1-2-3 Pass Card Game

Description: The objective of the game is to get 4 cards with the same ranks. Each player will start with four random cards. Everyone will then say "1-2-3 Pass" which will start the game and everyone will pass a card on their right. This will continue until a player gets four cards with the same ranks. The last person to match all four will lose the game.

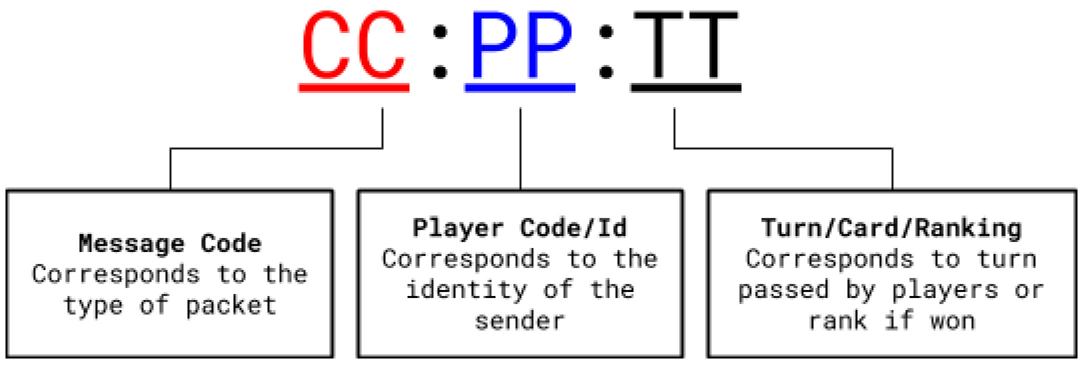
Objectives: To create a client-server simulation of the 1-2-3 Pass game using socket programming

Group Name: I 1-2 Pass
Group Members:

- Abarro, Jethro
- Samarista, Rainier
- Tapia, Orlando
- Villanueva, Ezra

Programming Language: Java
Integrated Development Environment (IDE): IntelliJ, Netbeans 11
Github Repository: https://github.com/jetabarro/CMSC137-PROJECT

Communication Protocol:
The client and the server will communicate using the following string as a packet:



CC:PP:TT
Message Code
Corresponds to the type of packet
Player Code/Id
Corresponds to the identity of the sender
Turn/Card/Ranking
Corresponds to turn passed by players or rank if won

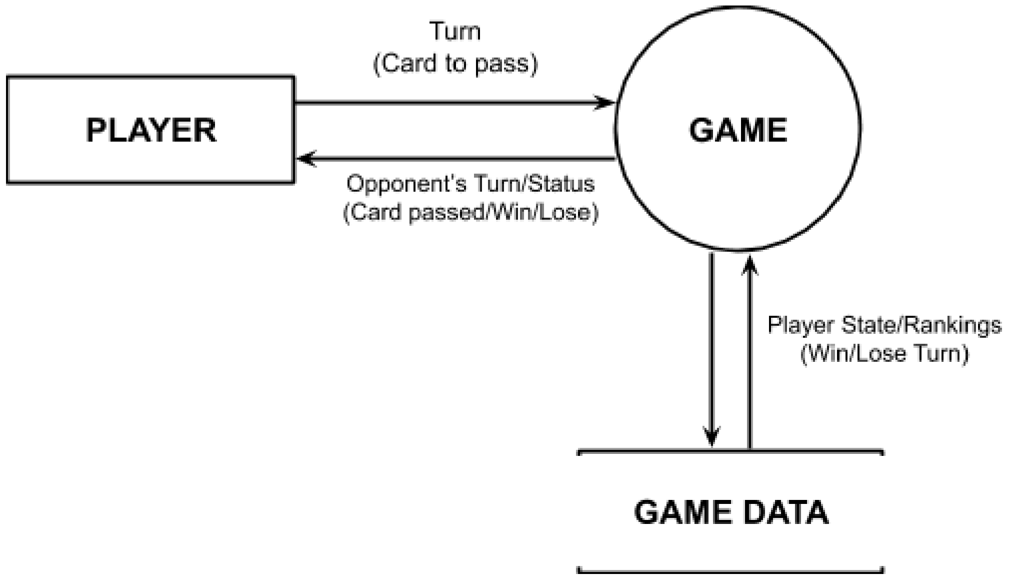
The following codes will be used for the cards to be passed (**TT**):

Ranks	Suits
A - Ace **2-0** - Cards 2-10 **J** - Jack **Q** - Queen **K** - King	**D** - Diamonds **H** - Hearts **S** - Spades **C** - Clubs

The following codes will be used as packet codes (**CC**):

Codes	Description
00	Ready
01	Start/Count
02	Pass
03	Cards Matched

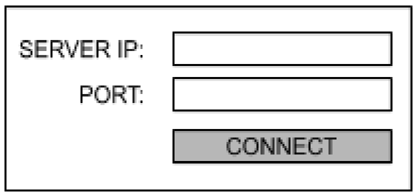
Data Flow Diagram:

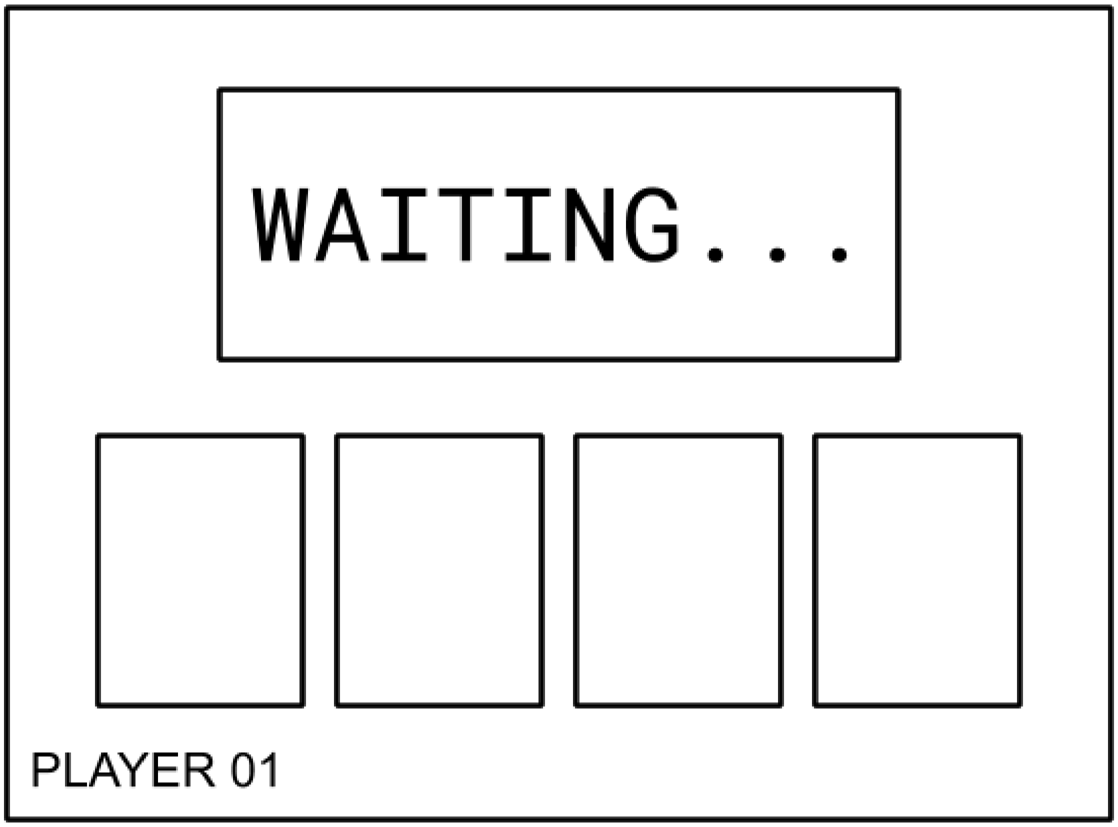


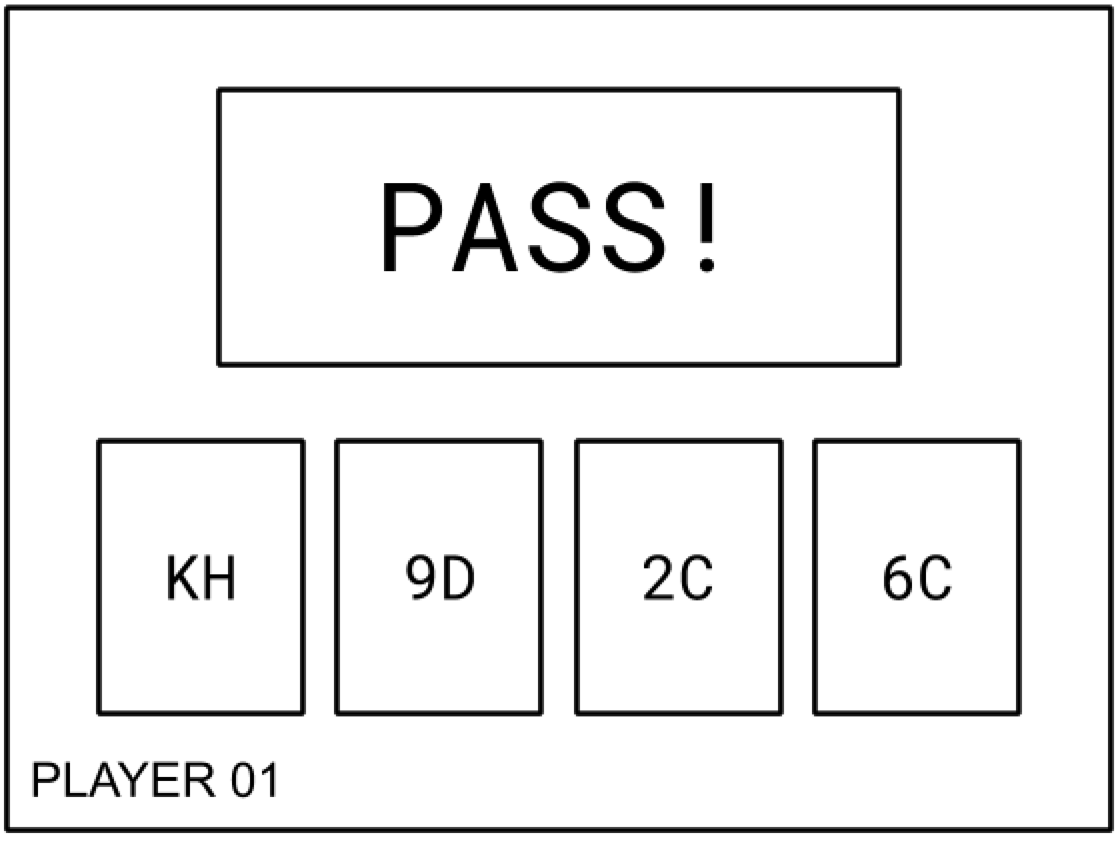
Game Layout:

Client: The basic layout of the client will be composed of a large label containing the status of the game as well as the “1-2-3 Pass” signal and the buttons which the player could press to mark his/her turn. To ensure that the card will always be passed, the rightmost card will always be selected before the game and after every passing.

The user will need to enter the ip and port of the server to be able to connect to the game. A label indicating that the client has connected will be displayed upon successful connection.




WAITING...
PLAYER 01


PASS!
KH
9D
2C
6C
PLAYER 01

Coding Guide:

The following will serve as a guide in the development of the project.

Client	
	● Contains the main method, extends JFrame. ● Contains three JPanels (one for the join/connect to game, one for the game) 　○ *connectPanel* - contains the inputs where the user will input the server ip and port. This will be hidden upon successful connection. 　○ *gamePanel* - contains the cards as well as the timer or message prompt to pass. Timing or signal to pass will come from the server. ● Contains four cards initialized using the randomized labels given by the server. This means that until the game starts, blank cards will be displayed. ● Contains a *queuedCard* variable which is set by the selected card. If a card is selected, an event will trigger that sets this variable with the card name/label. However, to prevent players who have not selected any cards, the leftmost card or card[0] will always be set initially. ● A listener will continuously receive a packet from the server and will be interpreted depending on the type of packet it receives. 　○ 00 - Signifies that the client has connected to the server and will set the *clientId* variable 　○ 01 - Signifies that the client should now display/start the countdown and pass accordingly depending on the packet. 　○ 02 - Signifies that a client passed the said card and the passed card must be reset by calling the *Card.setCard(TT)* method. 　○ 03 - Signifies that a client has won following the player number. Receiving packet 04:00:TT means that the game is already finished. **Methods:** ● *connect(serverIp, port)* - this method connects to the server using an ip address and port number. The server returns a two number string (PP), excluding 00, which represents the id of the user using the code: 00:PP:00. This is then saved on a *clientId* variable. The client now waits for the first player to start the game or receive a packet with 01 prefix.

	• *pass()* - this method sends the value (TT) of the *queuedCard* variable to the server using the code: 02:PP:TT • start() - this method is only available to the first player who joined the game. This sends a packet which signals that the game will start. This will send the code 01:PP:00 (PP signifies the id of the user) to the server and the server will now generate the pool of cards, randomize them and distribute them to the players through code 00:PP:TT in 4 different packets. Upon completion of the distribution of cards, the server will now broadcast 01:00:01 which will start the timer for all clients. The end suffix (TT) with prefix 0 will denote the 1-2-3 Pass signal, 01:00:01 for 1, 01:00:02 for 2, 01:00:03 for 3 and 01:00:04 for pass.

Server	

	• No UI, only console-based. Logs will be printed depending on the signal received/acted upon.
	Sample log for end of passing signal:
	`[PLAYER 01] AK 0H 2H 7H`
	`[PLAYER 02] 6S 7H 1C 9D`
	
	Sample log for passing:
	`[PLAYER 01 > 02] AK`
	`[PLAYER 02 > 01] 7H`
	
	Sample log for end game:
	`RANKING:`
	`  1. PLAYER 2`
	`  2. PLAYER 1`
	
	Sample log for connected players:
	`[PLAYER 01] 192.168.2.31 has joined the game.`
	`[PLAYER 02] 192.168.2.32 has joined the game.`
	
	Sample log for game start:
	`[PLAYER 01] has started the game.`
	
	`Distributing cards...`
	
	`INITIAL CARDS:`
	`  PLAYER 01 - AK 0H 2H 7H`
	`  PLAYER 02 - 6S 7H 1C 9D`
	• A listener will continuously receive a packet and act on it as described above.
	    ○ 00 - Signifies that a client is trying to connect. This stores the ip to the dedicated data structure and assigns an id to it. The server will

	broadcast the appropriate packet to notify the client of its id. ○ 01 - Signifies that the game will now start. Server will now start its internal timer and broadcast the appropriate packet to signal 1-2-3 Pass. *check()* is called after every pass signal before starting the countdown/sending the 01:00:01 packet again; ○ 02 - Signifies that a client is passing the said packet to the specified player. This sends the same packet to the specified client, replacing the player id of the packet with the id of the sender.