# UIC Spring 2019

# CS 342 Project 5

# UNO

---

**Team Number:** 14

**Section:** 41553

**Members:** Michael Aiello, Ryan Anderson, Thomas Hein, Anthony Slas

---



---

## Purpose Of Project

This JavaFX UNO project will serve as a responsive server client desktop GUI application. Clients will be able to connect to server and play a four person game of UNO. The server will handle game logic and all events and information will be relayed via the server and client GUIs as well as their communication threads.

There is no problem that we are attempting to solve, except that this project is acting as our CS 342 final team project. I suppose the problem can be how to implement a four person JavaFX networking involved game.

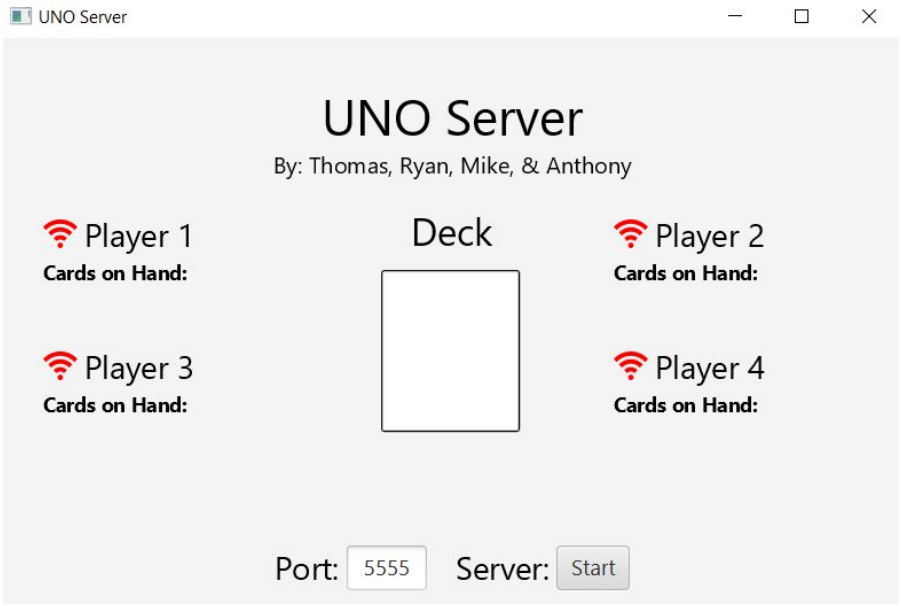
This project needs to exist as it is a big component of our team members final grade in the course, and without it we would be taking a rather large grade deduction.

As for who will use it, we say anyone who pleases. We will be playing the game as we develop it as well as after it is turned in. The grading instructor will have to play it as well.

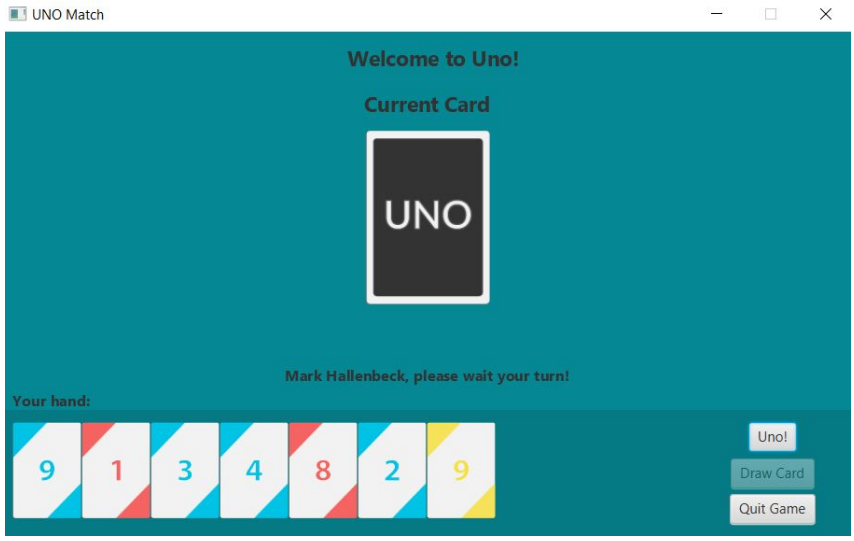
## Define High Level Entities

- **Server Side**
    - Controller
        - Will access the GUI
    - Server
        - Will control everything game related (players, cards, deck, scoring, etc)
    - Various FXML files
        - Will hold the actual design/code for look and feel of server GUI
- **Client Side**
    - Client Manager
        - Will handle all the client related data (connection, name, cards, etc)
    - Game Scene
        - Will have all game playing related attributes here (uno buttons, clickable cards, draw button, etc)
    - Server Connection
        - Will handle the communication to the server
    - Welcome Scene
        - Will server as the landing page for clients before they connect to the server
    - Wait Scene
        - Will server as the waiting scene while clients are waiting for enough players to connect to begin gameplay
    - Error Scene
        - Will server as a general error screen for when a gameplay or communication error occurs
    - Various FXML files
        - Will hold the actual design/code for look and feel of server GUI

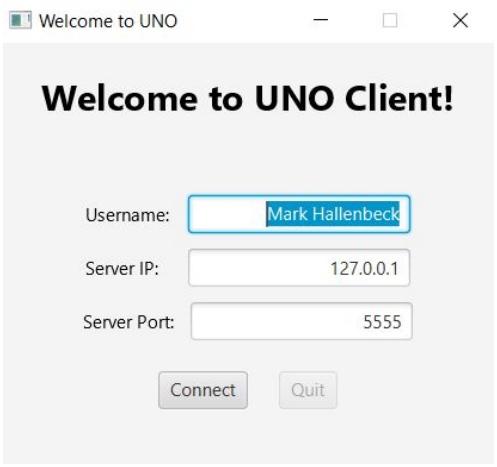
## Define Low Level Entities

- **Server Side**
  - 

  - Controller
    - Will have direct access to the server GUIs start server button, port text box, the info box, as well as the players turn arrow, players name, players hand string
  - Server
    - Card
      - Will hold all data related to a card, what type, the name, the number, speciality
    - Deck
      - Will hold all data related to the deck such as dealing cards, the deck of cards itself
    - Server Socket Thread
      - Will be responsible for listening for incoming client connections
    - Player Thread
      - Will be responsible for handling all client related communication, string parsing, and reaction to said strings
    - WIll store the player threads and deck in list data structures
  - Various FXML files
    - Will hold the actual design/code for look and feel of server GUI
- **Client Side**

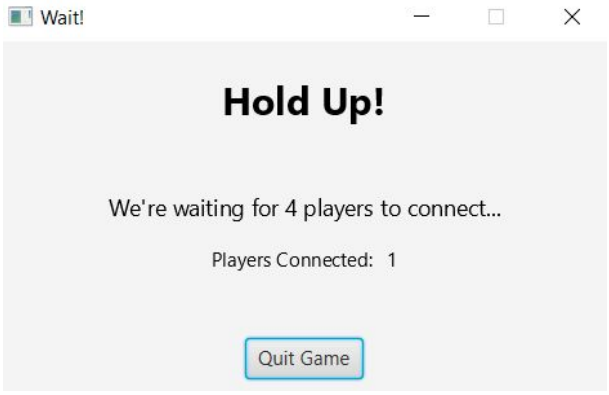
- Client Manager
  - Will handle all the client related data as well as scene switching relative to what the client has pressed on the GUI. It has access to connect to the server, reset menus, change scenes depending on input
- Game Scene
  - 

  - This scene will hold all game buttons and game data. It will show the current top card of the discard pile, the client's current hand, whether or not it is their turn, a draw button, a uno button, and a quit button
- Server Connection
  - This is important as it is responsible for sending and receiving messages from the server. These messages are vital to gameplay and scene switching. It has an important function interpretData that parses the server message and acts accordingly.
- Welcome Scene



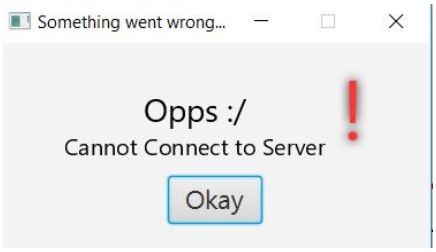

- Has a username, port, and ip input box. A connect and a quit button for the user to enter data and connect to the server or quit

- Wait Scene




- This scene is quite simple. It simply just says wait, and keeps a visible count of the connected players on the server

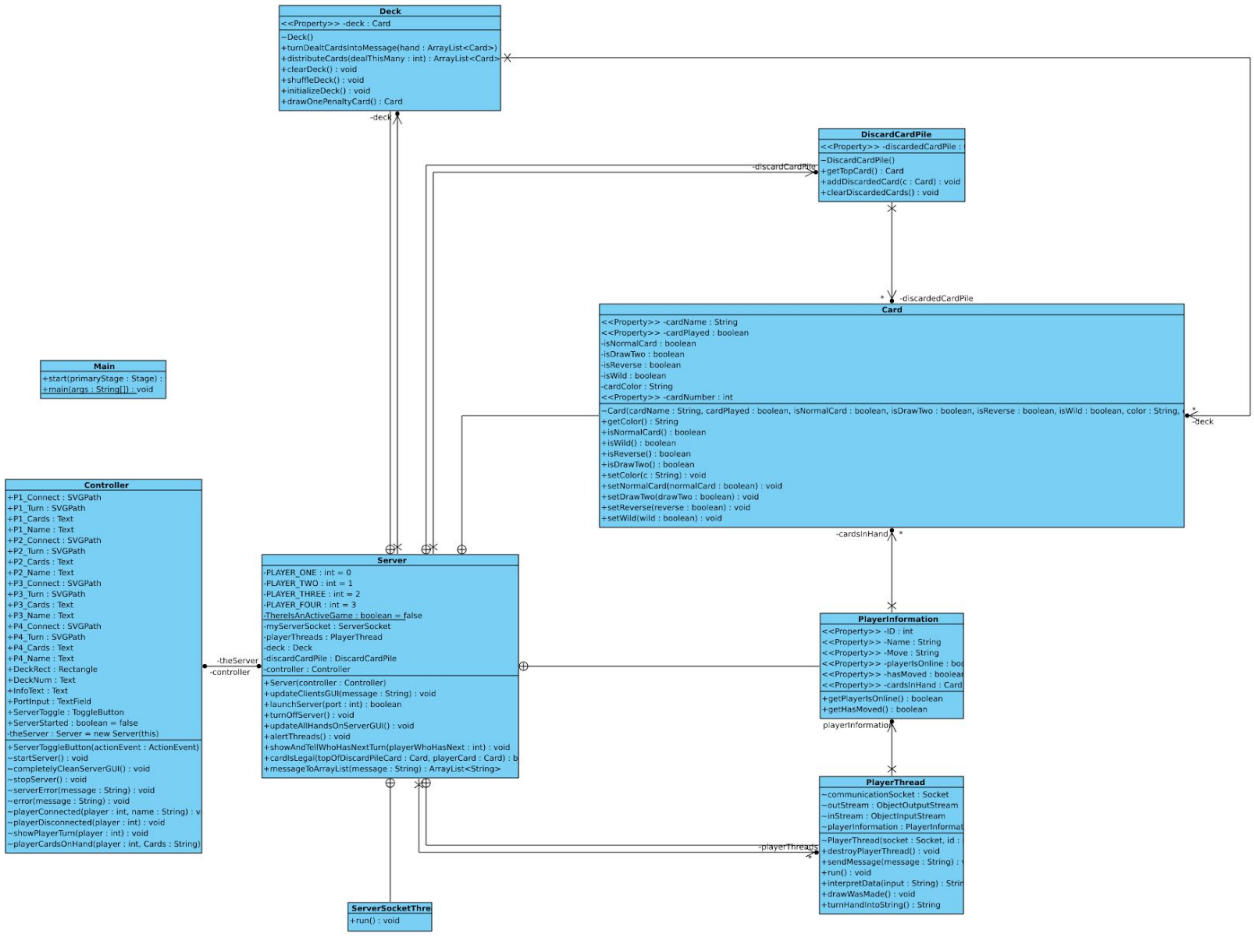
- Error Scene



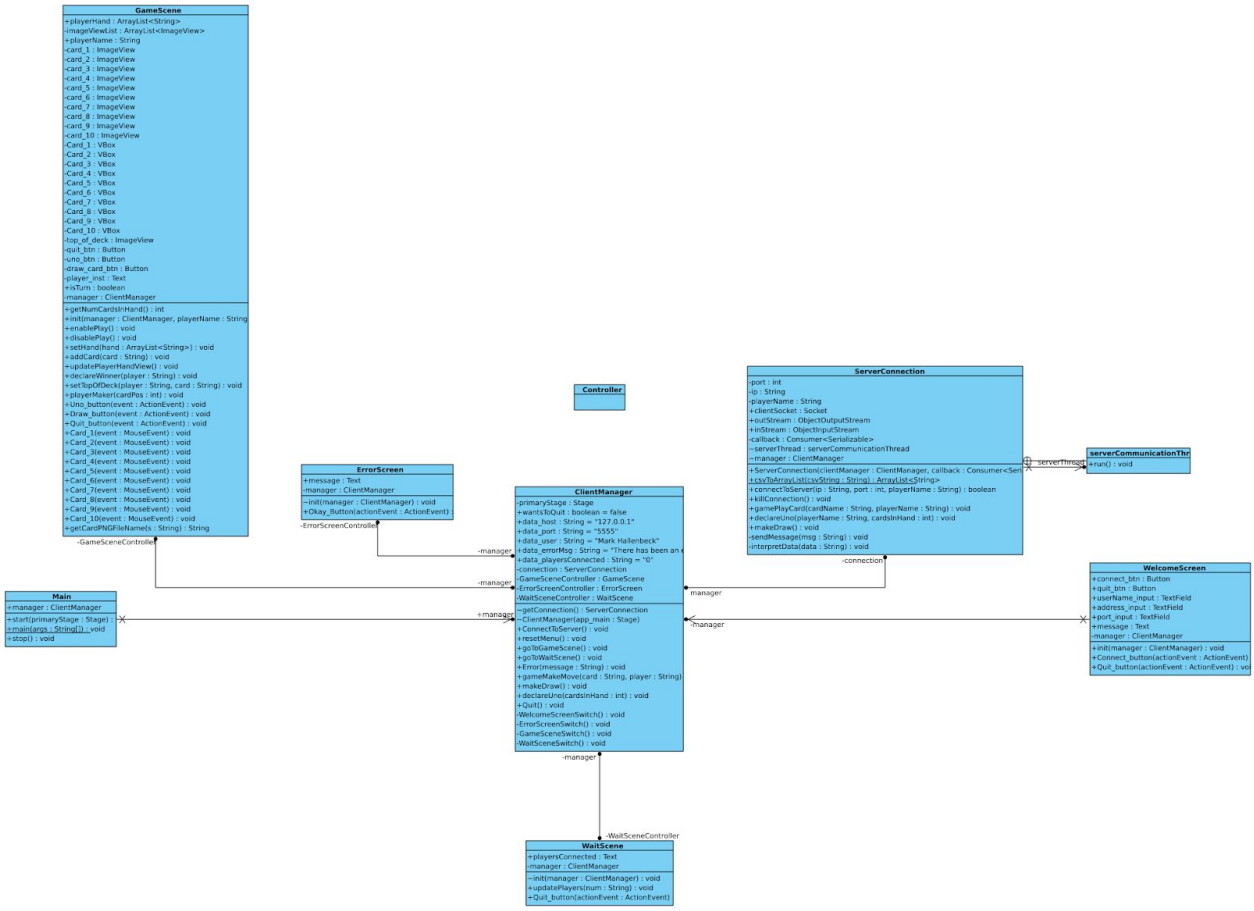

- Just a scene that notifies of an error and promptly returns the user back to the welcome scene

- Various FXML files
  - Will hold the actual design/code for look and feel of server GUI

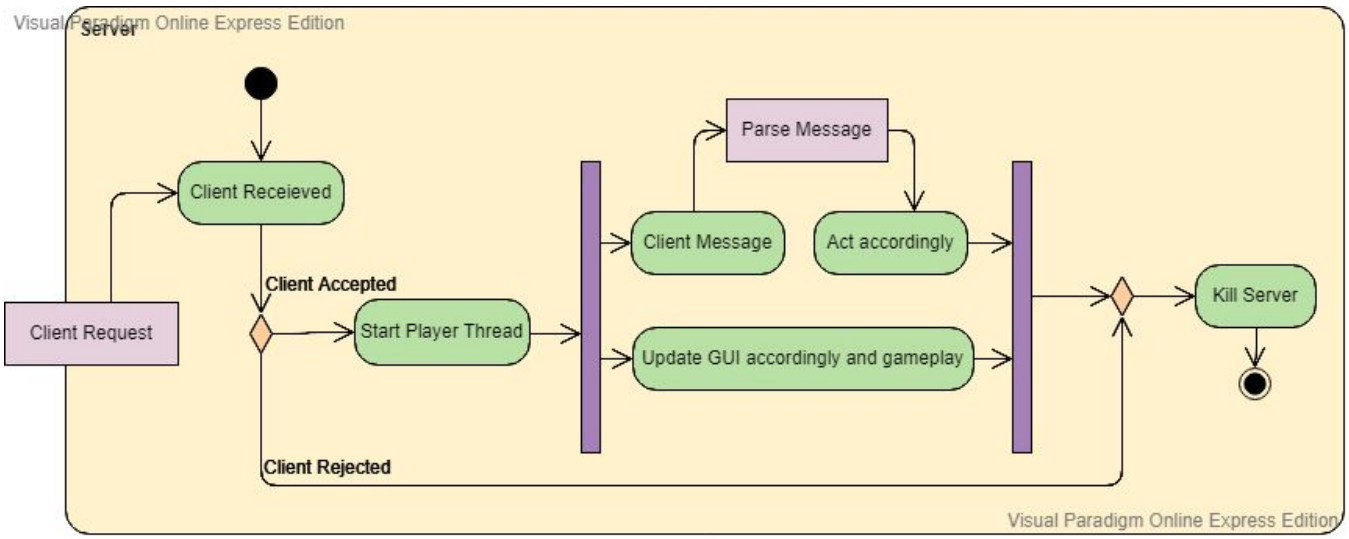
# UML Diagram Server



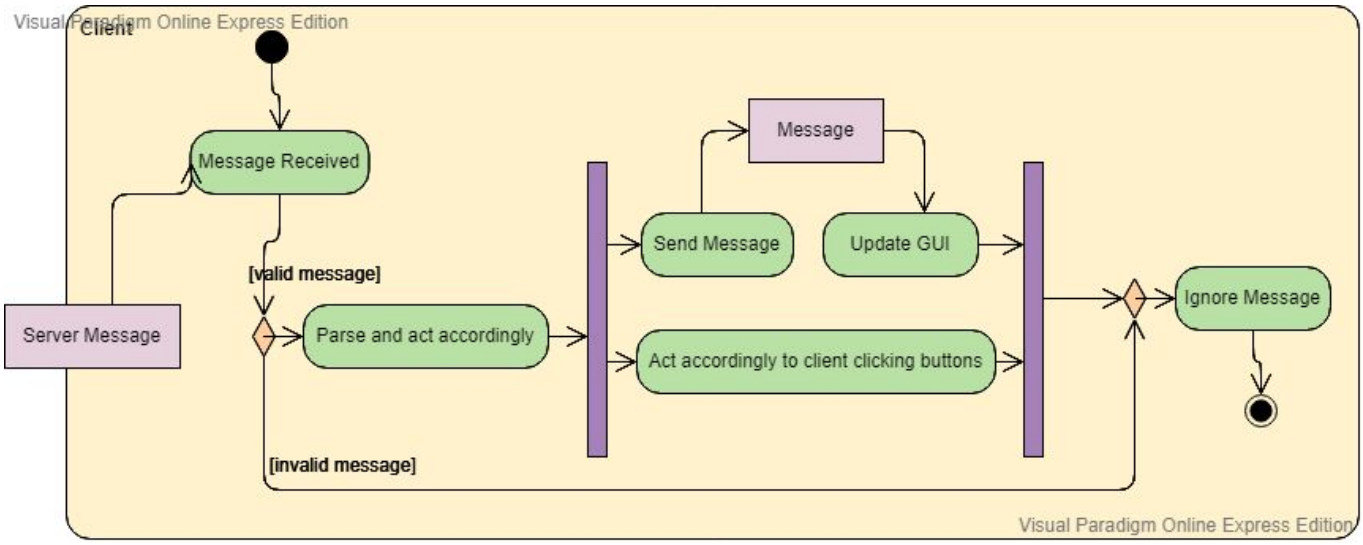
# UML Diagram Client



## Activity Diagram Server




## Activity Diagram Client



## Benefits, Assumptions, Risks and Issues

- Benefits
    - The server and client are separate entities
        - Better to split up code, GUIs, and functionality, etc
    - All gameplay is done through the server (limits errors on client side as there are multiple clients vs one server)
    - Scalability
        - By making the server only responsive to certain messages we can add or remove functionality quite quickly.
    - By having the server automatically accept clients until four players have been accepeted, there is no need to hard code a certain number of threads
    - We simplified gameplay, so there is less chance of gameplay/runtime errors
- Risks and Issues
    - If we do not correctly kill connections we can have hidden running threads (this is not an issue, just a POTENTIAL risk)
    - There can be a lot of concurrency exceptions if we do not utilize Platform.RunLater() functionality
- Assumptions
    - That UNO will only be played with 4 players. No more no less
    - That the server host and client have working internet connections as well as open ports to host the game