Cards Against Humanity

Detailed Design

Team List:

Jamal Moon - jamalmoo@usc.edu

Corey Chen - coreyche@usc.edu

Peter Lu - peterjlu@usc.edu

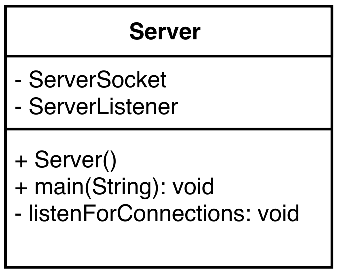
Justin Ku - kujustin@usc.edu

Vincent Espino - vlespino@usc.edu

Anush Kadoyan - kadoyan@usc.edu

Cards Against Humanity - Detailed Design

Server Class Diagrams



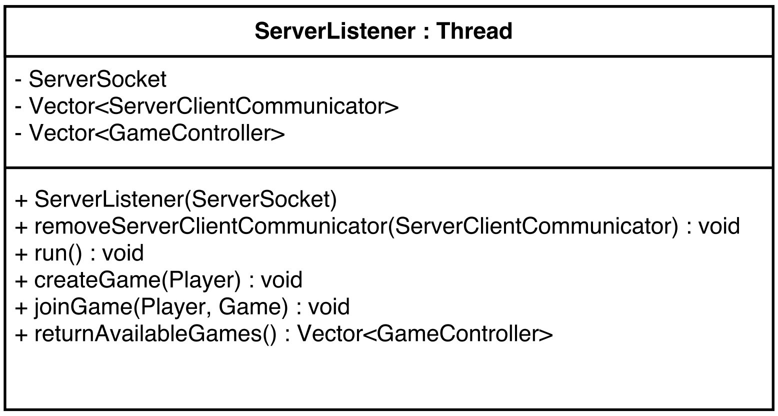
Server

- ServerSocket
- ServerListener

+ Server()
+ main(String): void
- listenForConnections: void

Server - Sets ServerSocket, creates ServerListener.

Will currently try ports until reaching an available port(subject to change).

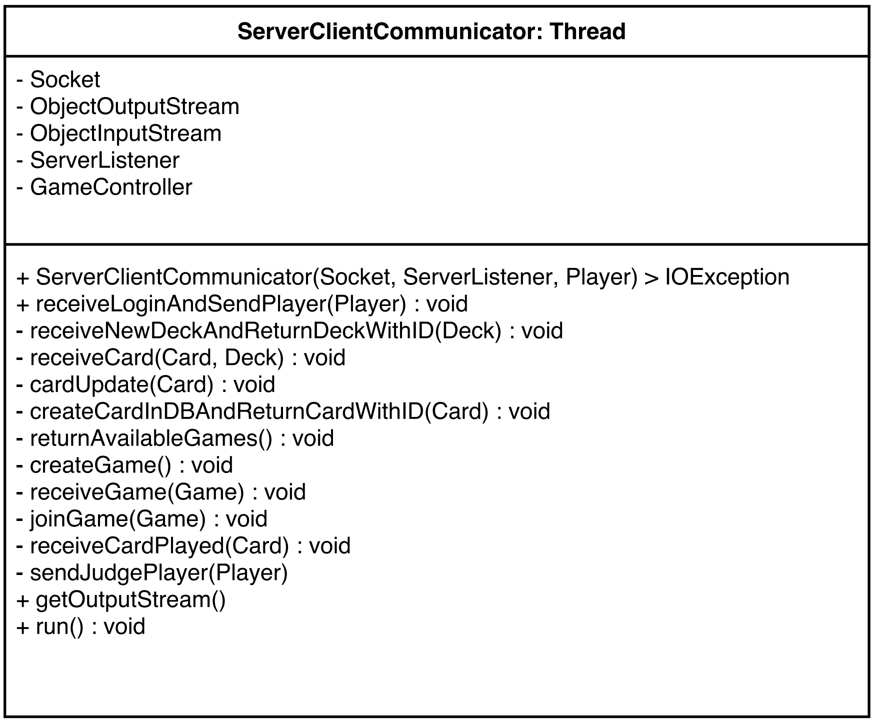


ServerListener : Thread

- ServerSocket
- Vector<ServerClientCommunicator>
- Vector<GameController>

+ ServerListener(ServerSocket)
+ removeServerClientCommunicator(ServerClientCommunicator) : void
+ run() : void
+ createGame(Player) : void
+ joinGame(Player, Game) : void
+ returnAvailableGames() : Vector<GameController>

ServerListener - Listens for incoming connection on ServerSocket.
Once a connection is established with a client and the log-in is verified(or is guest),
the ServerListener creates a new socket, and passes it to a ServerClientCommunicator.



ServerClientCommunicator: Thread

- Socket
- ObjectOutputStream
- ObjectInputStream
- ServerListener
- GameController

+ ServerClientCommunicator(Socket, ServerListener, Player) > IOException
+ receiveLoginAndSendPlayer(Player) : void
- receiveNewDeckAndReturnDeckWithID(Deck) : void
- receiveCard(Card, Deck) : void
- cardUpdate(Card) : void
- createCardInDBAndReturnCardWithID(Card) : void
- returnAvailableGames() : void
- createGame() : void
- receiveGame(Game) : void
- joinGame(Game) : void
- receiveCardPlayed(Card) : void
- sendJudgePlayer(Player)
+ getOutputStream()
+ run() : void

ServerClientCommunicator - Holds a single connection with a GameClient. It sends the PlayerClientListener the Player.

Supports: receiving a new deck from the PlayerClientListener and returning the PlayerClient with the Deck with the DB ID;

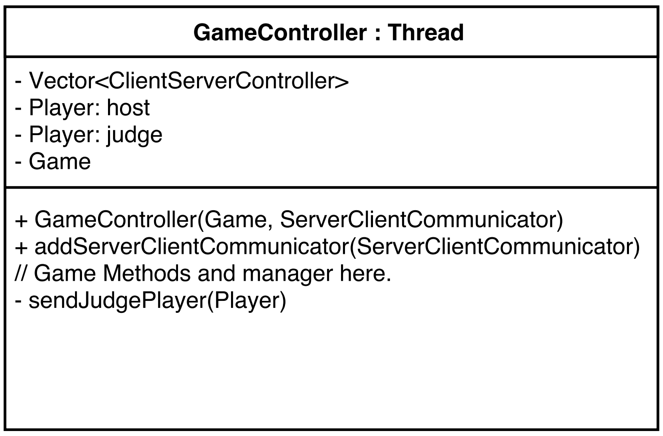
receiving a Card from PlayerClientListener -- if it has an ID, then update the Card in the DB,

else create a new Card in the DB and return the Card with DB ID.

returnAvailableGames, createGame, and joinGame contact the ServerListener to handle.

joinGame is called if GameController is null and also sets the GameController.

if the GameController is not null and a Game was accepted, then it prepares to accept a Card.


GameController : Thread
- Vector<ClientServerController>
- Player: host
- Player: judge
- Game
+ GameController(Game, ServerClientCommunicator)
+ addServerClientCommunicator(ServerClientCommunicator)
// Game Methods and manager here.
- sendJudgePlayer(Player)

Client Class Diagrams

PlayerClient

+ field: type

+ PlayerClient()
+ main(String) : void

PlayerClientListener : Thread

- Socket
- ObjectInputStream
- ObjectOutputStream
- PlayerManager

+ PlayerClientListener(PlayerManager, Socket)
- initializeVariables() : boolean
+ sendNewDeck(Deck)
+ sendCard(Card, Deck)
+ sendGameSignal(Game)
+ sendSelectedCard(Card)

PlayerClientListener - Listens and communicates with ServerClientListener.
It supports sending cards and sending the selected card after sending the game to signal a game move.

LoginPanel : JPanel

+ long : serialVersionUID
- JTextField : loginTF, passwordTF
- JButton : signInButton, createAccountButton, guestButton
- JLabel : titleLabel, loginLabel, passwordLabel
- String : username, password

+ LoginPanel()
+ initializeVariables() : void
+ createGUI() : void
+ addActionListener() : void
+ createAccount() : void
+ accountLogin(username, password) : void
+ guestLogin() : void

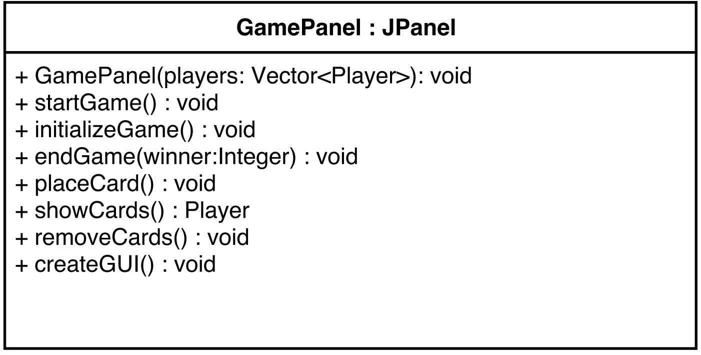
LoginPanel - Contains the interface for logging in or creating a user account.
Users can either enter a username and password to log in, log in as a guest, or create an account.
If the user wants to create an account, a window will pop up, prompting the user for a username and password.
Otherwise, the user can log in through an account or as a guest.

GameLobby : JPanel

+ long : serialVersionUID
- JLabel : titleLabel
- Button : editDeckButton, createGameButton, connectButton
- JScrollPane : jsp
- JTable : gamesTable
- Player : currentPlayer

+ initializeVariables() : void
+ createGUI() : void
+ addActionListener() : void
+ createGame(Player host) : void
+ connectToGame(Player player) : void
+ editDeck(Player player) : void

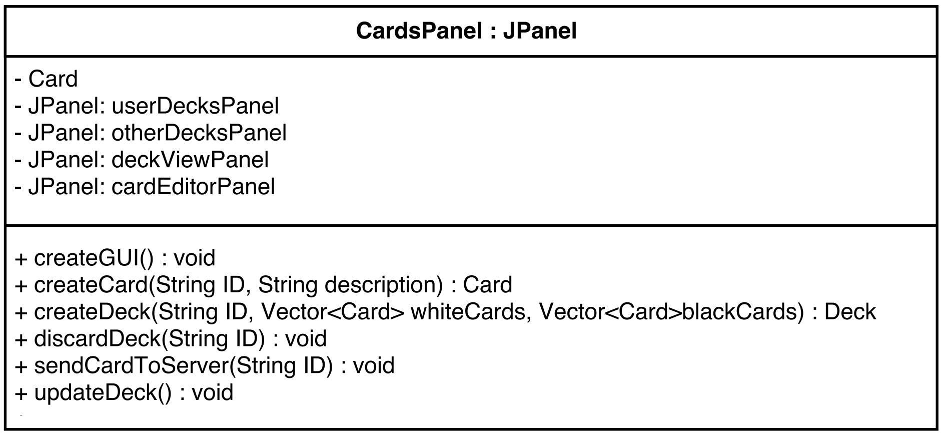
GameLobby - Holds the interface for the lobby screen, seen immediately after logging in.
It contains JButtons to view/edit the deck, create a new game, and connect to a pre-existing game.
All of the current games are listed in a JTable with a scroll bar.
The lobby could also contain a way to view and chat with friends or other users online.



GamePanel : JPanel

+ GamePanel(players: Vector<Player>): void
+ startGame() : void
+ initializeGame() : void
+ endGame(winner:Integer) : void
+ placeCard() : void
+ showCards() : Player
+ removeCards() : void
+ createGUI() : void

GamePanel - Holds all the graphics associated with the game.
It shows the black card picked by the judge and shows the cards (buttons) of the current player on the bottom of the screen.
When the players select cards, those cards will appear in the middle of the screen next to the black card.

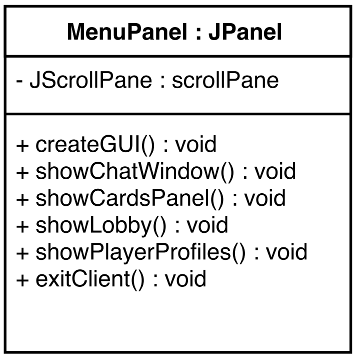


CardsPanel : JPanel

- Card
- JPanel: userDecksPanel
- JPanel: otherDecksPanel
- JPanel: deckViewPanel
- JPanel: cardEditorPanel

+ createGUI() : void
+ createCard(String ID, String description) : Card
+ createDeck(String ID, Vector<Card> whiteCards, Vector<Card>blackCards) : Deck
+ discardDeck(String ID) : void
+ sendCardToServer(String ID) : void
+ updateDeck() : void

CardsPanel - holds the graphics for all card and deck editing.
The panel will allow users to view currently available decks as well as the contents of the decks.
In this panel, users will also be able to create their own decks, create and add cards to decks, as well as discard both decks and cards.



MenuPanel : JPanel

- JScrollPane : scrollPane

+ createGUI() : void
+ showChatWindow() : void
+ showCardsPanel() : void
+ showLobby() : void
+ showPlayerProfiles() : void
+ exitClient() : void

MenuPanel - shown when a user clicks on the menu button on any of the screens.
It shows a list/dropdown of options such as “Go back to Lobby,” “Log out” and “Exit.”

<u>Resources</u>

Player
+boolean:isJudge
+int:ID
+string:name
+Deck
+int:numPoints
+string:password
//Lifetime stats for profile

Player - Holds all the information about a player, including their total points, decks, name, and whether they are a judge or not in the game they are playing.

Game
+int:ID
+Vector<Player>
+Vector<Card>: turnCards
+int:turn
+Player:Host
+Player:Judge

Game - The class that has the main aspects of the game including cards and players.

Deck
+int:ID
+Vector<Card>: whiteCards
+Vector<Card>:blackCards
+addCardToDeck(Card): void

Deck - Has vectors of white and black cards and the ability to add a card to itself.

Card
+int:ID
+string:description
+editDescription(): void
-Image: bImage
Card - Has the information about each card such as the content, background image, and the ID of the card.

Database Schema

Player_Table

This table contains essential information for the Player to function. Ideally, the password attribute will be encrypted upon creation of the account.

int	id
string	user_name (or email)
string	password
string	name
datetime	time_created

Decks_Table

The table for decks. It is short because the Cards table will contain all the information (and point to the Deck).

int	id
string	deck_name
datetime	time_created

Players_Decks_Table

This is a join table; this allows us to find which Decks that a Player has.

int	id
int	deck_id

int	player_id
datetim e	time_created

Cards_Table

This is sufficient to know the Card's text/phrase, if it is white or black, and which Deck it belongs to.

int	id
string	text
boolean	is_white
int	deck_id
datetim e	time_created
