CSE 223 Programming Assignment #5 - NetDot!!!

May 28, 2020

Introduction

For this project, you're going to extend your Dots program to allow two people to play the game across a network. Again, 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.

All the description etc, from PA4 applies here as far as basic gameplay and so on.

This is also a group project; you can work in the same group, a different group, or alone. Same 3-person groupsize limit as before.

Requirements

Some aspects of PA5 have more-specific requirements than PA4. These requirements are to help you use the approach we'll be discussing in class.

- Your program should include a pair of RadioButtons for choosing whether you want to be the server or the client. One player must be the server, and the other the client. The server is considered Player 1 and gets to go first; the client is Player 2 and goes second.

Depending on which button is selected, your main GUI will appear/behave as follows:

- For the Server:
  - the START button creates (a thread to create) a ServerSocket to accept connections on port 1234
  - the Player 2 textfield should be set to non-editable
  - if no name has been entered for Player 1 when START is pressed, set it to "Server"

  - the turn is set to Player 1, meaning this player is allowed to move (once a connection to another player has been made)

- For the Client:
  - a Host ID field should appear, in which the user can enter the address of the machine to connect to (with a default value of localhost)
  - the START button should be re-labeled "CONNECT"
  - The CONNECT button creates (a thread to create) a socket to connect to the Host ID, port 1234
  - the Player 1 textfield should be set to non-editable
  - if no name has been entered for Player 2 when CONNECT is pressed, set it to "Client"
  - the turn is set to Player 1, meaning this player should wait until the other player moves before allowing mouse clicks

Once the START/CONNECT button has been pressed, the button is re-labeled QUIT and can be used to terminate the gameplay. No re-play is required. A connection is good for a single game, after which the users should click QUIT to close the connection.

After clicking START/CONNECT, you should send your name to the other player; receive their name; and update their name in your GUI.

After that, gameplay proceeds as usual, with players taking turns clicking on their own copy of the game board. Lines, boxes, initials, scores, etc. should be updated in both GUIs . Information will be passed between the Client/Server sockets (managed by the communication thread).

You should prevent users from taking a move when it's not their turn.

I will demo sample code in class Friday to show how things should look.

Setup/Suggestions

You should be able to work from your PA4 code, more or less as is, simply adding the GUI changes described above, with mouse clicks conveyed via the network connection.

Threads

All network management should be handled by a separate thread ("netThread"). This is important, since your GUI class should not do any looping (such as waiting for a message to arrive). Construct a thread, call its start() method, and let the thread establish the connection and manage message exchange.

Synchronization

The two programs will move in lock-step. They share a common notion of whose turn it is, which simplifies knowing when to send and when to receive:

- if it is your turn, you will eventually click the mouse, which will bring you into your mouse listener. Confirm that it is your turn, update your data structures and call repaint() as in PA4; but then, send a message to the other player indicating the XY coordinates of the mouse click (have a method in your netThread to send that information).
- if it's the other player's turn, don't allow mouse clicks; but your net-Thread should receive a message containing the XY coordinates of the other player's click: treat that (x,y) like you do in PA4: as a click by the other player.

Everything else should work the same as in PA4: scoring, deciding whose turn it is, etc. Since both programs (Server and Client) see the exact same mouse clicks, they should remain in sync.

Network Protocol

You will pass three types of messages between the Client and the Server:

- player names (in the beginning);
- (x,y) coordinates of mouse clicks; and
- a QUIT message.

You can structure this however you like: but here's a simple protocol that should work well:

- as soon as the connection is established, send a single line containing your name; then read a single line, which you interpret as the other player's name
- send x,y coordinates as a single line such as C x y (where x and y are numbers). Likewise, when you receive a message beginning with a 'C' you should assume it's followed by x y coordinates (separated by spaces). Consider using the split(" ") method of String to break that line into pieces. Integer.parseInt will also be useful here
- A line beginning with 'Q' is considered a quit message.

General

There are a lot of ways to structure this project. The above suggestions are intended to let you reuse most of PA4, adding a bit of protocol to the beginning, and a layer of network I/O during game play. You can do what you like, but if you follow these suggestions, you should be able to play with other people :)

Regardless though, I'll only be testing your code against itself, so it only needs to be consistent with itself.

If you test in Eclipse, get comfortable with managing separate console windows, running two programs, etc. A more-efficient way to test your code during development might be to open windows (terminal windows or DOS shells) in your eclipse workspace/bin, and run your code from there.

Whatever approach you use, it is well worthwhile to develop a streamlined method of running and managing your program instances. I'll show you Friday how I have been doing this. Keep in mind, you could easily end up running your code hundreds of times during the course of your code development.

Grading

This assignment is worth 100 points, allocated as follows:

- .java files can be downloaded and compiled: 15 points
- game starts, draws a GUI, has radio buttons, text fields, etc.: 15 points
- Client/Server radio buttons change GUI as described above: 10 points
- Server START button and Client CONNECT buttons establish a connection: 10 points
- player names exchanged and displayed properly: 10 points
- mouse clicks exchanged successfully and GUI responds appropriately (lines, boxes, initials, score, turn indicator, etc.): 20 points
- game tracks player turns and does not allow out-of-turn mouse clicks: 10 points
- QUIT button available and exits the game on both ends: 10 points

Submission

Your main class (containing your main method) should be named NetDot. Upload all your .java files to a GITLab repo named CSE223NetDot 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.