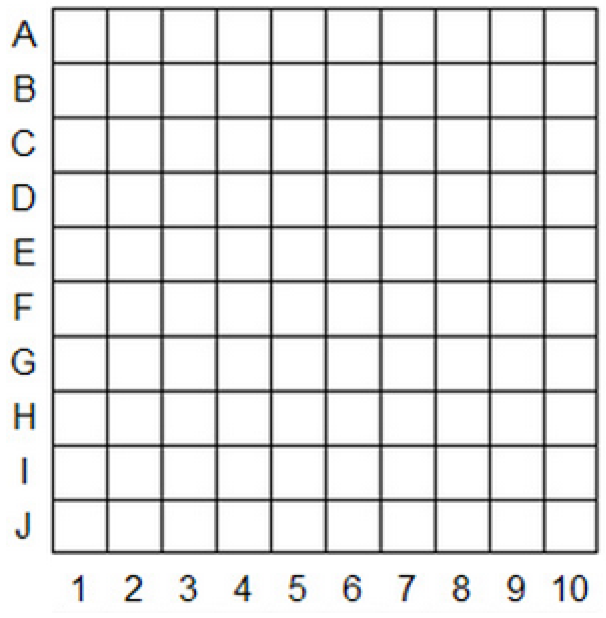
CSCI 201L Assignment #5
Spring 2015
6.0% of course grade

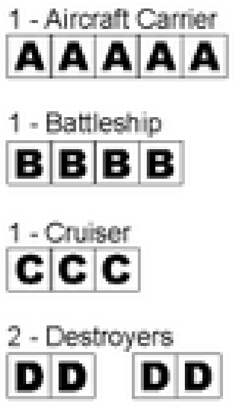
Networking: An exposure to Java Networking with Battleship

Overview
This assignment will implement networking into your Battleship game. All of the core features in the game are finished – you will now provide the ability for a player to play against the computer or against another player. You will allow for the player to connect to a friend, or let a friend connect to them. Once two players are connected, they will create a map and press start. The game will play exactly the same, only you must account for cases such as a player quitting part-way through the game.

Introduction
We have provided you a solution to the previous assignment in order to complete this assignment. We highly recommend that you use your own code though – it is an important skill to be able to build off old code and maintain good reusable code. In addition to this, it will be easier since you should already be familiar with the code you built.

Background
Battleship is a two-player game where each player has a 10x10 grid playing field. This field represents a sea in which players place their ships. One aircraft carrier, one battleship, one cruiser, and two destroyers. Each type of ship is of a different size, but all ships must be placed vertically or horizontally – there is no diagonal placement of ships. The players take turns guessing grid locations of their opponent's ships. Once all grid spaces that are occupied by a ship have been guessed, the ship sinks. The first player to sink all of the opponent's ships wins.
NOTE: This is the same background as last assignment – this is for easy reference.





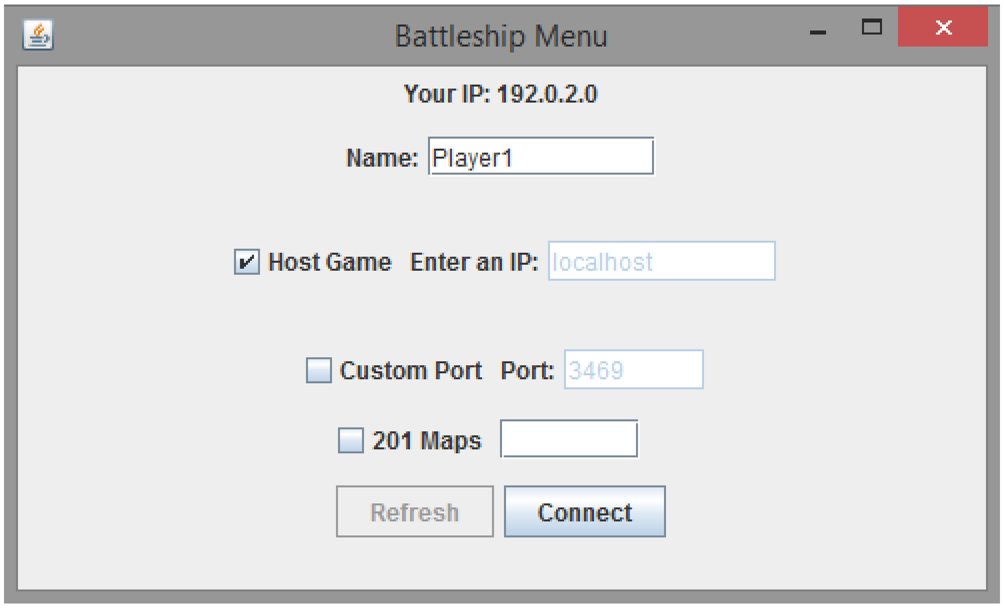
Implementation

Step 1
First, start by adding a start menu where the player can enter the required information in order to play.

You must display the users' own IP address. This way, the user could text their friend the IP so they can play together. You must also allow for the user to choose to host their own game, or allow them to enter their friend's IP. If a user chooses to host, a server will be created on their local machine once they press 'Connect'. As for the port number, provide a port to use by default. However, in case the user is running something else on the port, provide a way for the users to enter a custom port number. You will also need to add an option to select a .battle file from the CSCI 201 website in-case the player wants to play against the computer.

Note: Use '127.0.0.1' or 'localhost' to connect to yourself. If you want to connect to a friend on a different network, your friend will have to open a port through the firewall. You won't be able to do this on USC wireless. You don't have to worry about opening ports if you run two clients on the same machine.

Player 1:


Battleship Menu

Your IP: 192.0.2.0

Name: Player1

[✓] Host Game   Enter an IP: localhost

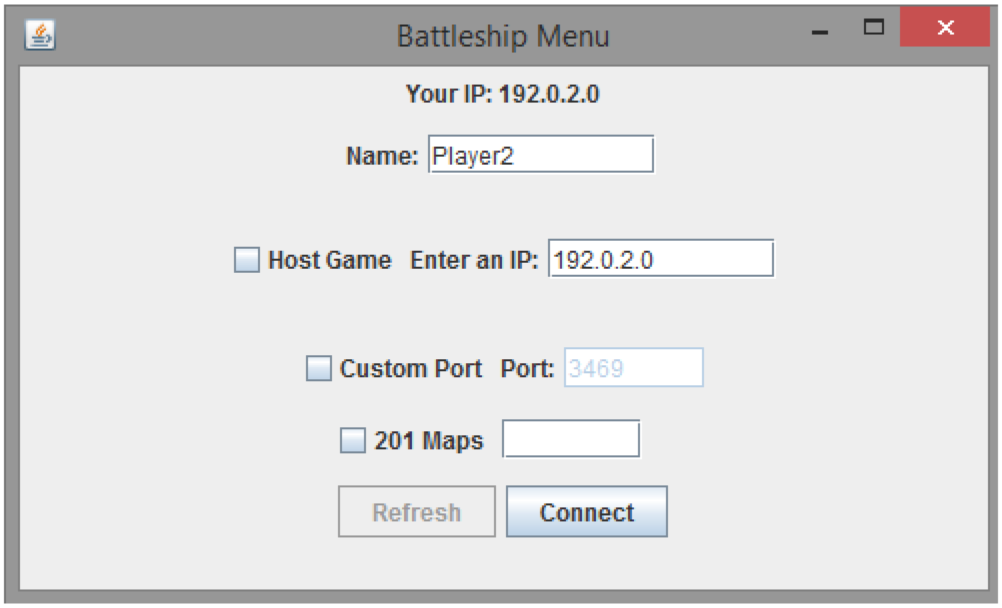
[ ] Custom Port   Port: 3469

[ ] 201 Maps

In order for a friend to connect – they will need a public IP address unless you both are in the same network (in which case a private IP address would work). Use the following code to get your IP address that other players can use to connect to you:

```
URL toCheckIp = new URL("http://checkip.amazonaws.com");
BufferedReader in = new BufferedReader(new InputStreamReader(toCheckIp.openStream()));
String ip = in.readLine();
System.out.println(ip);
```

Player 2:



Your IP: 192.0.2.0

Name: Player2

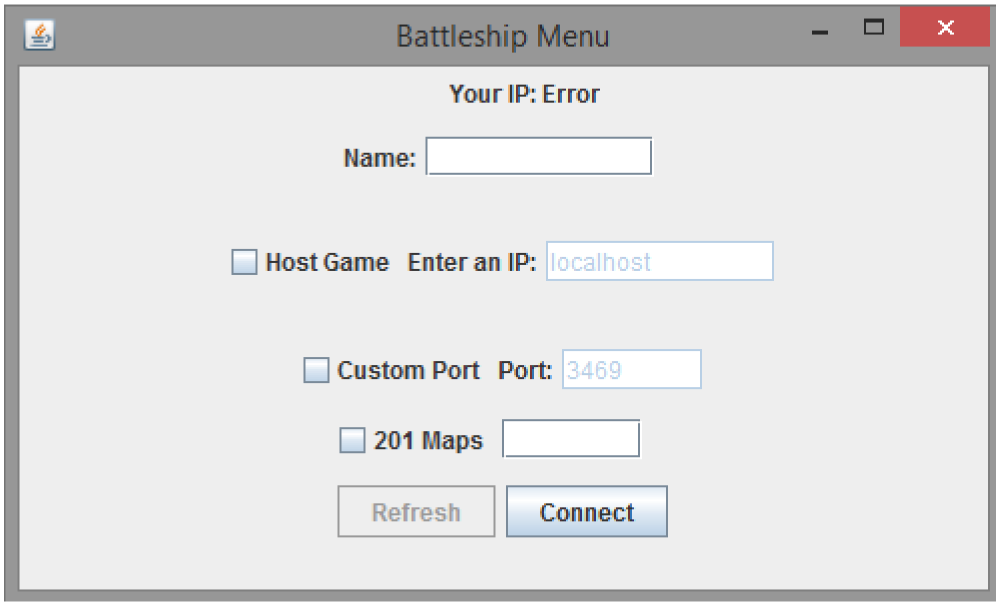
Host Game Enter an IP: 192.0.2.0

Custom Port Port: 3469

201 Maps

This will also help to determine if the user is connected to a network or not. Add a refresh button so they don’t have to close and reopen the application in case they disconnect.

When the user starts the program while they are not connected to the internet, the following will appear:



Your IP: Error

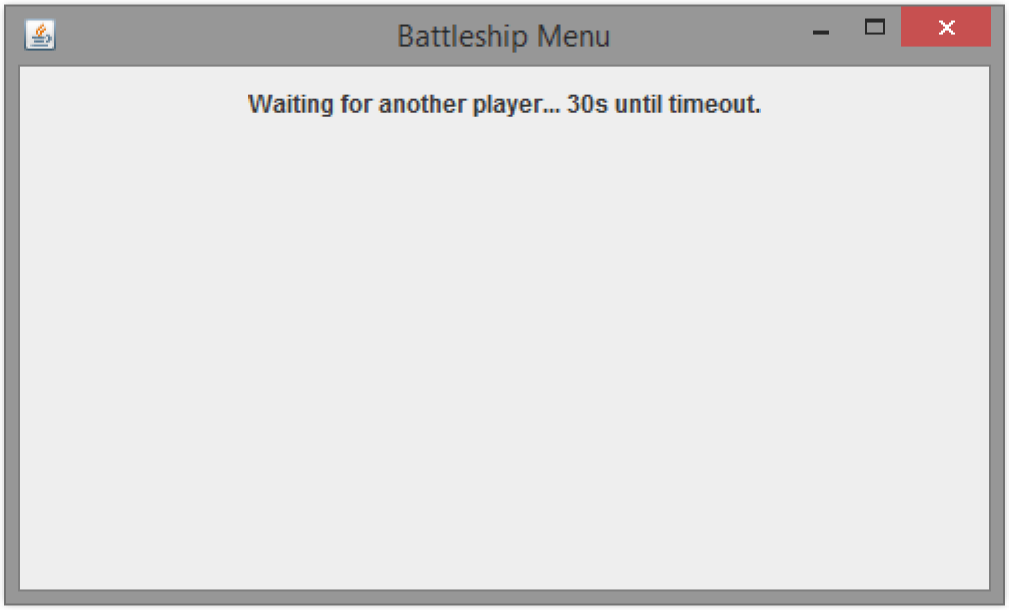
Name:

Host Game Enter an IP: localhost

Custom Port Port: 3469

201 Maps

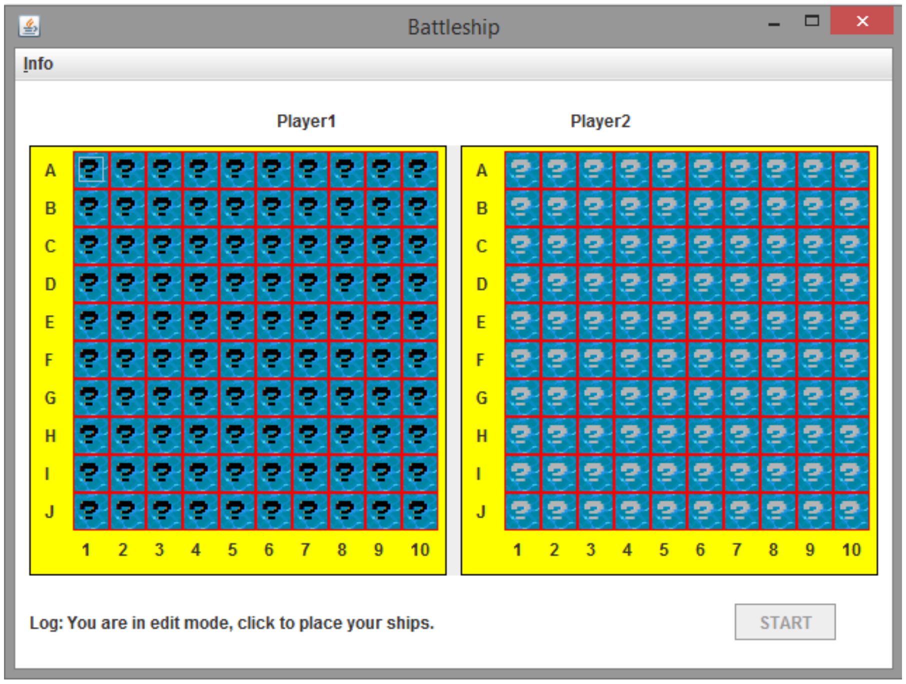
Step 2
The host will need to start the server and wait for the other player to connect. It may appear that the game is frozen for them. Add a window that displays 'Waiting for opponent…'. Have this window time out after 30 seconds. If it times out, stop the server and return the host to the first menu.



Waiting for another player... 30s until timeout.

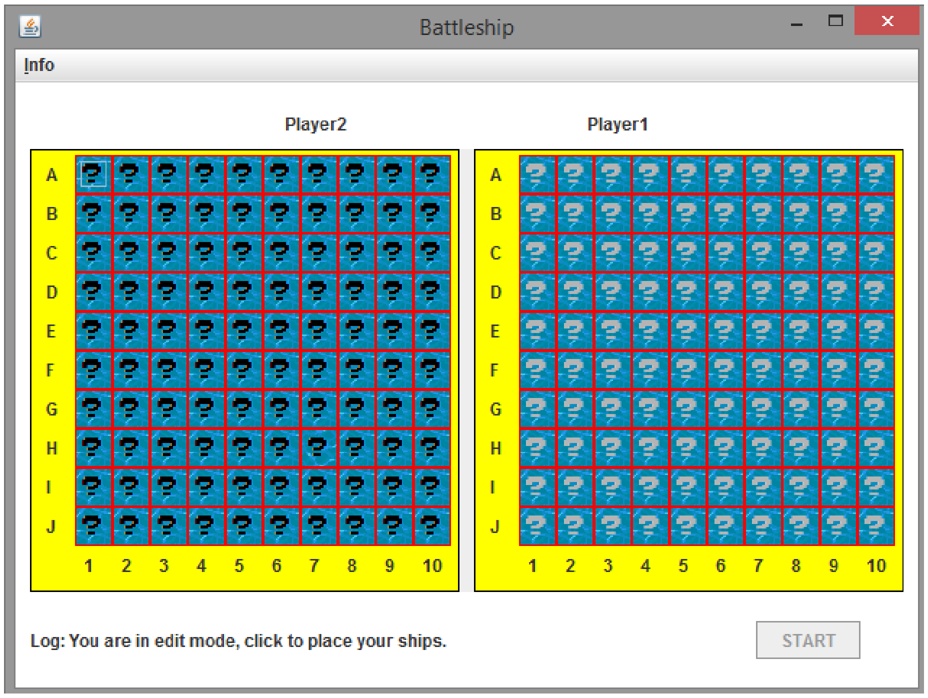
If the connection is made between two players, bring them both to the Battleship game board. Instead of 'PLAYER' and 'COMPUTER', enter the names they chose at the beginning.

Player 1:



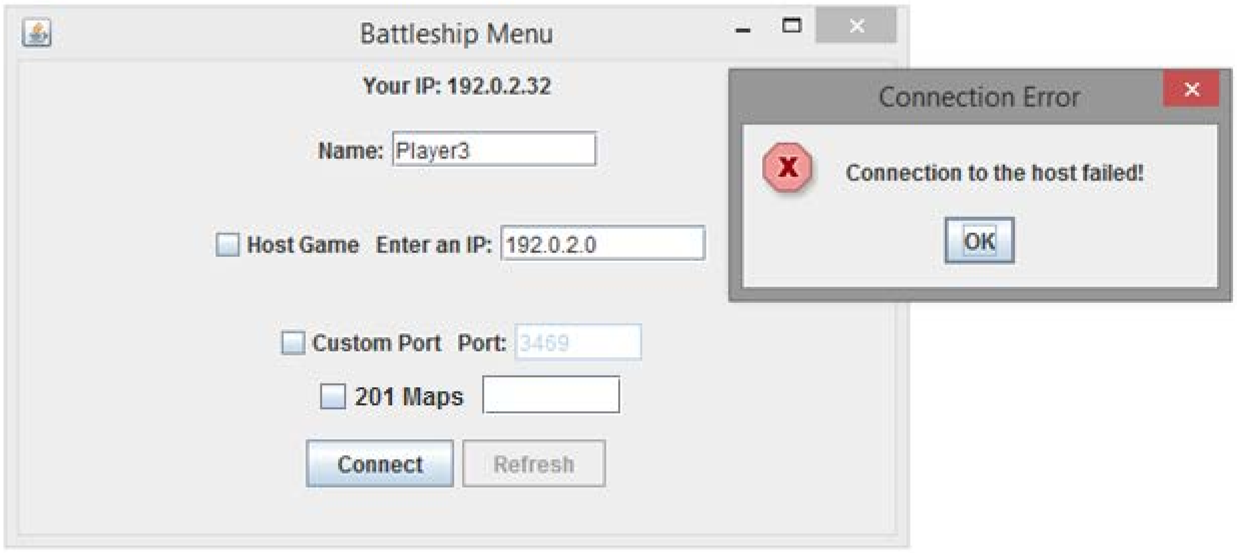
Log: You are in edit mode, click to place your ships.

Player 2:



Log: You are in edit mode, click to place your ships.

We only want to handle two players at once. If a third player tries to join, or if the player can't connect to the given host, simply open an error window letting the user know the connection failed.



Your IP: 192.0.2.32

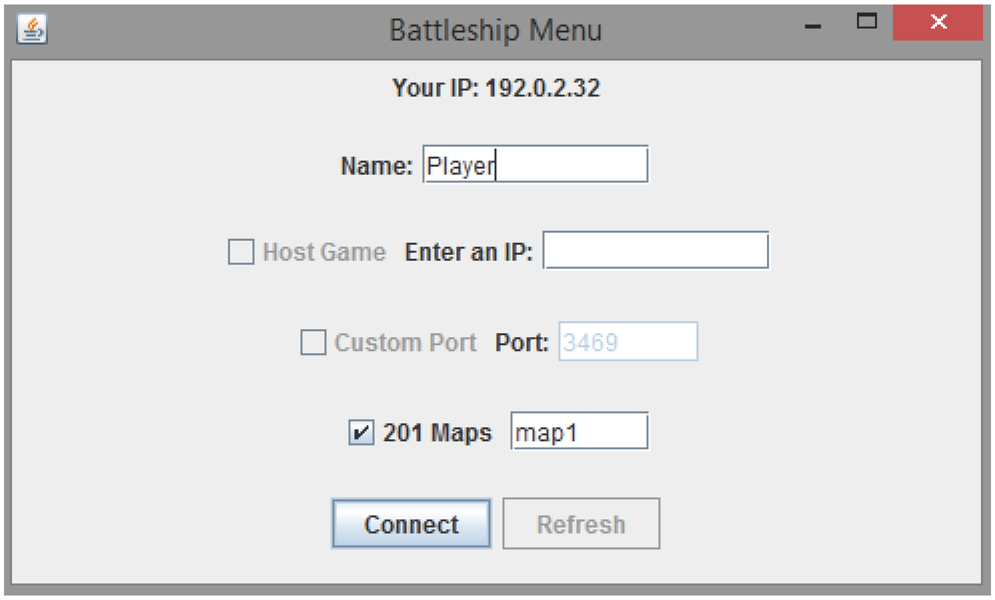
Name: Player3

Enter an IP: 192.0.2.0

Port: 3469

Connection to the host failed!

Step 3
Add the ability for players to get a map from the CSCI201 website. These maps will be in the old .battle format. The user will check the "201 Maps" option, along with a file name, but not with the directory. The files will be found in the http://www-scf.usc.edu/~csci201/assignments/ directory, and they are named map1.battle, map2.battle, map3.battle, map4.battle, and map5.battle. You should not download the files, but they should be accessed by your program over the Internet.



Your IP: 192.0.2.32

Name: Player

[ ] Host Game Enter an IP:

[ ] Custom Port Port: 3469

[✓] 201 Maps map1

When this option is selected, the player will play against the computer. The computer will use the file that the user selected from the server. The play will be the same as in the previous assignment – the only difference is that instead of choosing a file that is on the user's local machine, the user will choose a file from the website.

Step 4
We need to let the users create their maps. Done! –we did this in a previous assignment! We just need to save the data to a file and send it to the server. That way, when a player makes a move, the server can decide if the player hit a ship or not.

You will probably want to use .json format, although it is up to you. Just don't use the bad file format we've been using. Remember when our original file format had that annoying issue with D's while parsing it? We don't want the user to choose two horizontal ships and the server to interpret that as two vertical ships.

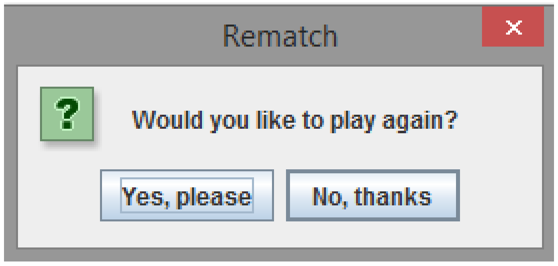
Send the file to your server program once a player presses 'Start'. Only allow users to press 'Start' once they fill in all the ships – just as in the previous assignments.

Note: JSON is just a recommendation, you can use whatever you'd like. However – it's strongly suggested since you have experience with parsing JSON files from lab.

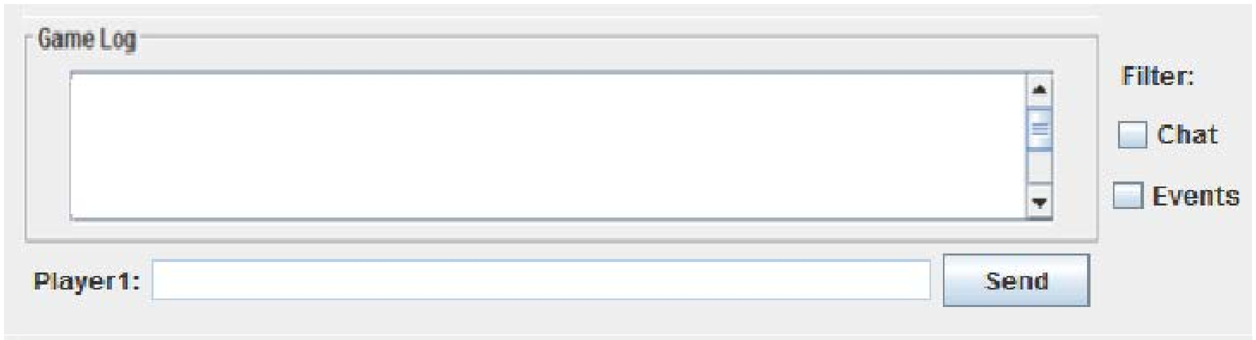
Step 5
Now the server has all the information it needs for the players to play the game. It’s time to network the actual play! Just send the coordinate the player selects to the server, have the server process it, and update the clients.

Everything from the previous assignments should still work, including the timer functionality. You are just taking input from the server instead of from the computer. Animations, sounds, and event messages are all handled by the client. You will just need to remember to replace the names of the players in the event log.

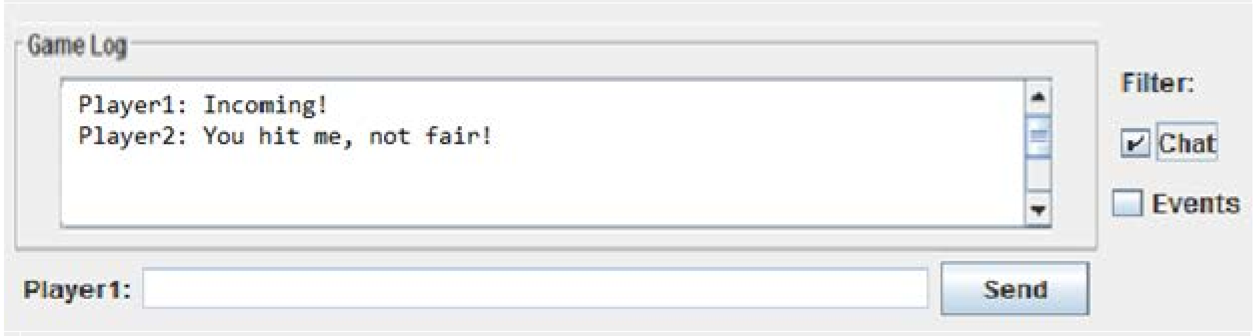
Once the game is over, let the users decide if they want to play again or quit. If the players both decide to play again, they will both start back at the menu to create a map. If they quit, they will return to the main menu.



Step 6
Let’s add the ability for players to be able to communicate with one another. We already have a log, but we are also going to use that for chat in addition to announcing events. Let the user be able to select if they want to view events, messages, or both. Make sure to keep track of events or chat even if the player isn’t viewing them. When they switch back, they should be able to see what they missed.

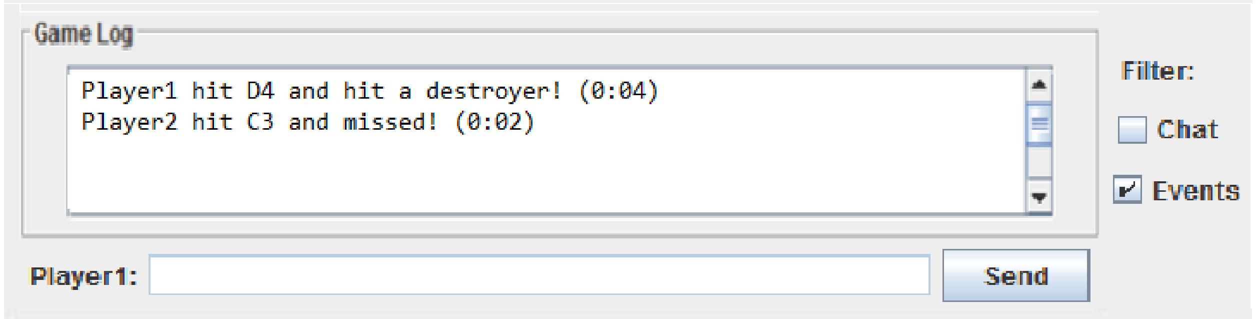


Filter:
☐ Chat
☐ Events



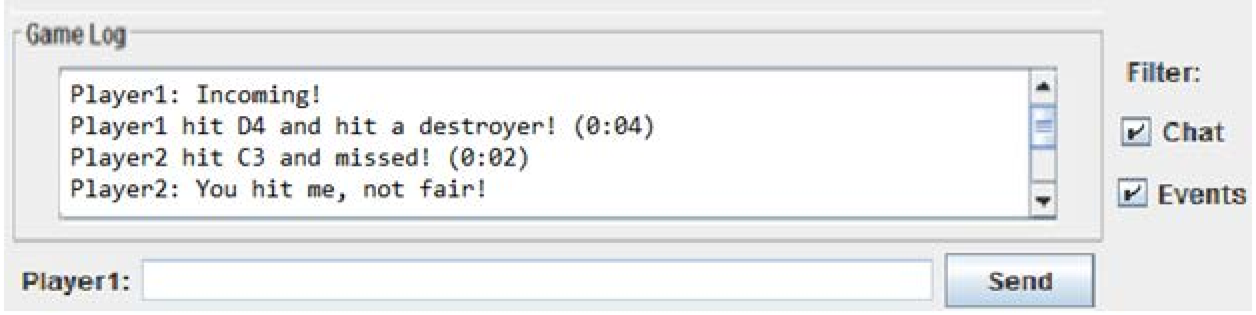
```
Player1: Incoming!
Player2: You hit me, not fair!
```

Filter:
☑ Chat
☐ Events



```
Player1 hit D4 and hit a destroyer! (0:04)
Player2 hit C3 and missed! (0:02)
```

Filter:
☐ Chat
☑ Events

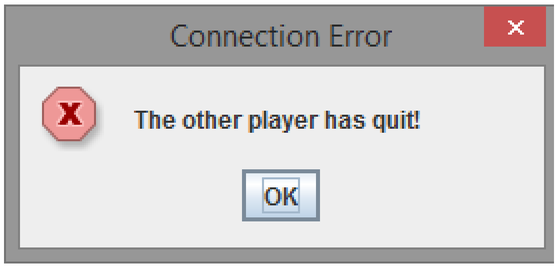


```
Player1: Incoming!
Player1 hit D4 and hit a destroyer! (0:04)
Player2 hit C3 and missed! (0:02)
Player2: You hit me, not fair!
```

Filter:
☑ Chat
☑ Events

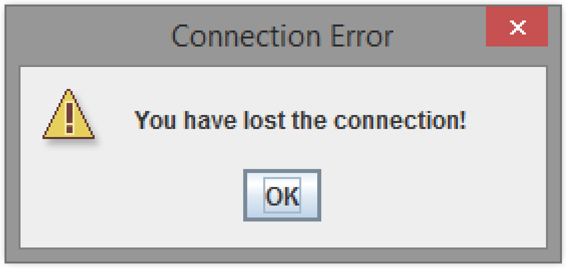
Step 7
In previous assignments, the computer never gave up or quit. In a multiplayer game, it is possible for the player to quit or disconnect unexpectedly. Make sure to handle this.

If a user simply closes the application, display the following message to the user who is still connected. After pressing 'OK', simply return the player who didn't quit to the main menu. This means that you will need to transmit a message to the server (and then to the other player) when a user closes the application.


Connection Error

The other player has quit!

If a user loses internet connection, display the following message. Take the user back to the main menu, as you did when the user didn't have a connection initially in Step 1.


Connection Error

You have lost the connection!

Grading Criteria

Category and % of grade	Criteria
Menu 0.75%	
0.25%	The menu contains all the required fields.
0.25%	The menu displays the user's IP.
0.25%	The menu can refresh if no connection is found.
Connection to Server 1.0%	
0.4%	The host waits 30 seconds before timing out.
0.4%	If a connection is made between two players, a battleship grid appears with both players' names.
0.2%	If a player attempts to join a server with a game in progress, he will get an error message.
201 Website Maps 0.75%	
0.75%	The user can retrieve .battle files from the CSCI201 website and use them in a single-player game mode against the computer.
Custom file 0.75%	
0.5%	A non .battle file type is used to transmit map info to the server.
0.25%	The two Destroyer ship issue is solved in multiplayer mode.
Networked Gameplay 1.25%	
0.8%	The game is fully functional with networking.
0.2%	The player names are used in the event log.
0.25%	Players can play more than once if they choose to.
Chat System 1.0%	
0.5%	Players can send messages to one another through the server.
0.5%	The chat/event log can be filtered.
Error handling 0.5%	
0.25%	If the player loses their network connection, they get an error message and are returned to the main menu.
0.25%	If the player quits the game during play, the other user gets an error message and is returned to the main menu.