Intro to Network Programming, 2025 Autumn

Final Project

Group 8: Big 2

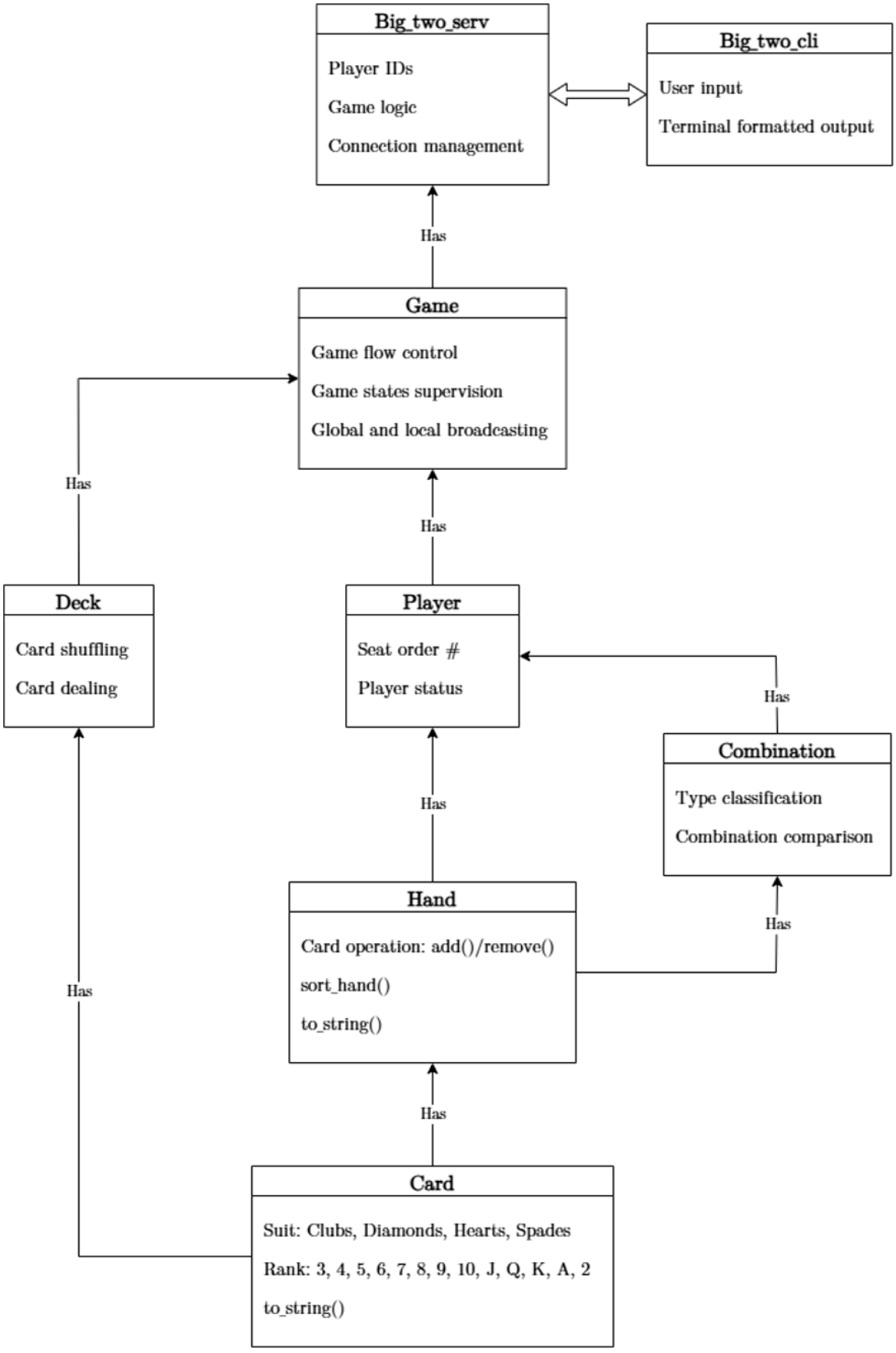
Report

Author:	112550119	Chia-Chi Cheng
	112550145	Yen-Yu Chen
Prof.:		Li-Hsing Yen

Part A: Abstract

Introduction	This project is based on a simple yet widespread card game, Big 2. In each game, there are four players, and each tries their best to clear their hands. Players must follow the previous combination unless a new round starts. The first player to clear their hand wins the game. In terms of network design, this project adopts a server-oriented architecture. The server takes charge of most of the work, such as establishing connections, configuring the games, controlling the gaming flow, broadcasting game states, and so on. Clients play a minimal logical role; they mainly react to connection-related events and maintain the server-player interactions, including display aesthetics.
Development environment	Game logic → Windows Network programming and final integration → Ubuntu All code is written in C++, though network-related functions are in C format.
Task allocation	112550119 Chia-Chi Cheng Network programming, final integration 112550145 Yen-Yu Chen Game logic, final integration

Part B: Implementation



△ The rough structure of our project.

Main Program: <Server_Client>

big_two_serv.cpp	include: all game logic units, network libraries 1. Initialize and manage room creation, client connection/disconnection, and the overall game flow. 2. Shuffle the players' seat order during connection. 3. Provide board information, including current hand to all players; provide legal actions for the player of the current turn to choose from. 4. Automatically skip the player who has passed in the current round or is disconnected. 5. Be responsible for handling all abrupt disconnections from any client at any stage of the game flow, regardless of gracefulness. (However, demo logic fails to handle when there are exactly 2 players left and the current player disconnects.)
<Server Implementation>	1. Use fork() to create new rooms when 4 clients join. 2. Use select() to detect disconnections in real time, while maintaining a play-turn polling logic.
big_two_cli.cpp	include: network libraries 1. Initialize connection with the server and handle server disconnection. 2. Parse the received messages and use ANSI escape sequences to clear the shell for consistent display.
<Client Implementation>	1. Use ANSI escape sequence to clear the shell for display.

Subprogram: <Game_logic>

card.cpp	1. Configure the basic object, card. 2. Support the conversion between card objects and strings. 3. Support the comparison of cards.
hand.cpp	include: card.cpp 1. Based on the card object, configure the object for **each** player, hand. 2. Support the basic operations, such as appending, removing, and sorting cards. 3. Support the conversion from hand objects to strings.
deck.cpp	include: card.cpp 1. Based on the card object, configure the object for **all** players, deck. 2. Set up the "true" deck, i.e., do dealing and shuffling.
combination.cpp	include: card.cpp, hand.cpp 1. Define and identify the valid combinations. 2. Define and implement the rules for comparing the combinations.
player.cpp	include: card.cpp, hand.cpp, combination.cpp 1. Set up the player information. 2. Manage the player status.
game.cpp	include: card.cpp, hand.cpp, deck.cpp, combination.cpp, player.cpp 1. Integrate all aforementioned subprograms. 2. Control the game flow. 3. Supervise the game state. 4. Broadcast.

Part C: Results

- Game initialization.

```
sent: alice
recv: Welcome to Big Two arena. You are the #3 player. Wait for other players!
```

  - ⊙ Clients are free to set their usernames.
  - ⊙ Each client is assigned a random seating order once the username is set.

- Broadcasting information.

```
--------------------------------
| cards left | #1:13 | #2:13 | #3:13 | #4:13 |
(Abbreviations: C for Club; D for Diamond; H for Heart; S for Spades.)
Player #2 "d" hand: 3C 3S 4S 6C 6S 7D 8C 8S 9H 9S 10S JH AH
First play.
Current round: 1
Current player: player#2
Last play: -
Players passed this round:
--------------------------------
If it's your turn press any key to play
Available combinations:
[0] 3C
[1] 3C 3S
Enter a combination to play or pass:
```

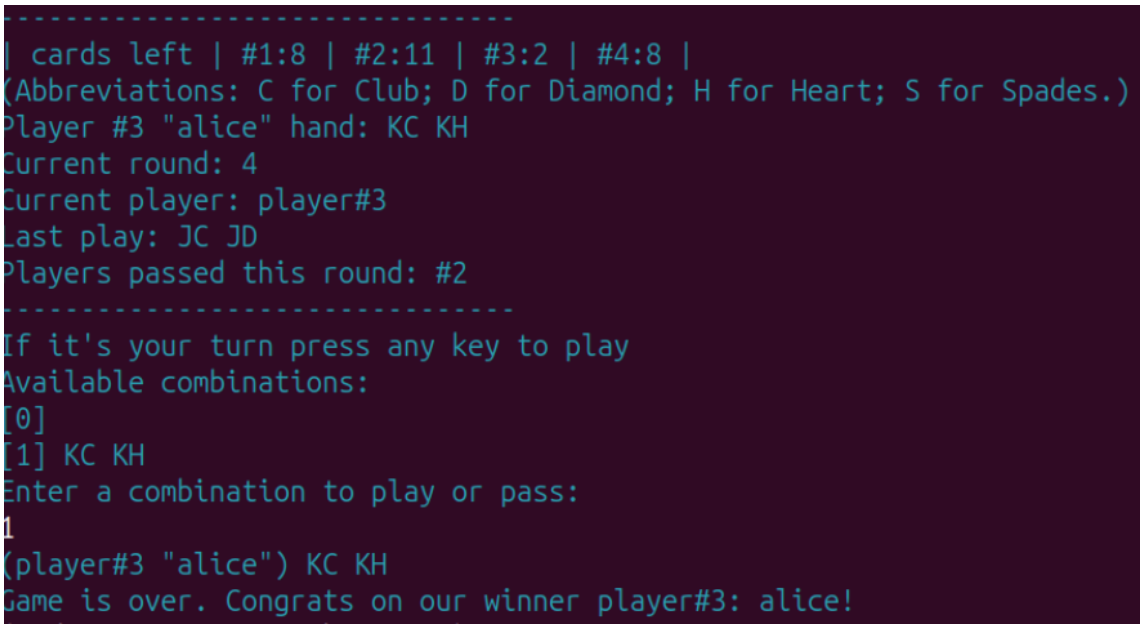
  - ⊙ Global broadcasting:
    - ▶ Number of remaining cards of each player. (Line 1)
    - ▶ Clarification of the abbreviation of four suits. (Line 2)
    - ▶ Round status. (Line 4-8)
  - ⊙ Local broadcasting:
    - ▶ The hand of the current player. (Line 3)
    - ▶ The available actions that the current player can choose from. (Line 11 onward)

- Disconnection.

```
(player #2 "d" left the room.)
```

  - ⊙ Once a player disconnects, the server broadcasts to all remaining players.
  - ⊙ Disconnected players will automatically call “Pass” in all subsequent rounds.

- End of game.



  - ⊙ Once a player clears the hand, the server broadcasts the winner. (Line 16)
  - ⊙ Once a winner is determined, the game ends and the server closes all player connections.

Part D: Conclusion

<Skills We Learnt>

1. Multi-process programming with an acceptor and a game room manager for each game room.
2. The use of TCP network functions to establish connections.
3. How to handle abrupt disconnections at different server states with the help of error codes.
4. Object-oriented programming with network compatibility.

<Future Improvements>

1. The client program should have been built with more local state tasks, such as generating card combinations.
2. In an advanced version, the client program should maintain a complete local game state, communicate with server, and present local state to user (as this is critical for real-time games and their UX).
3. Although not currently integrated into the final code, this project is able to be combined with the database functionality to support features such as data storage, user registration, identity verification (i.e., authentication), leaderboard display, and more.

Part E: Appendices and References

GitHub link: https://github.com/AlanCheng1000/Network-Programming-2025-Big-Two.git