Shien Hong, Kegan Schaub

CSc422 - Project 2 report

Patrick Homer

05/06/15

The project we set out to create is of the client-server model, where a server communicates to a distributed set of computers and coordinates the card game Crazy Eights with them. The following report will explain how our project is designed, what we did to test it, and what we added to make it more polished.

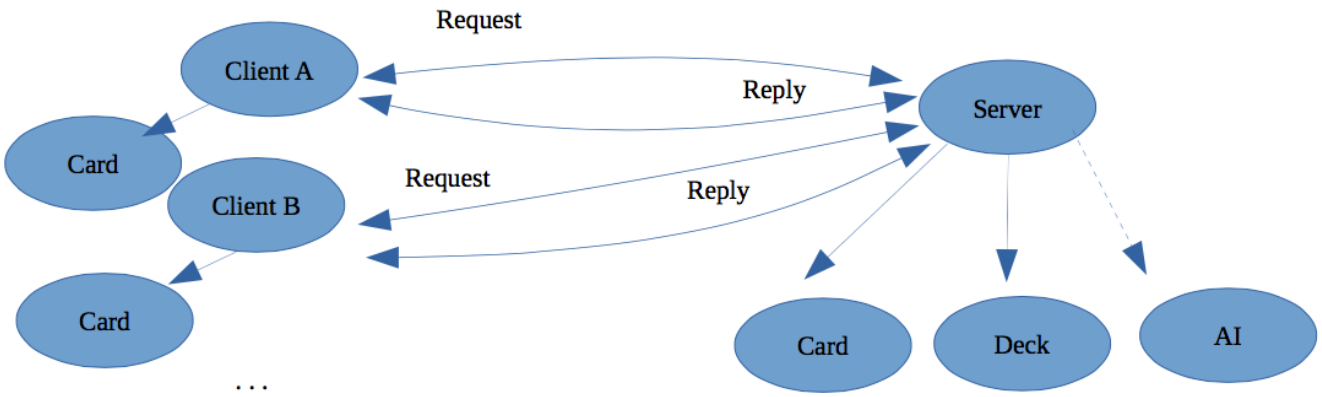
In our implementation, we used Java with sockets. The project consists of the Java programs Server, Client, Deck, Card, and AI. Server is responsible for creating a deck of cards using the Deck class, and taking advantage of all the methods that it provides, such as shuffle(), draw(), topDiscardCard(), and discard(Card). Once the server starts, it prompts the user with "How many players? 1 for playing against AI" and "Number of Draws before skipping to next person? num <= 0 : for unlimited drawing " and waits for all clients to connect. The client then connects using server's IP address and port, and receives 5 cards from Server. The server will pick who's turn it is and show the top discarded card to them before the client gets to pick which card to play. The first client to run out of cards wins the game. We used the following to implement the card game.

Rules of Crazy Eights

- Everyone starts with 5 cards.
- For every for players, add another deck of cards to the current deck.
- Players must play a suit or rank that matches the top discarded card.
- If a player plays an eight, they must choose a suit to put down.
- If a player cannot match, they must draw a card until they can play a card or have reached a maximum draw count.

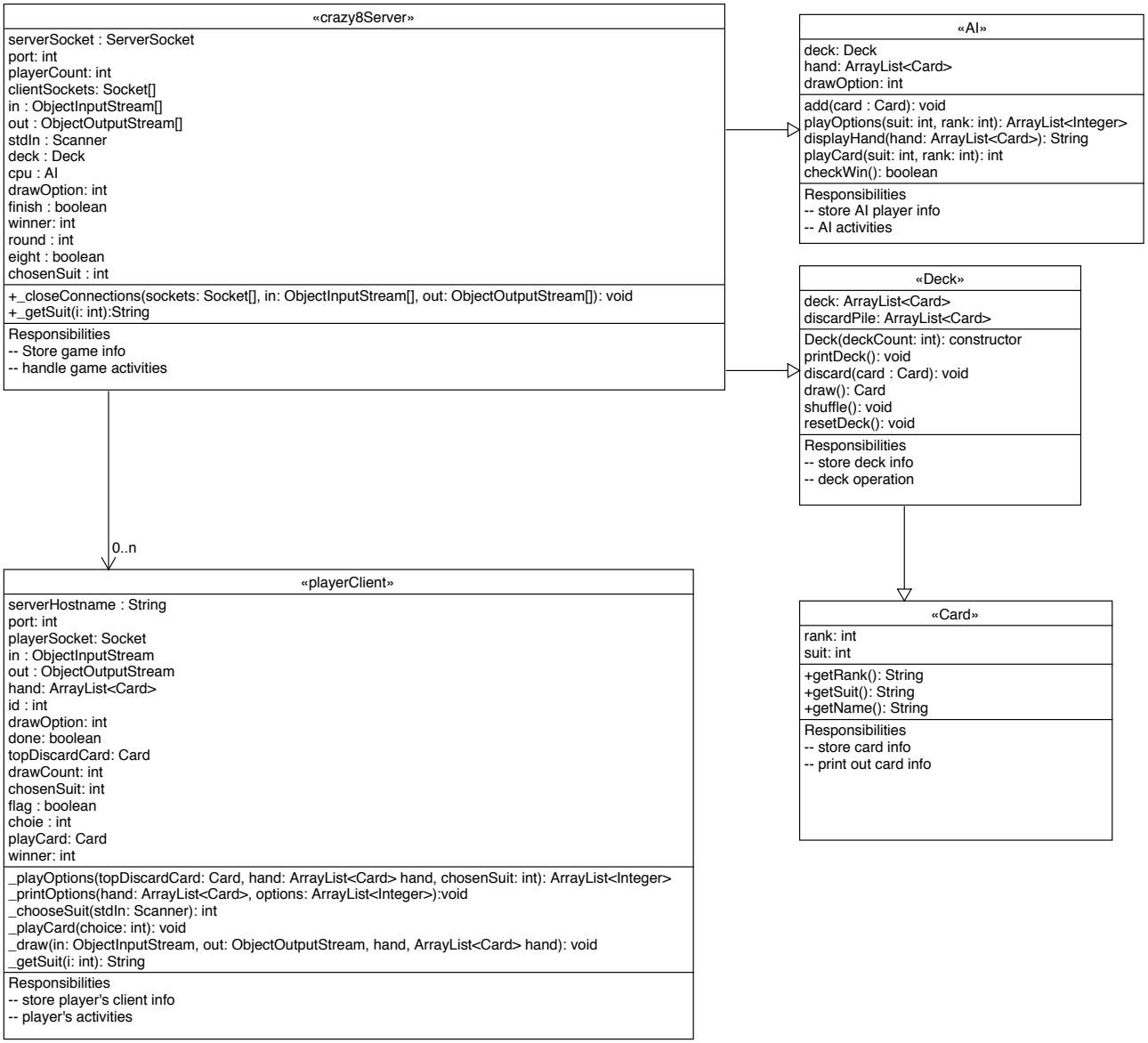
- A player wins once they have played their last card.

For testing the project at home, we used the integrated development environment IntelliJ IDEA because it made debugging fast and easy. Once we were comfortable with how it ran on our own machines, we decided to test it on multiple machines at the GS-930 lab. The server and clients connected flawlessly with each other, and we tested up to 4 machines to verify that it worked. In the diagram below, you can see how the server and client communicate with each other.



Client-server diagram

The server's job is to hold the deck of cards, follow the rules, and talk to the players (aka clients) to let them know who's next and what should be done. Both Server and Client depend on other Java class(es) to get work done. Server needs to address Card, Deck, and maybe AI if the Server was told in stdin that only 1 client was to play. Client needs to address the Card class so that they can receive cards to put in their hand, as well as send cards to the server. When we were coming up with the structure of the program, we came up with the following UML diagram to plan our design.



Since we are a group of two, we decided to make our project more ambitious by adding a command-line argument prompts for stdin, and an artificial intelligent player for a client to go up against. Before, the user would have to edit the host name and port variables in Server.java or Client.java before running the programs. Now, they are able to run Server with the port as a command-line argument, or wait for the stdin prompt for port and IP address after starting Server or Client. On the server prompt for how many players there are, it says to type 1 to go up against the AI. Once this option has been enabled, the server consults the AI class as a player on the game.

In our report, we discuss how we implemented the card game Crazy Eights in our distributed

project, verified that we were able to make a connection, and explained how we have made the project more polished. Overall it was fun experienced and we learned a lot about distributed design.