should have booleans for top, bottom, left, right indicating whether a line exists on that edge of the box. Each Box should also have a char (or some other mechanism) indicating which player owns that box (or a default symbol like ' ' or '-' if the box is not complete).

### Modularize, modularize, modularize...

Make a method for initialize the gameboard. Make a method for updating the score. Make a method for checking whether the game is over. Make a method for checking whether an (x,y) click is a legal move, updating the game board array based on the that click, and adjusting whose turn it is. Make a method for resetting all attributes of the game. Make methods for everything you can think of (within reason). If it's a standalone task, put it in a method. You will likely find that some of your methods call other methods (that's a sign of good modularization). For example, you will want to update the score when the game first starts, or anytime a player clicks the mouse, or when the player asks to restart the game. If you can simply call an "updateScore()" method, your code will be simpler, cleaner, and easier to write and debug.

### Mouse Clicks

When a user clicks the mouse, your mouse listener for the game area should pick up the (x,y) coordinates; decide which Box the use clicked in; decide which side (top, bottom, left or right) they clicked closest to; and then act accordingly (we will go over these steps in class).'

Once deciding which side of which box was selected by the user, the program will see if that line has already been drawn. If it has, report that on the screen and ignore the click (leave it as the same player's turn etc.)

If the line has not already been drawn, set the appropriate boolean in the Box. **Then, also update the boolean in the appropriate adjacent box - this is critical!**

After each click, re-draw the board (call repaint()), update the score, player's turn, game status, etc.

### Drawing the Board

It's important to realize that mouse clicks do not draw on the board: they update your array of boxes, score variables, and so on. All of this is about updating data in memory. Only after all the updates have been made should you call the repaint() method, and **that** is what actually draws the board, lines, initials, etc.

To draw the board, simply loop through the 2-D array, and for each Box, find the (x,y) coordinates of each corner, and then draw the appropriate line based in the Box's top, bottom, left and right flags. Then, if the box belongs to someone, draw their initial near the middle of the box using g.drawString(...)

### Be Methodical

Once you have things set up in a logical and orderly fashion, this code becomes really fun to write. Getting things setup in the right way may take a few tries. Be open to re-thinking how you have setup your code and data structures, and be willing to re-work this if it feels right to do so. You've got to modularize; it will help if you can keep your sections of code as single-purpose as possible, calling other methods to do sub-tasks.

## Grading

This assignment is worth 100 points, allocated as follows:

- .java files can be downloaded and compiled: 15 points
- game starts, draws a field of dots, shows initial score: 15 points
- game allows users to enter their names and uses their initials: 5 points
- game responds in some way to mouse clicks in the field of dots: 15 points
- game reliably draws lines between dots based on where mouse is clicked: 20 points
- game prevents illegal moves (already-drawn lines, clicks outside the field): 5 points
- game adds initials to completed boxes: 10 points
- game correctly tracks whose turn it is: 5 points

- game correctly reports score: 5 points
- restart button allows restart of game: 5 points

## Submission

Your main class (containing your main method) should be named Dots. Upload all your .java files to a GITLab repo named CSE223Dots by the due date. **Use the default package** and test your code outside of Eclipse, by compiling on the command line and then running with the "java" command. If you do this on the Linux server, it should compile fine, but you will get an error message when you try to run which begins:

```
java.awt.HeadlessException:
No X11 DISPLAY variable was set, but this program performed an operation which requires it.
```

That tells you that things compiled fine, which is a good sign. If you have an X server, you can run your code on the linux server and push X11 commands to your own system (this is slow but not a bad check).

Alternatively, if you push your repo early and ask me (more than 48 hours before the deadline), I will be happy to do a trial run of your code and let you know if it is downloadable and runnable (I won't pre-grade it though: this is just a check to make sure I can run your program).