CS351: Design of Large Programs
Project 5: Networked Scrabble (or Domino) Game

Brooke Chenoweth

Spring 2024

In this project, you are going to build off one of your previous projects, modifying it to be more flexible in the number of players and allowing the players to be on separate computers.

The game server will exist on one machine at a static known address, the client programs will be dynamically created on other machines.

Extending the Scrabble (or Domino) Game

You already wrote a sequential two player human vs computer version of the game. For this project, you will use the same rules, but allow from two to four players in a game, with some players being human and some (or none) being computer players.[1]

Game Server

The game server is static and at a known address. You'll start this program before any of the client programs. *Do not hard code the port number!*

The game server should be able to support many simultaneous client connections and many concurrent Scrabble (or Domino) games.

Aside from possibly some initial configuration, the game server is not expected to interact with the user, though you should have some status messages printed to verify what is happening. If you are feeling fancy, you might even make a JavaFX dashboard to display the status of all the connections/games in one place.

You may assume the game server program will remain running throughout the simulation. You don't have to make the client programs robust to the game server terminating, but the game server should not break when a client program exits normally.

Tables

A game will be played on a table, but not all players will arrive at the same time, so a table needs to exist before the game actually begins. When clients connect to the game server,

[1] For Scrabble, the rules don't change with three or four players, but you may want to increase the number of dominoes for more players to keep the game from ending too early.

they may join an existing table, or configure and start a new table. Once the table is ready, the hosting player can start the game.

Game Play

Game play should follow the same rules as the original project, aside from allowing more flexible in the number of players and human/computer settings for the players.

To keep from waiting forever for distracted human players, if a player does not make a play after 60 seconds[2], they automatically pass and the turn moves on to the next player.

Client Program

Each client is dynamically created. The game server host and port is specified at startup somehow, but it is your choice whether you use command line arguments, a configuration file, or something else. *Do not hard code the host and port number!*

The program may terminate when not actively playing a game. The program should not allow exit when the client is still actively playing. A client terminating should not break the behavior of any other programs in the system.

Client Actions

When the client program joins the game server, they should register with a name to identify themselves.

After joining the server, they should either join an existing table or create and host a new one.

A table host can configure the table with number of players and how many players are client programs or local computer players. Once enough client programs have joined the table, the game may start. (You can have this happen automatically, or allow the host to manually start the game after making sure they have configured the game settings as desired and allowed in the players they wanted to join them.)

The client should be able to play (as appropriate for the game) or pass on their turn, and should be able to resign at any time. You can decide if a resigning client will end the game early, or if they will be replaced by a computer player, just document it in your readme file.

User Interface

The agent user interface may be console-based or may be a graphical JavaFX display. (I expect most of you will use a JavaFX display adapted from the previous project.)

[2]This is long enough to allow time for someone to actually play, but still short enough we don't have to wait forever to make sure the time out feature works. You may want to make the time out delay configurable as part of the game server and/or individual table settings so that you can allow for a more generous amount of time in a real game, especially if the game is Scrabble.

Bear in mind that taking a turn is a time sensitive activity, so however you design your user interface, it should not require typing very long commands and/or navigating overly complicated menus.

The user interface should update as the game proceeds, and be sure to notify the client when it is their turn. Do not allow the client to make a move when it isn't their turn, but they may still have other actions they can take at any time, such as resigning from the game or sending a message to the other players at the table. (Chat functionality is optional, but it might make the game more fun if the players can talk to each other.)

Testing

Testing should involve at least two concurrent games with different configurations of human and/or computer players. The game server will be started first. Make no assumptions to the order the client players join and play.

We will be testing your programs on multiple lab machines in B146, so you should make sure your programs work properly there.