Poker game

(Sprint1 Planning Document)

https://github.com/srhee91/PokerGame

Team 3

So Mi Choi, Bo Heon Jeong, Hanchen Li,
Sang Rhee, Yixin Wang, Yuying Wang

Sprint1 Planning Document

1. User stories to be implemented

PokerGameState:

1. As a user, I would like the *PokerGameState* to contain all the necessary fields to completely describe the current state of the game (e.g. whose turn it is, which round it is, the current bet, the current pot, etc).

GUI Part:

2. As a user, I would like to see the overview of the interface of four modes, including all the buttons, cards, player icons.

Network Part:

3. As a user, I would like the *HostMessageHandler* class to be able to listen on a specific port for player action messages from the *ClientMessageHandler* and to be able to send game state messages back.
4. As a user, I would like the *ClientMessageHandler* to be able to connect to the *HostMessageHandler* and send player action messages as well as receive game state messages from the *HostMessageHandler*.
5. As a user, I would like the game server to be able to communicate with up to 20 clients (players or spectators) with no discernible increase in latency. This way, the server will be able to easily communicate with all players in an ongoing game as well as a significant number of spectators.

2. Description of tasks

PokerGameState (80hrs)

1. Create the *PokerGameState* class and implement all the classes and variables needed for the *PokerGamestate*: *PlayerInfo*, *TableInfo*, *Pot*, *Bet*, etc.

   (Sang Rhee, 40hrs) (Boheon Jeong, 40hrs)

GUI (80hrs)

2. Design the UI elements and their placements on the screens of the four game modes. No interactivity will be implemented, but the look of the four UIs will be completely finished.

   (Yixin Wang, 40hrs) (Hanchen Li, 40hrs)

Network (75hrs)

3. Create the *ClientMessageHandler* class that implements Java Socket to connect to a specific port listened to by *HostMessageHandler*.

   (Yuying Wang, 15hrs)

4. Make the *ClientMessageHandler* class be able to continuously receive the game state from *HostMessageHandler*.

   (Yuying Wang, 10hrs)

5. Make the *ClientMessageHandler* class be able to send user actions through *Stream* to *HostMessageHandler*.

   (Yuying Wang, 10hrs)

6. Create the *HostMessageHandler* class that implements *SocketServer* to listen on a specific port and be able to send the game state.

   (Somi Choi, 15hrs)

7. Make the *HostMessageHandler* class be able to receive actions from *ClientMessageHandler.*

   (Somi Choi, 10hrs)

8. Create test cases to check if the *HostMessageHandler* can communicate with up to 20 *ClientMessageHandlers* without a noticeable increase in latency.

   (Somi Choi, 10hrs)