Communication Protocol (TUI version) (RMI)

Legend:

> : something is printed or typed

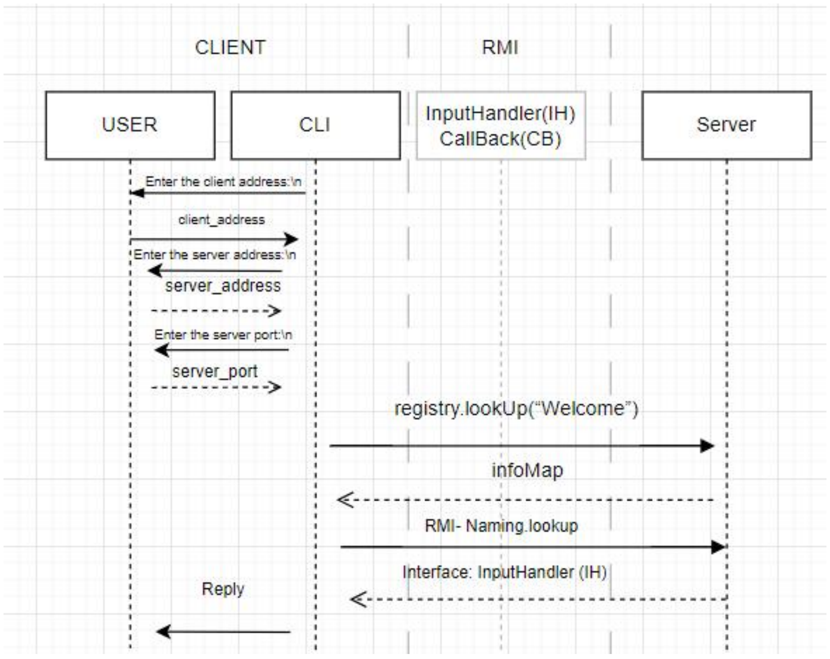
# : method invocation(static and not)

| : the value or the possible values of a variable are shown

CB : (RMI) CallBack interface (mostly used by server)

IH : (RMI) InputHandler interface (mostly used by client)

Connection Phase


CLIENT
RMI
USER
CLI
InputHandler(IH) CallBack(CB)
Server
Enter the client address:\n
client_address
Enter the server address:\n
server_address
Enter the server port:\n
server_port
registry.lookUp("Welcome")
infoMap
RMI- Naming.lookup
Interface: InputHandler (IH)
Reply

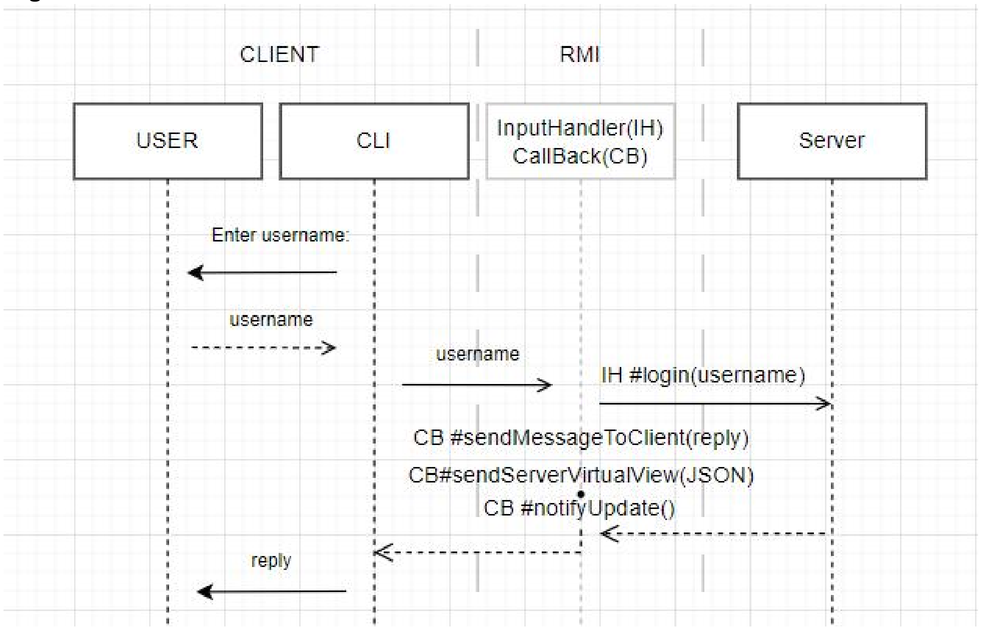
CLI - User Interactions:
CLI #askServerInfo()
CLI displays > Enter the client address: \n
User types > client_address\n
CLI displays > Enter the server address: \n
User types > server_address\n
| server_address: localhost,...,192.168.20.23,....
CLI displays > Enter the server port: \n
User types > server_port\n
| server_port: 1234,8807,...

RMI connection (client)
# registry.lookUp("Welcome")
RMI connection (server)
Return : infoMap -> which contains the inputHandler interface stub

RMI connection (client)
# Naming.lookup("rmi://"+infoMap.get("address")+":"+infoMap.get("port")+"/"+serverInfo.get("root"))
RMI connection (server)
Return : inputHandler stub

CLI displays > reply\n
| reply: Connection Successful, Connection Failed

Login Phase



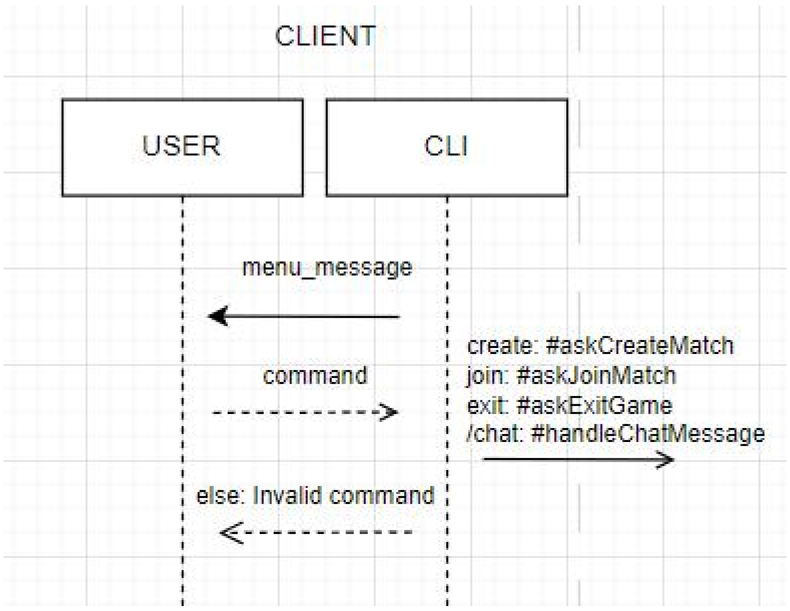
CLI - User Interactions:
CLI #askLogin()
CLI displays > Enter username:\n
User types > username\n
| username: [any name]

Login request (client)
IH #login(username)

Login reply (server)
CB #sendMessageToClient(reply)
CB #sendServerVirtualView(JSON)
CB #notifyUpdate()

CLI displays > reply\n
| reply:
| (if true) Login successful. Hi [username]
| (if false) Access denied. This username is already taken, please enter a new one...

Game Menu Phase

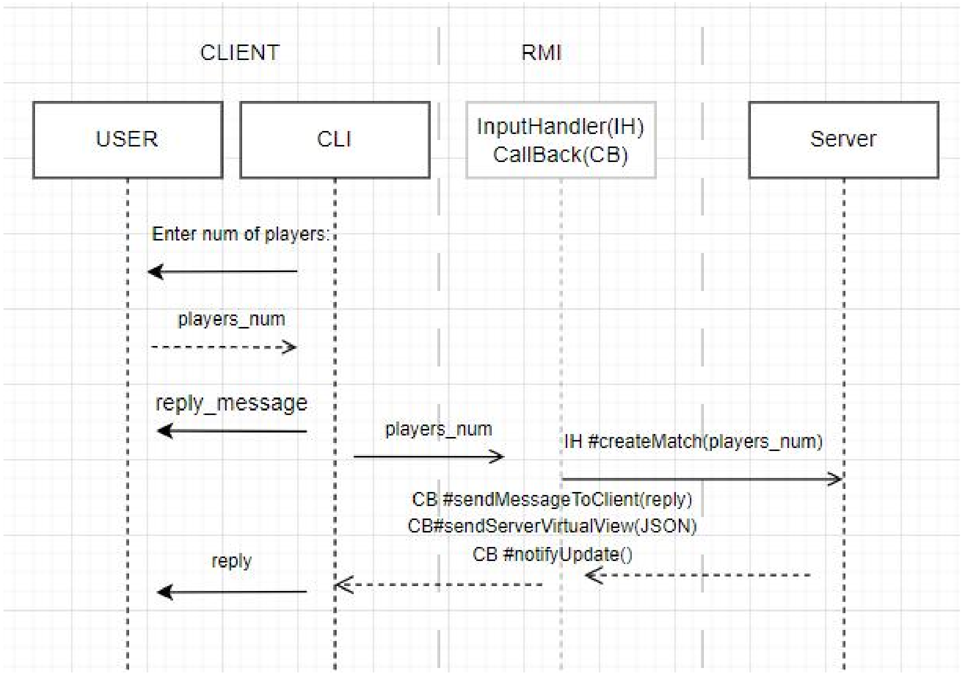


CLI - User Interactions:
CLI #askMenuAction()
CLI displays > menu_message\n
|menu_message:
|		Menu option:
|		create --> Create a new match
|		join --> Join a match
|		exit --> Exit game
|		To send a message to a online player type '/chat[nickname]' followed by your message in the console.
|		Enter the option number you wish to select (1,2 or3): \n

User types > command\n
| command: create,join,exit, /chat[[nickname]]:[text]

If [command] is 'create':
CLI #askCreateMatch() --->|Go to Create Match Phase|
If [command] is 'join':
CLI #askJoinMatch() --->|Go to Join Match Phase|
If [command] is 'exit':
CLI #askExitGame() --> |Go to Exit Game Phase|
If [command].substring(0,4) is '/chat':
CLI #handleChatMessage([command]) --> |Go to Chat Phase|
Else:
CLI displays > Invalid command. Try again.\n

Create Match Phase



CLI - User Interactions:
CLI #askCreateMatch()
CLI #askMaxSeats()
CLI displays > Please select the max number of players for this match(2,3 or 4):\n
User types > players_number\n
|players_number: 2,3,4
CLI displays > reply_message\n
|reply_ message:
| (positive) Selected [players_number] seats.\n
| (negative) Invalid number!\n , Invalid input!\n

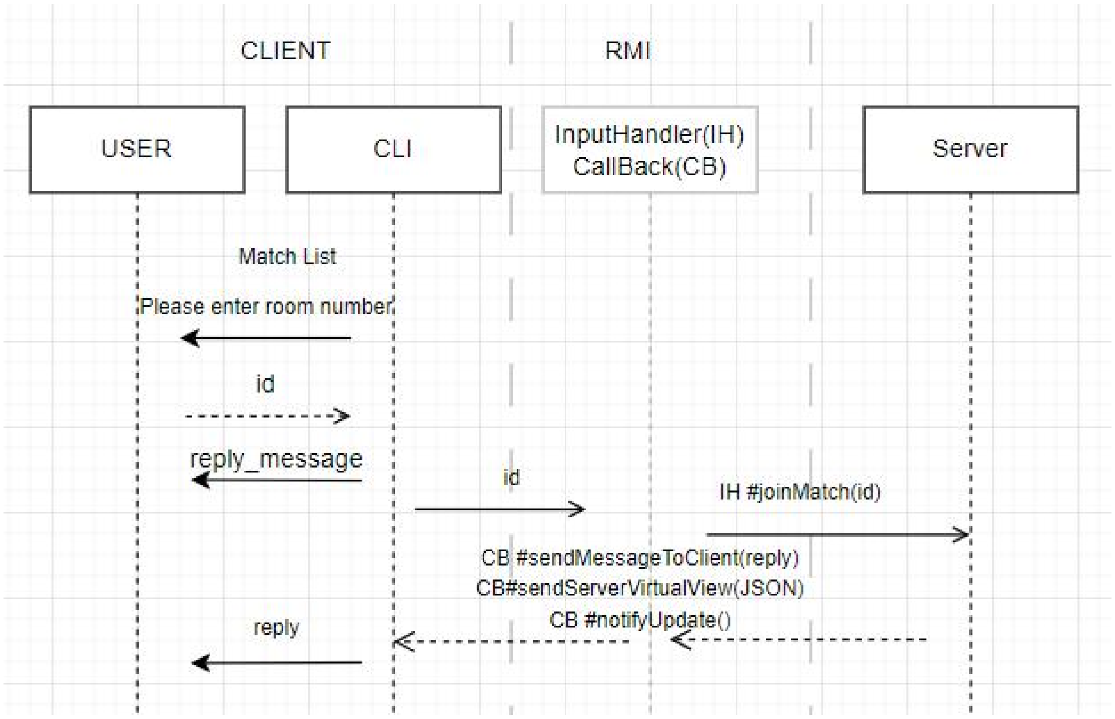
Create Match request (client)
IH #createMatch(players_number)

Create Match reply (server)
CB #sendMessageToClient(reply)
CB #sendServerVirtualView(JSON)
CB #notifyUpdate()

|reply:
|(negative)The player already exists in a match. In order to create a new match, you need to abandon the current one.
| Do you wish to continue?\n
|(negative) Exceeded player number limit. Try again.\n
|(positive) Match with [player_number] seats created. MatchID: [matchID]\n

CLI displays > reply\n

Join Match Phase



CLI - User Interactions:
CLI #askJoinMatch()
CLI displays > Match_List\n
|Match_List: a list of all matches in the game(from 0 to matchList.size()-1)
| Example of display: Room N.[matchID] - [GameState] ([nPlayers]/[maxSeats])
| GameState: WaitingPlayers, Ready, GameGoing, LastRound, Closed
| nPlayers: number of players in the room
| maxSeats: max number of players for the match
CLI displays > Please enter the room number:\n
User types > id\n
|id: from 0 to (matchList.size()-1)
CLI displays > reply_message\n
|reply_ message:
| (positive) Selected Room [id].\n
| (negative) Invalid number!\n , Invalid input!\n

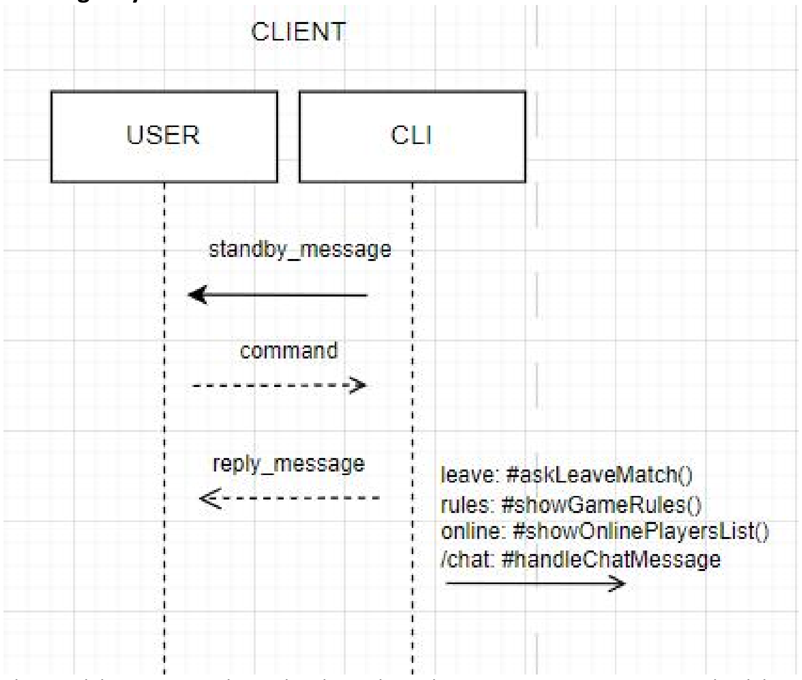
Join Match request (client)
IH #joinMatch(id)

Join Match reply (server)
CB #sendMessageToClient(reply)
CB #sendServerVirtualView(JSON)
CB #notifyUpdate()

|reply:
|(negative) The room selected does not exist. Try again.\n
|(negative|when the player try to join after connection loss) The player does not exist in any room. Try to create a new one.\n
|(negative) The room is full\n
|(positive) [username] joined the match [matchID]\n

CLI displays > reply\n

Waiting Players Phase



The match has not started yet. The players have the option to use some commands while waiting.

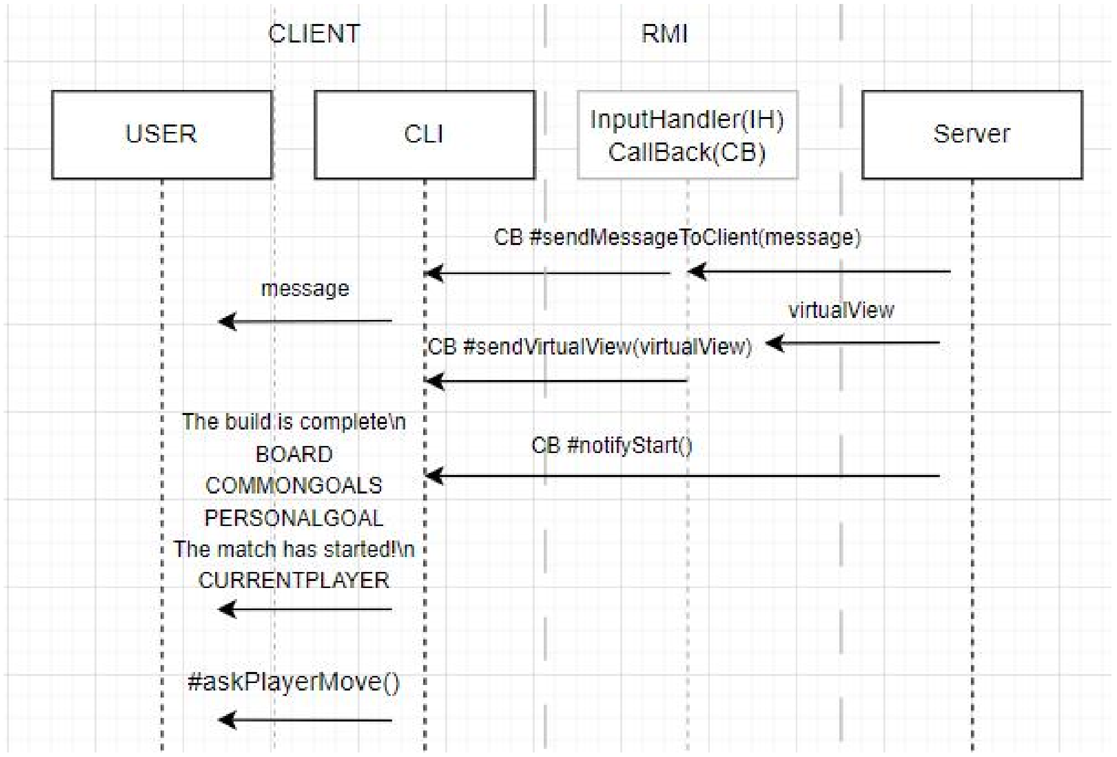
CLI - User Interactions:

```
CLI #askWaitingAction()
CLI displays > standby_message\n
| standby_message:
|       The match has not started yet. Waiting for more players to join... [maxSeats - nPlayers] seats available.
|       These are the commands available:
|       leave --> Leave Match
|       rules --> Read Game Rules
|       online --> Show Online Players
|       To send a message in the Match Chat type '/chat ' followed by your message in the console.
|       To send a message to a online player type '/chat[nickname]' followed by your message in the console.
|       Enter the number of the command you wish to execute (1 to 3):\n

User types > command\n
| coommand: leave,rules,online, /chat [message],/chat[[nickname]] [message]
CLI display > reply_message\n
| reply_message:
| (positive) [command] command selected  \n
| (negative) Invalid command!\n , Invalid input!\n

If [command] is 'leave':
CLI #askLeaveMatch() --> |Go to Leave Match Phase|
If [command] is 'rules':
CLI #showGameRules() --> |Go to Show Game Object Phase|
If [command] is 'online':
CLI #showOnlinePlayersList()
If [command].substring(0,4) is '/chat':
CLI #handleChatMessage([command]) --> |Go to Chat Phase|
```

Match Initialization Phase



When a player joins and completes the room,then the match starts its initialization automatically.

Initialization Notification (server)

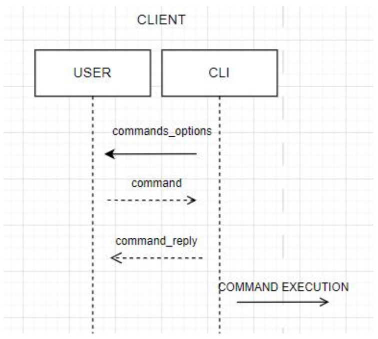
CB #sendMessageToClient(message)
|message: The match is about to start. Building game board...
CLI displays > message\n

CB #sendVirtualView(virtualView)
--Note: Client is going to receive a virtualization of the game model
CB #notifyStart()

```
CLI #showMatchSetup()
 {
        CLI displays > The build is complete.
        CLI #showBoard()
        CLI displays > [Board]
        CLI #showCommonGoals()
        CLI displays > [Common Goals]\n
        CLI #showPersonalGoal()
        CLI displays > [Personal Goal]\n
        CLI displays > The match has started!\n
        CLI #showCurrentPlayer()
        If [current_player]==[username]:
        CLI displays > Hey [username]! It's your turn\n
        Else:
        CLI displays > It's [current_player]'s turn\n
        CLI #askPlayerMove() --> |Go to Gameplay Phase|
}
```

Gameplay Phase

CLI - User Interactions:

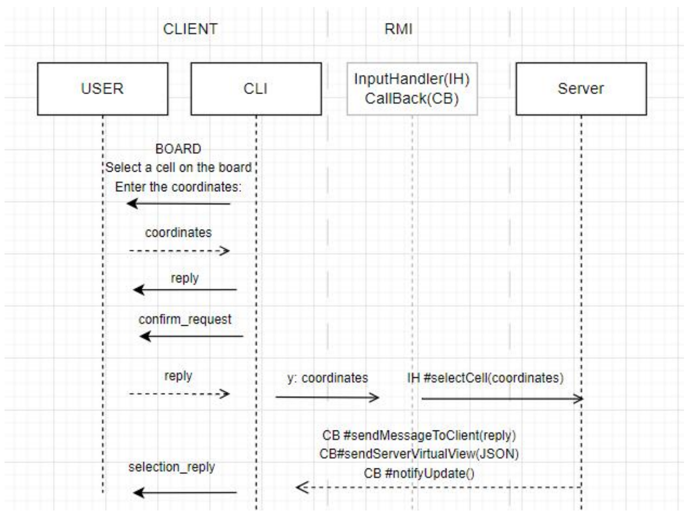


CLI #askPlayerMove()
CLI displays > commands_options\n
| commands_options:
|		What do you wish to do? These are the commands available:
|		select --> Select a Cell
|		deselect --> Deselect Items
|		insert --> Insert your items in the shelf (at least one item selected)
|		show --> Show Game Object(Hand,Goals,Board,Shelf,...)
|		leave --> Leave Match
|		exit --> Exit Game
|		To send a message in the Match Chat type '/chat ' followed by your message in the console.
|		To send a message to a online player type '/chat[nickname] ' followed by your message in the console.
|		Enter the command you wish to use: \n
User types > command\n
|command: select, deselect, insert, pgoal, cgoal, shelf, board, stats, help,rules, end,timer,leave,exit,
|		/chat [message],/chat[nickname] [message]
CLI displays > command_reply\n
|command_reply:
|(positive) Executing command...
|(negative) This command is not valid.\n, This command is not allowed right now. Wait your turn...\n,
The conditions to use this command are not respected. Try again.\n

COMMAND EXECUTION (CLI):
If [command] is 'select' :
CLI #askSelection() --> |Go to Selection Phase|
If [command] is 'deselect':
CLI #askDeselection() --> |Go to Deselection Phase|
If [command] is 'insert':
CLI #askInsertion() --> |Go to Insertion Phase|
If [command] is 'show':
CLI #askShowObjects() --> |Go to Show Game Object Phase|
If [command] is 'leave':
CLI #askLeaveMatch() --> |Go to Leave Match Phase|
If [command] is 'exit':
CLI #askExitGame() --> |Go to Exit Game Phase|
If [command].substring(0,5) is '/chat:' :
CLI #handleChatMessage() --> |Go to Chat Phase|
If [command.substring(0,5) is '/text:' :
CLI #handleText() --> |Go to Chat Phase|

When the COMMAND EXECUTION is completed:
CLI #askPlayerMove() --> |Go to Gameplay Phase|

Gameplay Phase > Selection Phase



CLI - User Interactions:

```
CLI #askSelection()
{
CLI #showBoard()
CLI displays > [Board]\n Select a cell on the board.\n
CLI #askCoordinates()
{
        CLI displays > Enter the coordinates: \n
        User types > coordinates\n
        |coordinates: ([row],[column])
        CLI displays > reply\n
        |reply:
        |(negative) Invalid numbers., Invalid Input.
        |(positive) You have selected ([row],[column])
}
(end of: #askCoordinates) --> return:
[coordinates]=([row],[column])
CLI displays> confirm_request\n
|confirm_request:
| Now you can confirm your choice(y),cancel your choice(n), retry again(r), see a Game Object(Board,Shelf,Goals,...)(show)?
User types > reply\n
|reply: y,n,r, show

If [reply] is 'n':
|Exit Selection Phase|
CLI #askPlayerMove() --> |Go to Gameplay Phase|
If [reply]is 'r':
CLI #askCoordinates()
If [reply] is 'show':
CLI #askShowObjects() --> |Go to Game Object Phase|

If [reply] is 'y':
```

Selection request (client)

```
IH #selectCell([row],[column])
```

Selection reply (server)

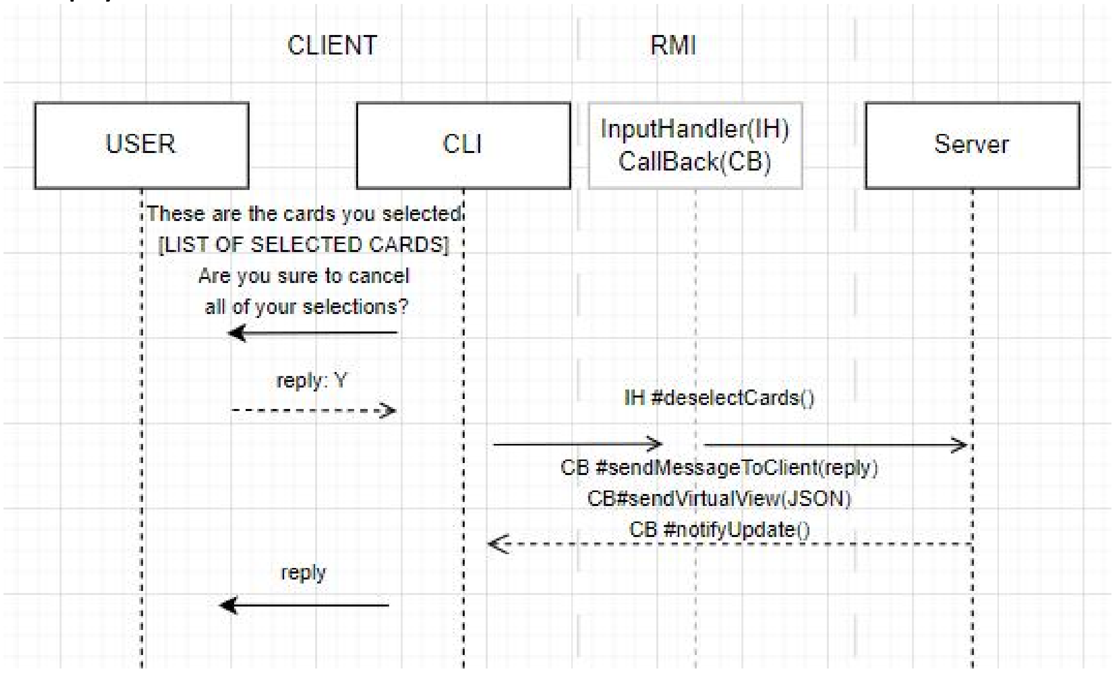
```
CB #sendMessageToClient(selection_reply)
CB #sendServerVirtualView(JSON)
CB #notifyUpdate()
|selection_reply:
|(positive) Selection successful!
|(negative) Selection failed: [reason]
        |reason:
        |(empty cell) You chose an empty cell
        |(already selected) You have already selected this cell
        |(hand limit) You reached the limit of items you could pick
        |(out of boundary) You selected an illegal cell
        |(no free side) The cell was not selectable. Pick an item which has a free side.
        |(no orthogonal)The cell was not selectable.Pick an item which is adjacent and in line with the other selected items.
CLI displays > selection_reply\n
CLI displays > Item selected: \n
CLI #showCell([coordinates])

CLI displays > Do you wish to select again? Y/N \n
User types > reply\n
|reply: Y,N

If Y:
Loop #askSelection
If N:
Continue

CLI #askPlayerMove() --> |Go to Gameplay Phase|
}
(end of: #askSelection)
```

Gameplay Phase > Deselection Phase



CLI - User Interactions:
CLI #askDeselection()
{
CLI displays > These are the cards you have selected:\n
CLI #showHand()
CLI diplays > [List of selected items]
CLI displays > Are you sure to cancel all of your selections? y/n
User types > reply\n
| reply: y , n

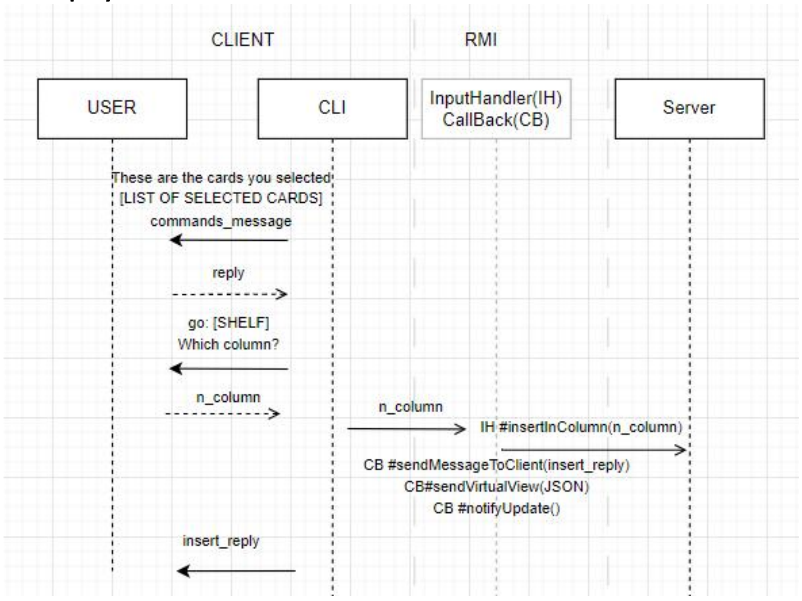
Deselection request (client)
IH #deselectCards()

Deselection reply (server)
CB #sendMessageToClient(reply)
CB #sendServerVirtualView(JSON)
CB #notifyUpdate()

| reply :
|(positive) Deselection successful.
|(negative) Deselection failed.
|(negative|hand.size()=0) You didn't select any card yet. Deselection is not necessary.\n
CLI displays > reply\n

CLI #askPlayerMove() --> |Go to Gameplay Phase|
}

Gameplay Phase > Insertion Phase



CLI -User Interactions:
CLI #askInsertion()
{
CLI displays > These are the cards you have selected:\n
CLI #showHand
CLI displays > [List of selected cards]
CLI displays > commands_message\n
|commands_message:
| These are the commands available:
| sort --> change the order of your cards
| show --> look at game board objects
| go --> go directly to insertion
| Type your command:
User types > reply\n
|reply: sort, show, go

If [reply] is 'sort':
CLI #askSort() --> |Go to Sort Phase|
If [reply] is 'show':
CLI #askShowObjects() --> |Go to Show Game Object Phase|
If [reply] is 'go':
Continue

CLI #showShelf()
CLI displays > [Shelf]\n In which column would you like to insert the items? 1/2/3/4/5\n
User types > n_column\n
| n_column: 1,2,3,4,5

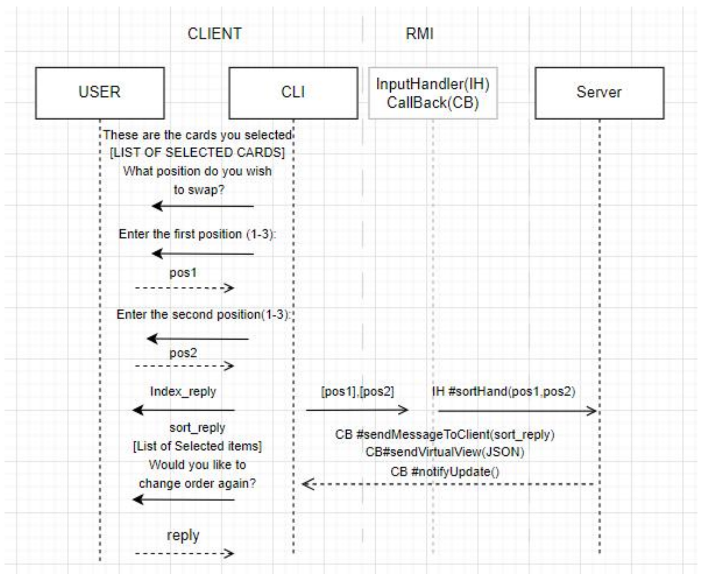
Insertion request (client)
IH #insertInColumn(n_column)

Insertion reply (server)
CB #sendMessageToClient(insert_reply)
CB #sendVirtualView(JSON)
CB #notifyUpdate()

|insert_reply:
|(positve) Insertion Successful
|(negative|no space in column) There is not enough slot available in this column. Try again.
CLI displays > reply\n

CLI #askPlayerMove() --> |Go to Gameplay Phase|
}

Gameplay Phase > Sort Phase



CLI #askSort()
{
CLI displays > These are the cards you have selected:\n
CLI #showHand()
CLI displays > What position do you wish to swap?\n
CLI #askIndex()
{
CLI displays > Enter first position (1-3):\n
User types > pos1\n
CLI displays > Enter second position (1-3):\n
User types > pos2\n
|pos1&2: 1, 2, 3

CLI displays >index_reply
|index_reply:
|(positive) Position selected: [position1,position2]
|(negative|pos1=pos2) You selected the same position.
|(negative|pos<1 or pos>3) Not valid input. Try again.

}
(end of: #askIndex())-->return: [pos1,pos2]

CLI displays > Would you like to Continue(press enter), to Retry (retry), to exit sort(cancel)
User types > reply
|reply : 'press enter',retry, cancel

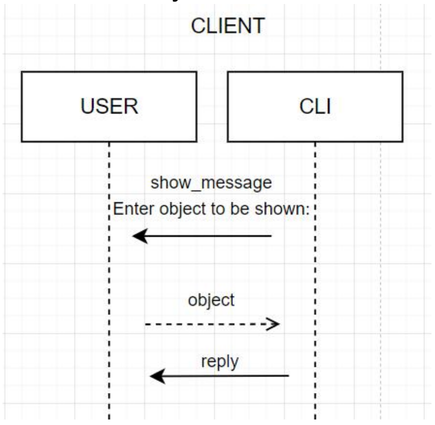
Sorting request (client)
IH #sortHand([pos1],[pos2])

Sorting reply (server)
CB #sendMessageToClient(sort_reply)
CB #sendVirtualView(JSON)
CB #notifyUpdate()

|sort_reply:
|(positive) Order changed.
|(negative) Index out of border
|(negative) Not enough cards in your hand
CLI displays > sort_reply\n
CLI #showHand()
CLI displays > [List of selected items]
CLI displays > Would you like to change order again? Y/N\n
User types > reply\n
|reply: Y,N

If Y:
Loop #askSort()
If N:
Continue
}

Show Game Object Phase



--**Note** : many of these objects are virtualized and already sent to CLIENT, so most of the time there no need to contact the SERVER for this task.

CLI - User Interactions:

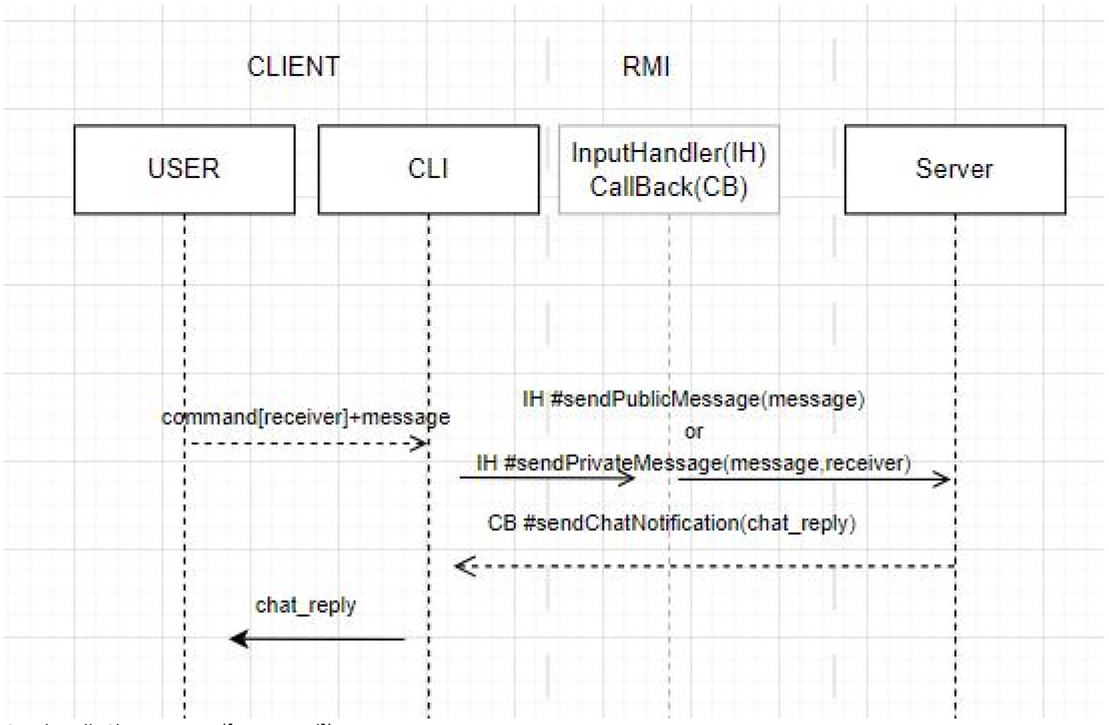
```
If #askShowObjects() is called:
CLI #askShowObjects()
{
        CLI displays > show_message\n
        |show_message:
        |        List of Objects:
        |        open --> Show Players Chat List
        |        hand --> Show selected items
        |        pgoal --> See your Personal Goal
        |        cgoal --> See Common Goals
        |        shelf --> See your shelf and the insertion limit
        |        board --> See Living Room Board
        |        stats --> See Players Stats
        |        rules --> See Game Rules
        |        end --> Show if the Endgame Token is taken (If it is, then it's the last round)
        |        online --> Show Online Players
        |        Enter the object you wish to be shown:
        User types > object\n
        |object : hand, pgoal,cgoal,shelf,board,stats,rules,end,timer

        If [object] is 'hand':
        CLI #showHand()
        If [object] is 'pgoal':
        CLI #showPersonalGoal()
        If [object] is 'cgoal':
        CLI #showCommonGoal()
        If [object] is 'shelf':
        CLI #showShelf()
        If [object] is 'board':
        CLI #showBoard()
        If [object] is 'stats':
        CLI #showPlayersStats()
        If [object] is 'rules':
        CLI #showGameRules()
        If [object] is 'end':
        CLI #showEndgameToken()
        If [object] is 'online':
        CLI #showOnlinePlayers()
        If [object] is 'open:
        CLI #showPlayersChat()
}
```

```
If generic #show[Object]() method is called:
CLI displays > reply\n
|(positive) [OBJECT]
|(negative) The [object] cannot be found
```

```
If #showGameRules() is called:
CLI displays > [FULLGAMERULE_TEXT]
```

Chat Phase


CLIENT
RMI
USER
CLI
InputHandler(IH)
CallBack(CB)
Server
command[receiver]+message
IH #sendPublicMessage(message)
or
IH #sendPrivateMessage(message,receiver)
CB #sendChatNotification(chat_reply)
chat_reply

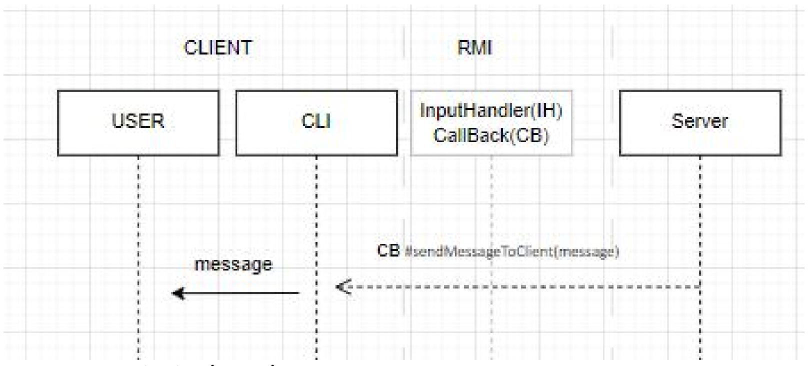
CLI #handleChatMessage([command])
{
Chat request (client)
IH #sendPublicMessage([command].substring(5,[command].length()))
Or
IH #sendPrivateMessage([command].substring(5,[command].length()),receiver)

Chat reply (server)
CB #sendChatNotificationt(chat_reply)
|chat_reply:
|(positive|to_player) Message sent to [nickname]
|(positive|to_group) [Match Chat History]
|(negative|to_player) Message didn't reach [nickname] ,
| The player [nickname] doesn't exist ,
| The player [nickname] is not online.
|(negative|to_group) Message was not sent.

CLI displays > chat_reply\n
}

Extra: Sending Message To Player (server)
CB #sendChatMessage(filtered_text)
|filtered_text:
| [sender_nickname] says:
| [command].substring(7+[nickname].length,[command].length())
CLI displays > filtered_text\n

Server Announcement



Server communication (server)
CB #sendMessageToClient(message)
|message:
|[nickname] has left the match
|[nickname] has left the game
|[nickname] has selected [CurrentPlayerHand]
|[nickname] has inserted items in the column [nColumn]
|[nickname] achieved [CommonGoal]
|[nickname] ended his turn.

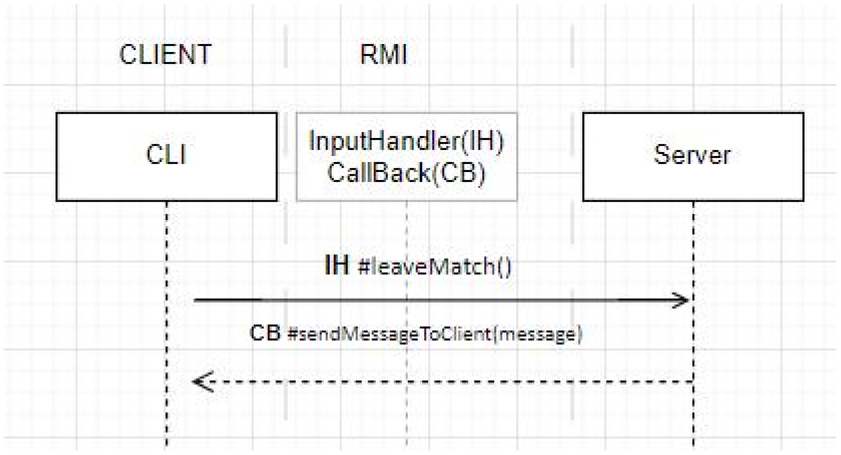
CLI displays > message\n

Gameplay Phase > Game Over Phase

Server communication (server)
CB #sendMessageToClient(message)
|message:
| GAME OVER
|---
|(If there is a winner) Congratulation! The winner is [winner]
CLI displays > message
CLI #showPlayerStats()
CLI displays > [Player Stats]

Then the players will all be removed from the match and match will be close and deleted.
CLI #askMenuAction() --> |Go to Menu Phase|

Leave Match Phase

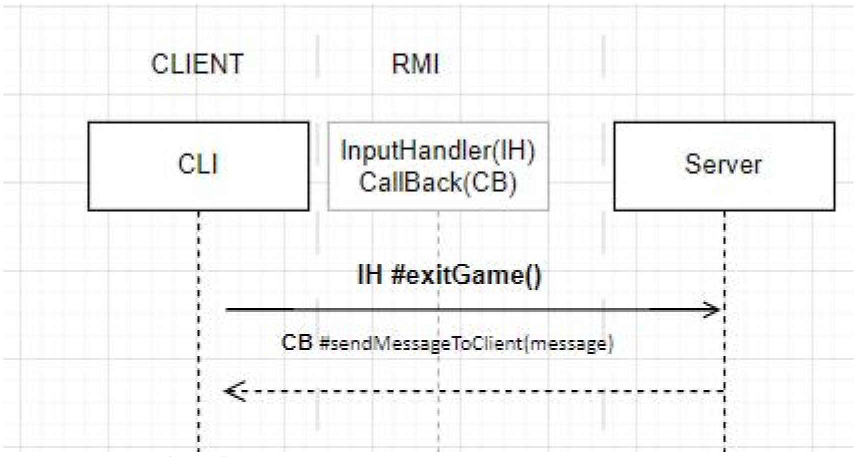


Leave match request (client)
IH #leaveMatch()

Leave match reply (server)
CB #sendMessageToClient(reply)
|reply : You left the match
CLI displays > reply\n

CLI #askMenuAction() --> |Go To Menu Phase|

Exit Game Phase



Exit Game request(client)
IH #exitGame()

Players Data will be removed from the server(nickname, ...)

Exit Game request(server)
CB #sendMessageToClient(reply)
|reply : Bye! See you soon.

CLI displays > reply\n
System.exit(0)

Communication Protocol (TUI version) (TCP)

Legend:

> : something is printed or typed

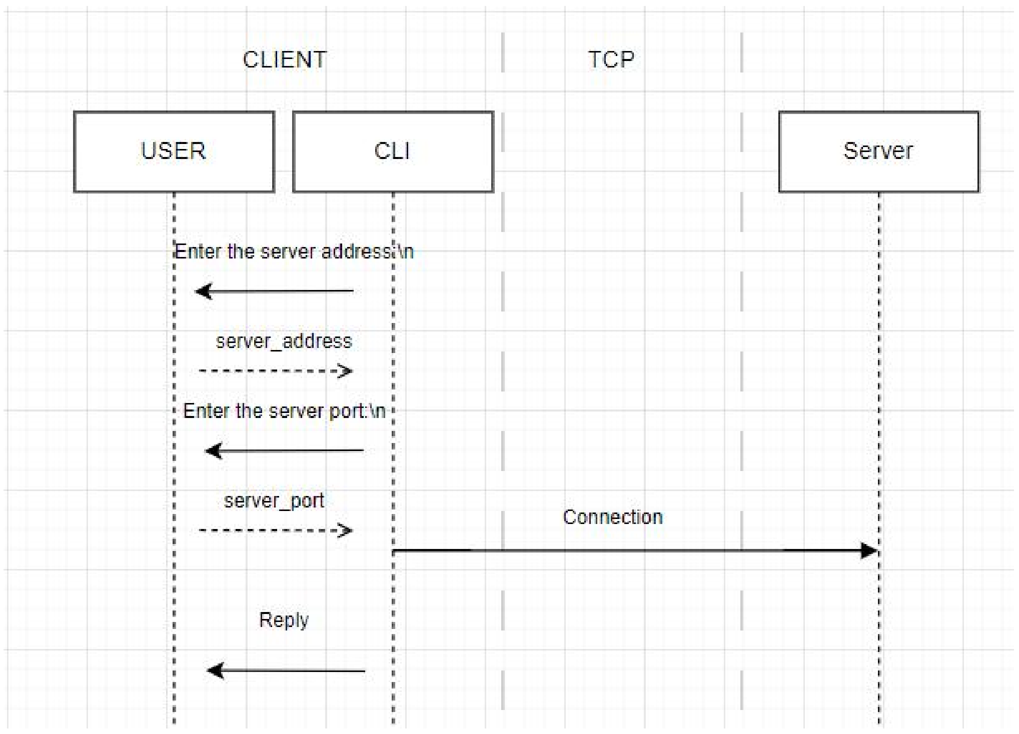
# : method invocation(static and not)

| : the value or the possible values of a variable are shown

SC : Socket Client

SS : Socket Server

Connection Phase



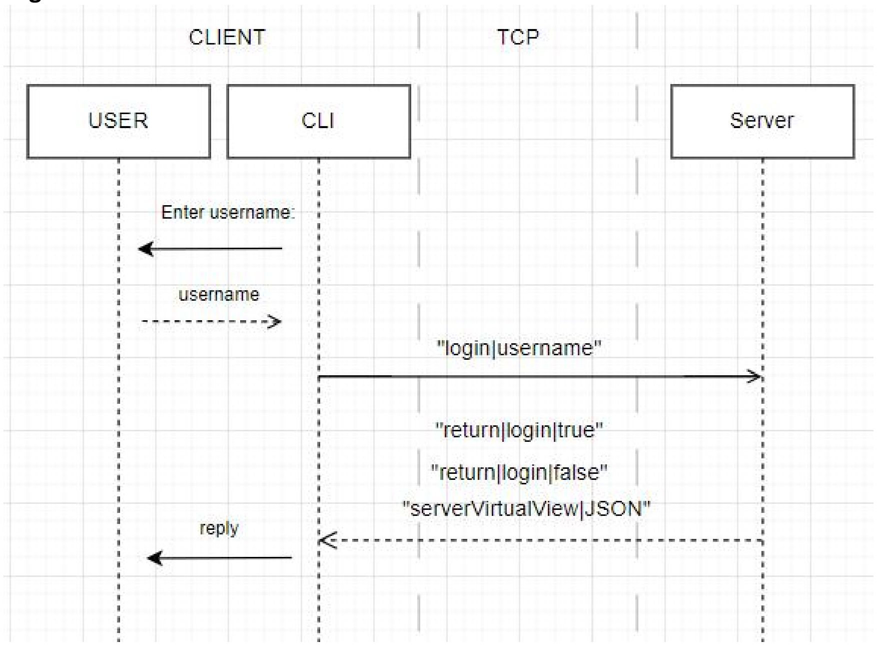
Socket initialization(server)
ServerSocket #run()

CLI - User Interactions:
CLI #askServerInfo()
CLI displays > Enter the server address: \n
User types > server_address\n
| server_address: localhost,...
CLI displays > Enter the server port: \n
User types > server_port\n
| server_port: 1234,8807,...

Socket initialization(client)
#SocketClient.run()
Socket acceptance (server)
SS #accept()

CLI displays > reply\n
| reply: Connection Successful, Connection Failed

Login Phase



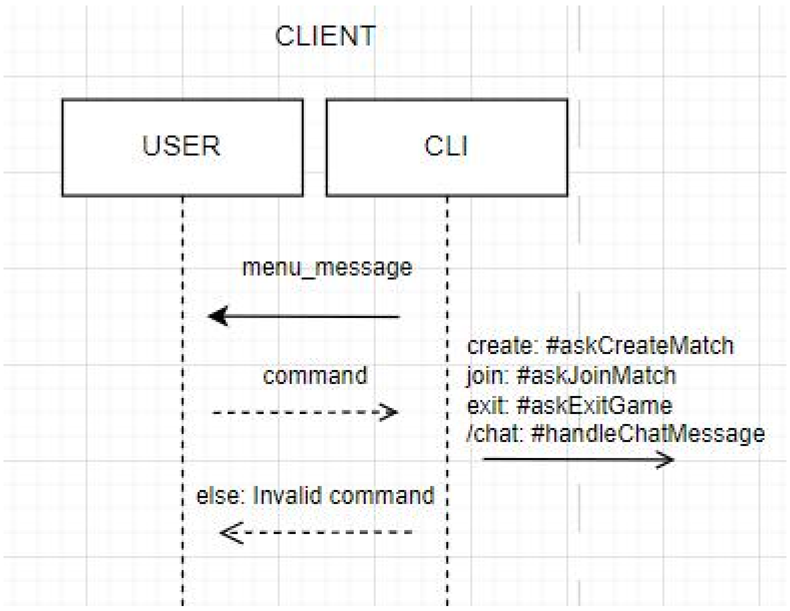
CLI - User Interactions:
CLI #askLogin()
CLI displays > Enter username:\n
User types > username\n
| username: [any name]

Login request (client)
SC #messageToServer(message)
|message:
    login|[username]

Login reply (server)
SS #callback(message)
| message:
    "return|login|[boolean]"
    "serverVirtualView|JSON"

CLI displays > reply\n
| reply:
| (if [bool] = true) Login successful. Hi [username]
| (if [bool] = false) Access denied. This username is already taken, please enter a new one...

Game Menu Phase

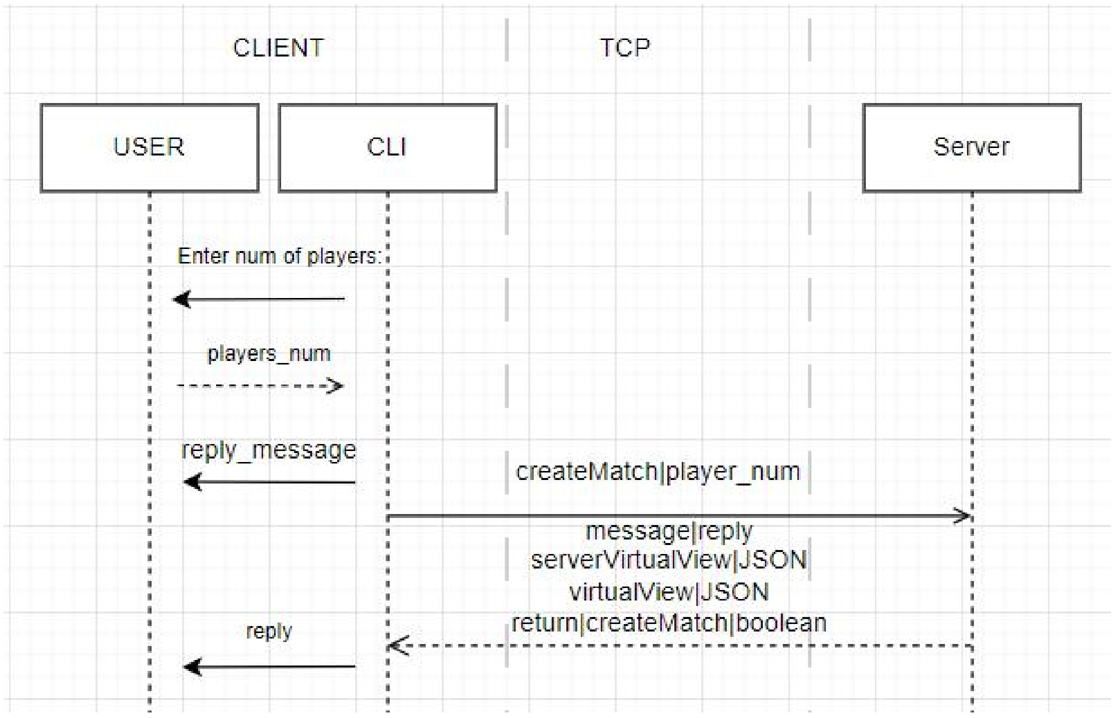


CLI - User Interactions:
CLI #askMenuAction()
CLI displays > menu_message\n
|menu_message:
| Menu option:
| create --> Create a new match
| join --> Join a match
| exit --> Exit game
| To send a message to a online player type '/chat[nickname]' followed by your message in the console.
| Enter the option number you wish to select (1,2 or3): \n

User types > command\n
| command: create,join,exit, /chat[[nickname]]:[text]

If [command] is 'create':
CLI #askCreateMatch() --->|Go to Create Match Phase|
If [command] is 'join':
CLI #askJoinMatch() --->|Go to Join Match Phase|
If [command] is 'exit':
CLI #askExitGame() --> |Go to Exit Game Phase|
If [command].startsWith '/chat':
CLI #handleChatMessage([command]) --> |Go to Chat Phase|
Else:
CLI displays > Invalid command. Try again.\n

Create Match Phase



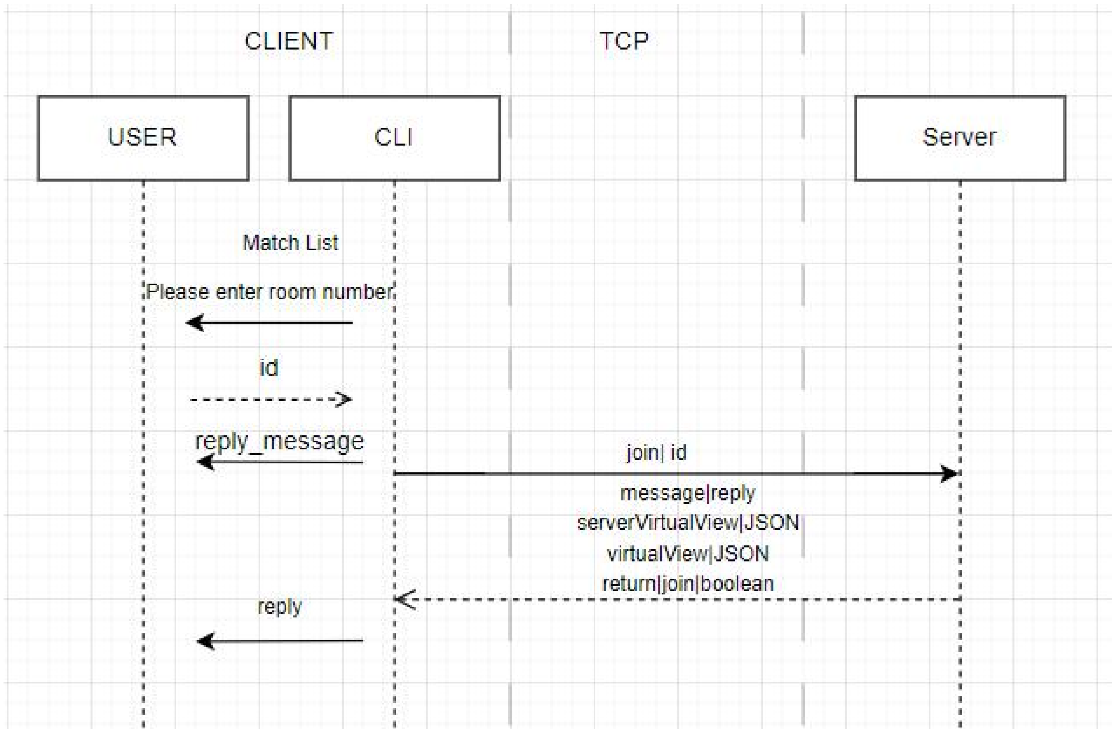
CLI - User Interactions:
CLI #askCreateMatch()
CLI #askMaxSeats()
CLI displays > Please select the max number of players for this match(2,3 or 4):\n
User types > players_number\n
|players_number: 2,3,4
CLI displays > reply_message\n
|reply_ message:
| (positive) Selected [players_number] seats.\n
| (negative) Invalid number!\n , Invalid input!\n

Create Match request (client)
SC #messageToServer(message)
| message:
“createMatch|[players_number]”

Create Match reply (server)
SS #callback(message)
|message:
“message|[reply]”
“serverVirtualView|JSON”
“virtualView|JSON”
“return|createMatch|[boolean]”

CLI displays > reply\n
|reply:
|(negative)The player already exists in a match. In order to create a new match, you need to abandon the current one.
| Do you wish to continue?\n
|(negative) Exceeded player number limit. Try again.\n
|(positive) Match with [player_number] seats created. MatchID: [matchID]\n

Join Match Phase



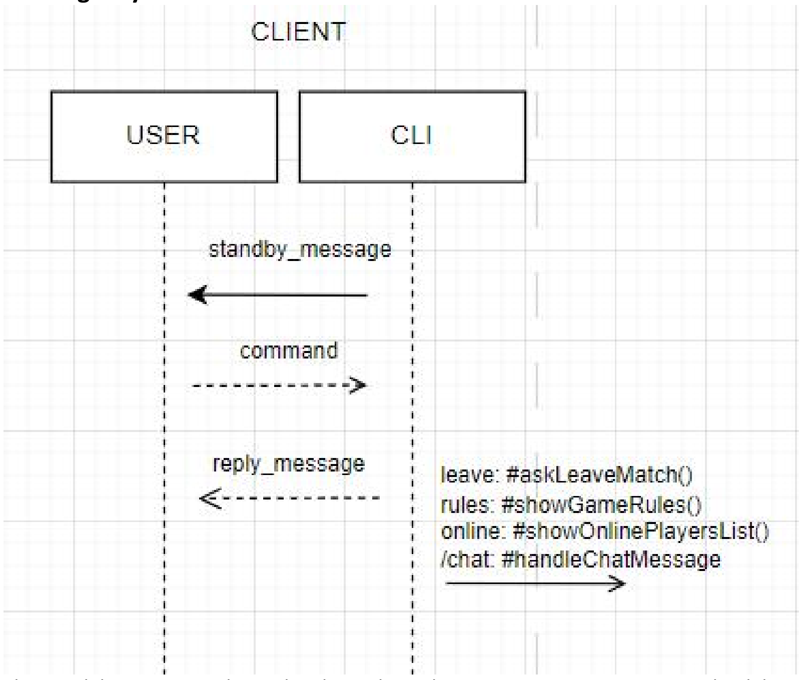
CLI - User Interactions:
CLI #askJoinMatch()
CLI displays > Match_List\n
|Match_List: a list of all matches in the game(from 0 to matchList.size()-1)
| Example of display: Room N.[matchID] - [GameState] ([nPlayers]/[maxSeats])
| GameState: WaitingPlayers, Ready, GameGoing, LastRound, Closed
| nPlayers: number of players in the room
| maxSeats: max number of players for the match
CLI displays > Please enter the room number:\n
User types > id\n
|id: from 0 to (matchList.size()-1)
CLI displays > reply_message\n
|reply_ message:
| (positive) Selected Room [id].\n
| (negative) Invalid number!\n , Invalid input!\n

Join Match request (client)
SC #messageToServer(message)
|message:
"join|[id]"

Join Match reply (server)
SS #callback(message)
|message:
"message|[reply]"
"serverVirtualView|JSON"
"virtualView|JSON"
"return|join|boolean"
|reply:
|(negative) The room selected does not exist. Try again.\n
|(negative|when the player try to join after connection loss) The player does not exist in any room. Try to create a new one.\n
|(negative) The room is full\n
|(positive) [username] joined the match [matchID]\n

CLI displays > reply\n

Waiting Players Phase



The match has not started yet. The players have the option to use some commands while waiting.

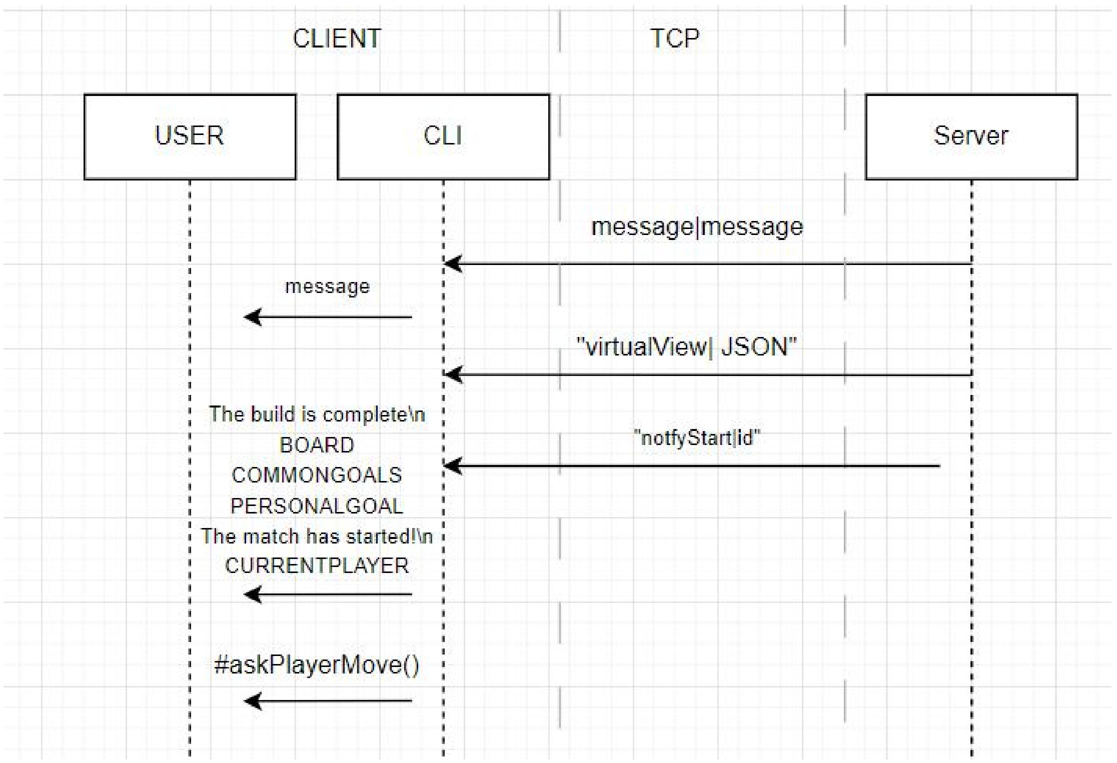
CLI - User Interactions:

CLI #askWaitingAction()
CLI displays > standby_message\n
| standby_message:
|	The match has not started yet. Waiting for more players to join... [maxSeats - nPlayers] seats available.
|	These are the commands available:
|	leave --> Leave Match
|	rules --> Read Game Rules
|	online --> Show Online Players
|	To send a message in the Match Chat type '/chat ' followed by your message in the console.
|	To send a message to a online player type '/chat[nickname]' followed by your message in the console.
|	Enter the number of the command you wish to execute (1 to 3):\n

User types > command\n
| coommand: leave,rules,online, /chat [message],/chat[[nickname]] [message]
CLI display > reply_message\n
| reply_message:
| (positive) [command] command selected \n
| (negative) Invalid command!\n , Invalid input!\n

If [command] is 'leave':
CLI #askLeaveMatch() --> |Go to Leave Match Phase|
If [command] is 'rules':
CLI #showGameRules() --> |Go to Show Game Object Phase|
If [command] is 'online':
CLI #showOnlinePlayersList()
If [command].substring(0,4) is '/chat':
CLI #handleChatMessage([command]) --> |Go to Chat Phase|

Match Initialization Phase



When a player joins and completes the room,then the match starts its initialization automatically.

Initialization Notification (server)

SS #callback(packet)
|packet: “message|[message]”

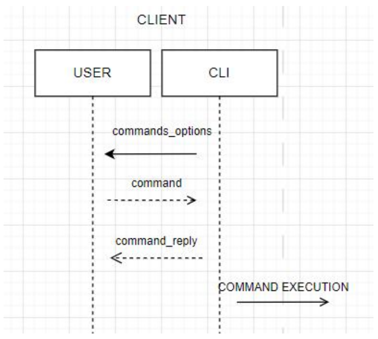
|message: The match is about to start. Building game board...
CLI displays > message\n

SS #callback(packet)
|packet: “virtualView|JSON”
--Note: Client is going to receive a virtualization of the game model

SS #callback(packet)
|packet: “notifyStart|id”

```
CLI #showMatchSetup()
 {
        CLI displays > The build is complete.
        CLI #showBoard()
        CLI displays > [Board]
        CLI #showCommonGoals()
        CLI displays > [Common Goals]\n
        CLI #showPersonalGoal()
        CLI displays > [Personal Goal]\n
        CLI displays > The match has started!\n
        CLI #showCurrentPlayer()
        If [current_player]==[username]:
        CLI displays > Hey [username]! It’s your turn\n
        Else:
        CLI displays > It’s [current_player]’s turn\n
        CLI #askPlayerMove() --> |Go to Gameplay Phase|
}
```

Gameplay Phase



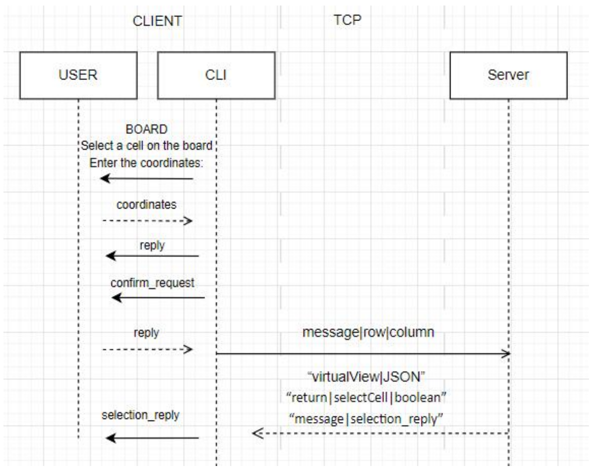
CLI - User Interactions:
CLI #askPlayerMove()
CLI displays > commands_options\n
| commands_options:
| What do you wish to do? These are the commands available:
| select --> Select a Cell
| deselect --> Deselect Items
| insert --> Insert your items in the shelf (at least one item selected)
| show --> Show Game Object(Hand,Goals,Board,Shelf,...)
| leave --> Leave Match
| exit --> Exit Game
| Enter the command you wish to use: \n

User types > command\n
|command: select, deselect, insert, pgoal, cgoal, shelf, board, stats, help,rules, end,timer,leave,exit,
| /chat [message],/chat[nickname] [message]
CLI displays > command_reply\n
|command_reply:
|(positive) Executing command...
|(negative) This command is not valid.\n, This command is not allowed right now. Wait your turn...\n,
The conditions to use this command are not respected. Try again.\n

COMMAND EXECUTION (CLI):
If [command] is 'select' :
CLI #askSelection() --> |Go to Selection Phase|
If [command] is 'deselect':
CLI #askDeselection() --> |Go to Deselection Phase|
If [command] is 'insert':
CLI #askInsertion() --> |Go to Insertion Phase|
If [command] is 'show':
CLI #askShowObjects() --> |Go to Show Game Object Phase|
If [command] is 'leave':
CLI #askLeaveMatch() --> |Go to Leave Match Phase|
If [command] is 'exit':
CLI #askExitGame() --> |Go to Exit Game Phase|
If [command].substring(0,5) is '/chat:' :
CLI #handleChatMessage() --> |Go to Chat Phase|

When the COMMAND EXECUTION is completed:
CLI #askPlayerMove() --> |Go to Gameplay Phase|

Gameplay Phase > Selection Phase



CLI - User Interactions:

```
CLI #askSelection()
{
CLI #showBoard()
CLI displays > [Board]\n Select a cell on the board.\n
CLI #askCoordinates()
{
        CLI displays > Enter the coordinates: \n
        User types > coordinates\n
        |coordinates: ([row],[column])
        CLI displays > reply\n
        |reply:
        |(negative) Invalid numbers., Invalid Input.
        |(positive) You have selected ([row],[column])
}
(end of: #askCoordinates) --> return: [coordinates]=([row],[column])
CLI displays> confirm_request\n
|confirm_request:
| Now you can confirm your choice(y),cancel your choice(n), retry again(r), see a Game Object(Board,Shelf,Goals,...)(show)?
User types > reply\n
|reply: y,n,r, show

If [reply] is 'n':
|Exit Selection Phase|
CLI #askPlayerMove() --> |Go to Gameplay Phase|
If [reply]is 'r':
CLI #askCoordinates()
If [reply] is 'show':
CLI #askShowObjects() --> |Go to Game Object Phase|

If [reply] is 'y':
```

Selection request (client)

SC #messageToServer(message)
|message: "selectCell|row|column"

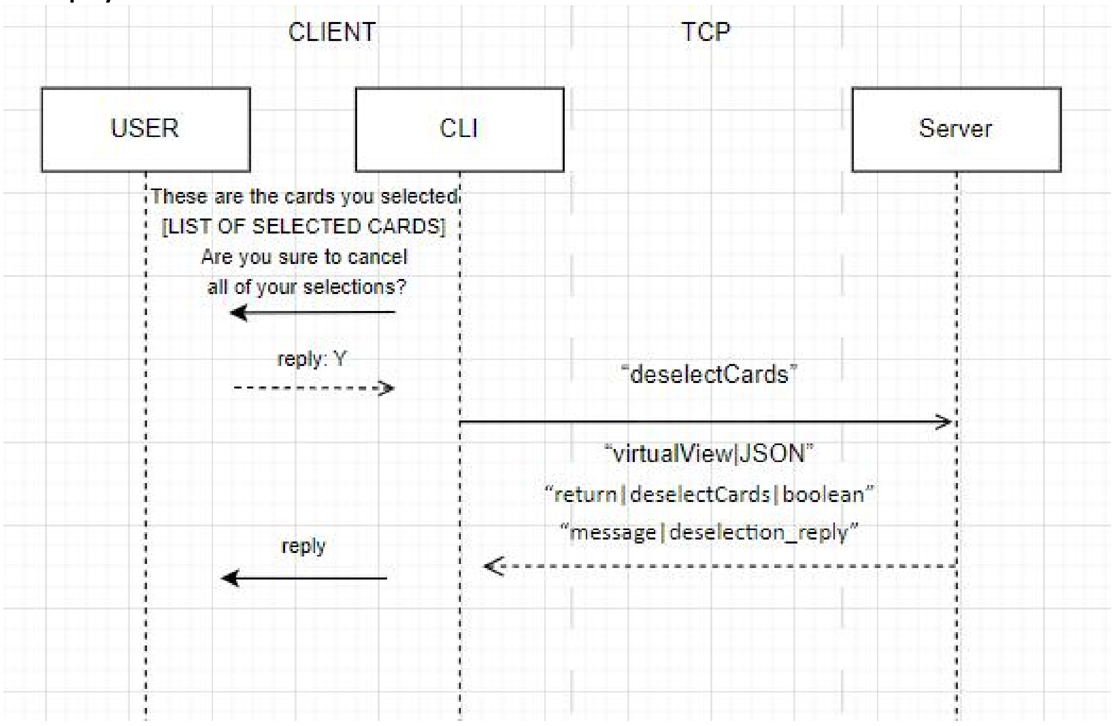
Selection reply (server)

SS #callback(message)

```
|message:
        "virtualView|JSON"
        "return|selectCell|boolean"
        "message|selection_reply"

|selection_reply:
|(positive) Selection successful!
|(negative) Selection failed: [reason]
        |reason:
        |(empty cell) You chose an empty cell
        |(already selected) You have already selected this cell
        |(hand limit) You reached the limit of items you could pick
        |(out of boundary) You selected an illegal cell
        |(no free side) The cell was not selectable. Pick an item which has a free side.
        |(no orthogonal)The cell was not selectable.Pick an item which is adjacent and in line with the other selected items.
CLI displays > selection_reply\n
CLI displays > Item selected: \n
CLI #showCell([coordinates])
```

Gameplay Phase > Deselection Phase



CLI - User Interactions:
CLI #askDeselection()
{
CLI displays > These are the cards you have selected:\n
CLI #showHand()
CLI diplays > [List of selected items]
CLI displays > Are you sure to cancel all of your selections? y/n
User types > reply\n
| reply: y , n

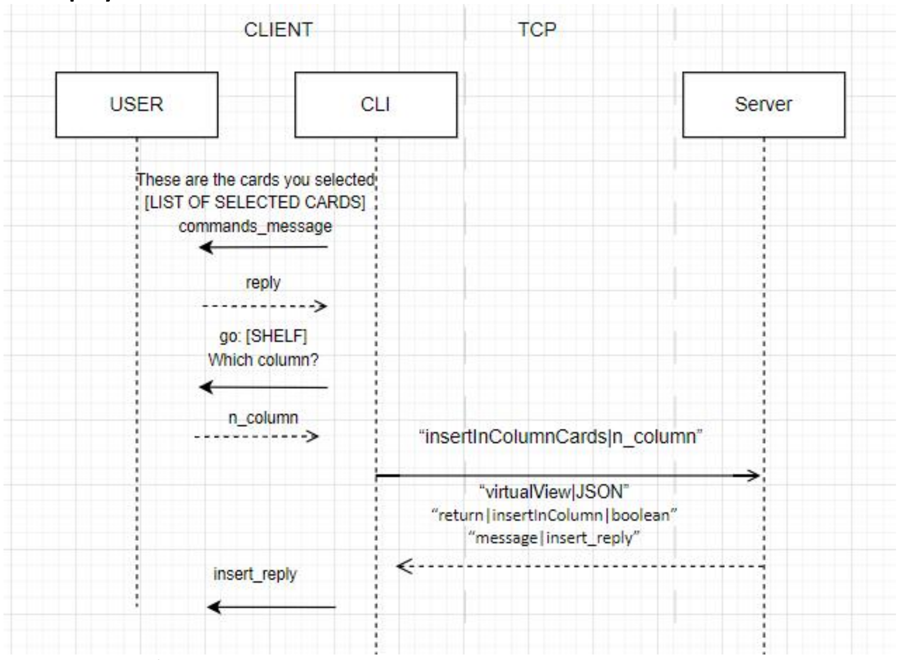
Deselection request (client)
SC #messageToServer(message)
|message: “deselectCards”

Deselection reply (server)
SS #callback(message)
| message:
“virtualView|JSON”
“return|deselectCards|boolean”
“message|deselection_reply”

|deselection_reply:
|(positive) Deselection successful.
|(negative) Deselection failed.
|(negative|hand.size()=0) You didn’t select any card yet. Deselection is not necessary.\n
CLI displays > message\n

CLI #askPlayerMove() --> |Go to Gameplay Phase|
}

Gameplay Phase > Insertion Phase



CLI -User Interactions:
CLI #askInsertion()
{
CLI displays > These are the cards you have selected:\n
CLI #showHand
CLI displays > [List of selected cards]
CLI displays > commands_message\n
|commands_message:
| These are the commands available:
| sort --> change the order of your cards
| show --> look at game board objects
| go --> go directly to insertion
| Type your command:
User types > reply\n
|reply: sort, show, go

If [reply] is 'sort':
CLI #askSort() --> |Go to Sort Phase|
If [reply] is 'show':
CLI #askShowObjects() --> |Go to Show Game Object Phase|
If [reply] is 'go':
Continue

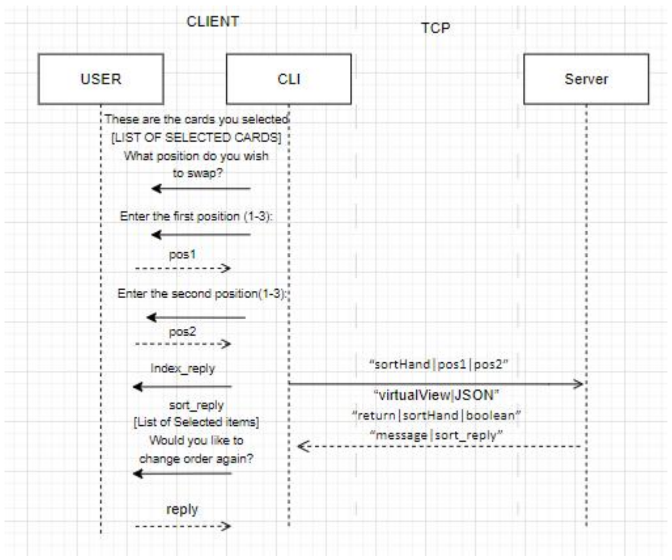
CLI #showShelf()
CLI displays > [Shelf]\n In which column would you like to insert the items? 1/2/3/4/5\n
User types > n_column\n
| n_column: 1,2,3,4,5

Insertion request (client)
SC #messageToServer(message)
|message: "insertInColumnCards|n_column"

Insertion reply (server)
SS #callback(insert_reply)
| insert_reply:
"virtualView|JSON"
"return|insertInColumn|boolean"
"message|insert_reply"

|(positve) Insertion Successful
|(negative|no space in column) There is not enough slot available in this column. Try again.
CLI displays > reply\n
CLI #askPlayerMove() --> |Go to Gameplay Phase|
}

Gameplay Phase > Sort Phase


CLIENT
TCP
USER
CLI
Server
These are the cards you selected
[LIST OF SELECTED CARDS]
What position do you wish to swap?
Enter the first position (1-3):
pos1
Enter the second position(1-3):
pos2
Index_reply
"sortHand|pos1|pos2"
"virtualView|JSON"
"return|sortHand|boolean"
"message|sort_reply"
sort_reply
[List of Selected items]
Would you like to change order again?
reply

CLI #askSort()
{
CLI displays > These are the cards you have selected:\n
CLI #showHand()
CLI displays > What position do you wish to swap?\n
CLI #askIndex()
{
CLI displays > Enter first position (1-3):\n
User types > pos1\n
CLI displays > Enter second position (1-3):\n
User types > pos2\n
|pos1&2: 1, 2, 3

CLI displays >index_reply
|index_reply:
|(positive) Position selected: [position1,position2]
|(negative|pos1=pos2) You selected the same position. |(negative|pos<1 or pos>3) Not valid input. Try again.

}
(end of: #askIndex())-->return: [pos1,pos2]

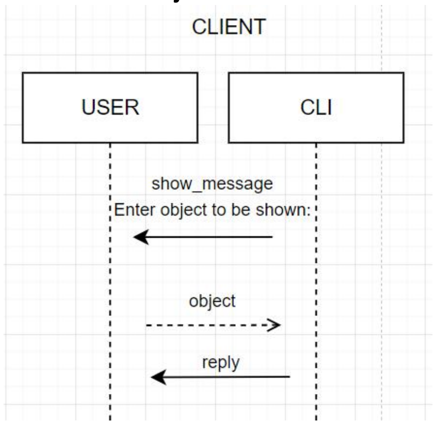
CLI displays > Would you like to Continue(press enter), to Retry (retry), to exit sort(cancel)
User types > reply
|reply : 'press enter',retry, cancel

Sorting request (client)
SC #messageToServer(message)
|message: "sortHand|pos1|pos2"

Sorting reply (server)
SS #callback(sort_message)
| sort_message:
"virtualView|JSON"
"return|sortHand|boolean"
"message|sort_reply"
|sort_reply:
|(positive) Order changed.
|(negative) Index out of border
|(negative) Not enough cards in your hand
CLI displays > sort_reply\n
CLI #showHand()
CLI displays > [List of selected items]
CLI displays > Would you like to change order again? Y/N\n
User types > reply\n
|reply: Y,N

If Y:
Loop #askSort()
If N:
Continue
}

Show Game Object Phase



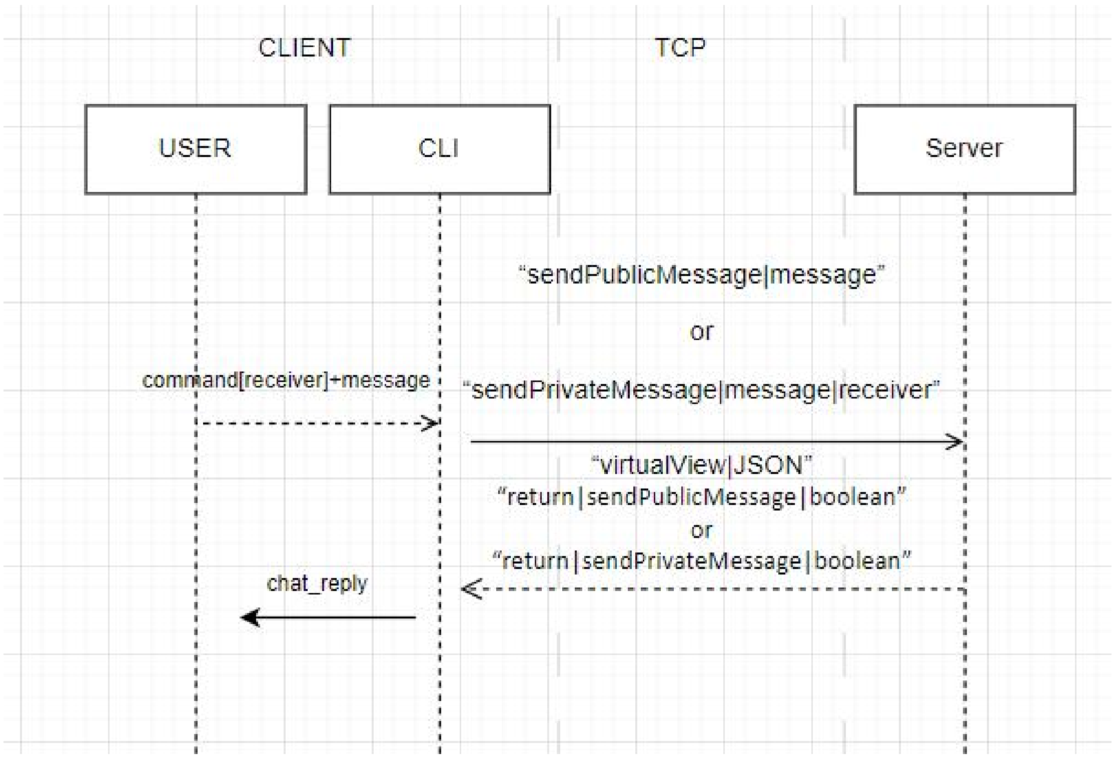
--**Note** : many of these objects are virtualized and already sent to CLIENT, so most of the time there no need to contact the SERVER for this task.

CLI - User Interactions:

```
If #askShowObjects() is called:
CLI #askShowObjects()
{
        CLI displays > show_message\n
        |show_message:
        |       List of Objects:
        |       open --> Show List of Player Chat
        |       hand --> Show selected items
        |       pgoal --> See your Personal Goal
        |       cgoal --> See Common Goals
        |       shelf --> See your shelf and the insertion limit
        |       board --> See Living Room Board
        |       stats --> See Players Stats
        |       rules --> See Game Rules
        |       end --> Show if the Endgame Token is taken (If it is, then it’s the last round)
        |       online --> Show Online Players
        |       Enter the object you wish to be shown:
        User types > object\n
        |object : hand, pgoal,cgoal,shelf,board,stats,rules,end,open

        If [object] is ‘hand’:
        CLI #showHand()
        If [object] is ‘pgoal’:
        CLI #showPersonalGoal()
        If [object] is ‘cgoal’:
        CLI #showCommonGoal()
        If [object] is ‘shelf’:
        CLI #showShelf()
        If [object] is ‘board’:
        CLI #showBoard()
        If [object] is ‘stats’:
        CLI #showPlayersStats()
        If [object] is ‘rules’:
        CLI #showGameRules()
        If [object] is ‘end’:
        CLI #showEndgameToken()
        If [object] is ‘online’:
        CLI #showOnlinePlayers()
        If [object] is ‘open:
        CLI #showPlayersChat()
}
-------------------------------------------------------------
If generic #show[Object]() method is called:
CLI displays > reply\n
|(positive) [OBJECT]
|(negative) The [object] cannot be found
---------------------------------------------------------------
If #showGameRules() is called:
CLI displays > [FULLGAMERULE_TEXT]
```

Chat Phase


CLIENT
TCP
USER
CLI
Server
"sendPublicMessage|message"
or
"sendPrivateMessage|message|receiver"
command[receiver]+message
"virtualView|JSON"
"return|sendPublicMessage|boolean"
or
"return|sendPrivateMessage|boolean"
chat_reply

CLI #handleChatMessage([command])
{
Chat request (client)
SC #messageToServer(message)
|message: "sendPublicMessage|[[command].substring(5,[command].length()]"
|Alternative:
|message: "sendPrivateMessage|[[command].substring(5,[command].length(|[receiver])"

Chat reply (server)
SS #callback(chat_message)
| chat_message:
    "virtualView|JSON"
    "return|sendPublicMessage|boolean" or "return|sendPrivateMessage|boolean"

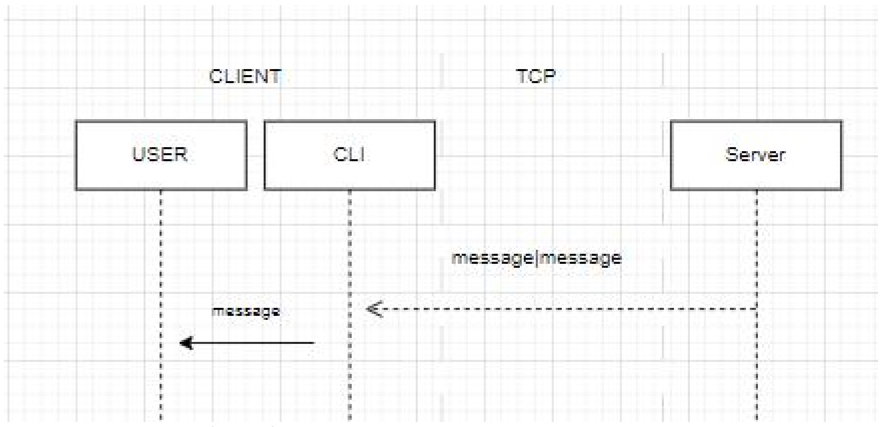
|chat_reply:
|(positive|to_player) Message sent to [nickname]
|(positive|to_group) [Match Chat History]
|(negative|to_player) Message didn't reach [nickname] ,
| The player [nickname] doesn't exist ,
| The player [nickname] is not online.
|(negative|to_group) Message was not sent.

CLI displays > chat_reply\n
}

Extra: Sending Message To Player (server)
SS #callback(notification)
|notification:
    "virtualView|JSON"
    "chatNotification|filtered_text"

|filtered_text:
| [sender_nickname] says:
| [command].substring(7+[nickname].length,[command].length())
CLI displays > filtered_text\n

Server Announcement



Server communication (server)
SS #callback(message)
|message:
|[nickname] has left the match
|[nickname] has left the game
|[nickname] has selected [CurrentPlayerHand]
|[nickname] has inserted items in the column [nColumn]
|[nickname] achieved [CommonGoal]
|[nickname] ended his turn.

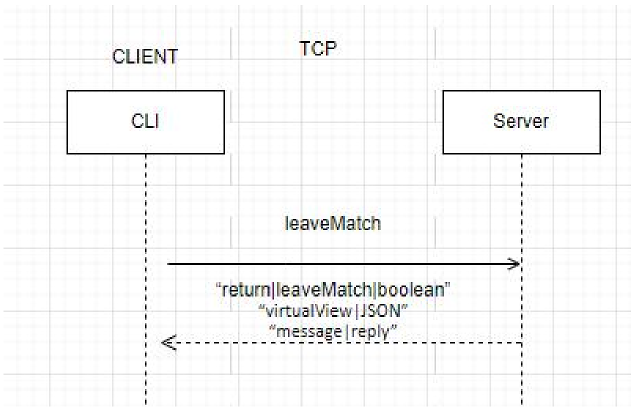
CLI displays > message\n

Gameplay Phase > Game Over Phase

Server communication (server)
SS #callback(message)
|message:
| GAME OVER
|--
|(If there is a winner) Congratulation! The winner is [winner]
CLI displays > message
CLI #showPlayerStats()
CLI displays > [Player Stats]

Then the players will all be removed from the match and match will be close and deleted.
CLI #askMenuAction() --> |Go to Menu Phase|

Leave Match Phase



CLI #askLeaveMatch()

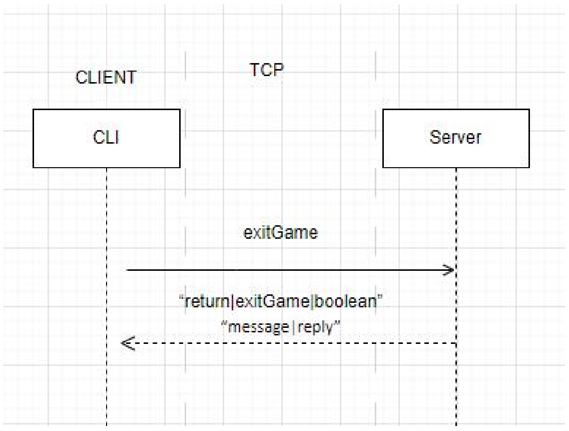
Leave match request (client)
SC #messageToServer(message)
|message: "leaveMatch"

Leave match reply (server)
SS #callback(message):
|message:
"return|leaveMatch|boolean"
"virtualView|JSON"
"message|reply"

|reply : You left the match
CLI displays > reply\n

CLI #askMenuAction() --> |Go To Menu Phase|

Exit Game Phase



CLI #askExitGame()

Exit Game request(client)
SC #messageToServer(message)
|message: "exitGame"

Players Data will be removed from the server(nickname, ...)

Exit Game request(server)
SS #callback(message):
|message:
"return|exitGame|boolean"
"message|reply"

|reply : Bye! See you soon.

CLI displays > reply\n
System.exit(0)