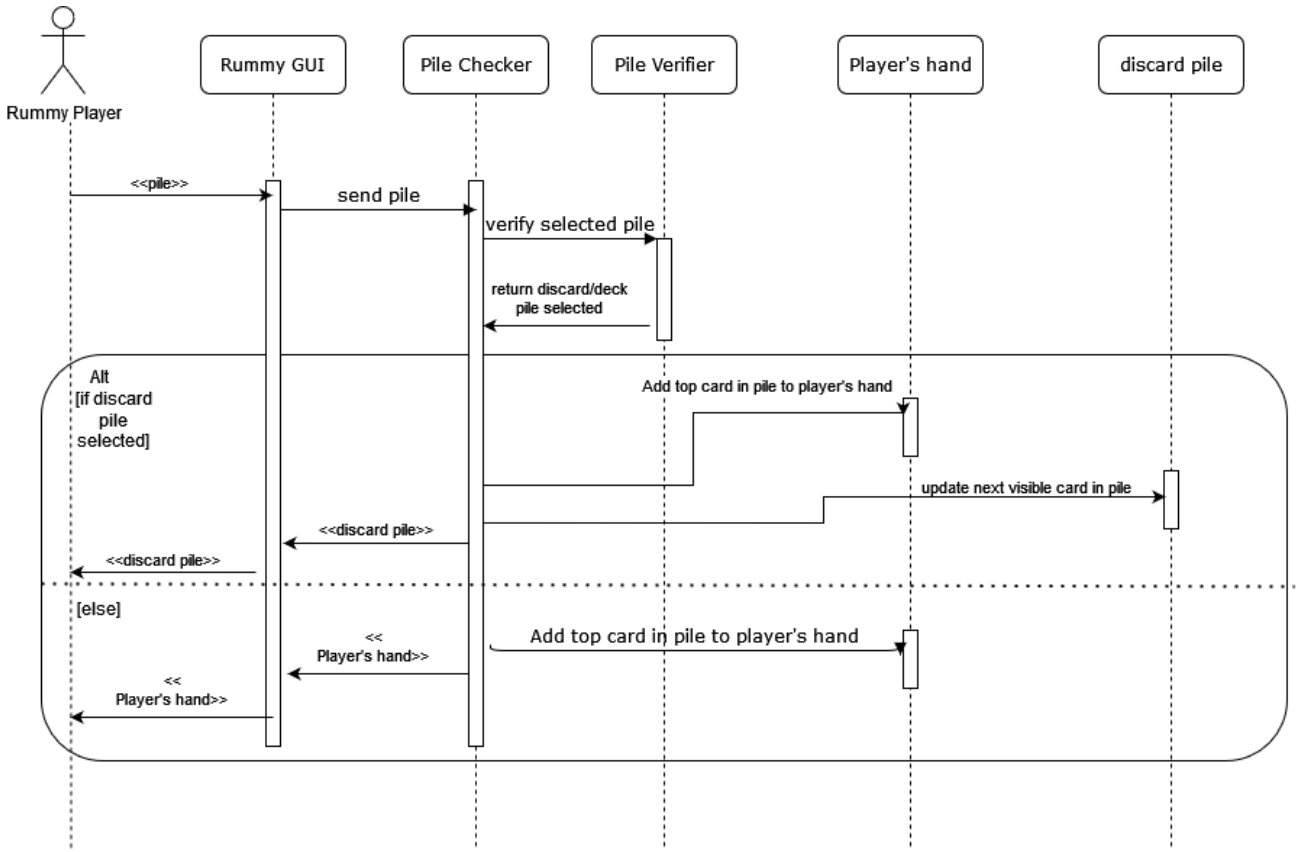
Non trivial steps (choose 7 of them):

- Use case 5: 2.) Super Rummy checks that the Save Code is valid, and that there aren't more people in the lobby than were in the saved game. If valid, it displays the "Choose Seats" screen.
- Use case 5: 4.) The system will display that user's username beneath the selected previous player's username.
- Use case 7: 2.) Super Rummy moves the top card of the tapped pile to the user's hand. If the discard pile was tapped, all users can see what is now the top of the discard pile.
- Use case 8: 2.) Super Rummy prevents the user from performing this move, providing a description of why the move was illegal, if possible.
- Use case 11: 2.) Super Rummy moves that card up slightly in the UI, so that it stands out from the others. It is registered as selected.
- Use case 12: 2.) Super Rummy moves that card to the discard pile, displaying it for all users, and ends User's turn.
- Use case 14: 4.) Super Rummy's next actions depend on users choice, however, base case would be to provide the user with a "resume play" input display

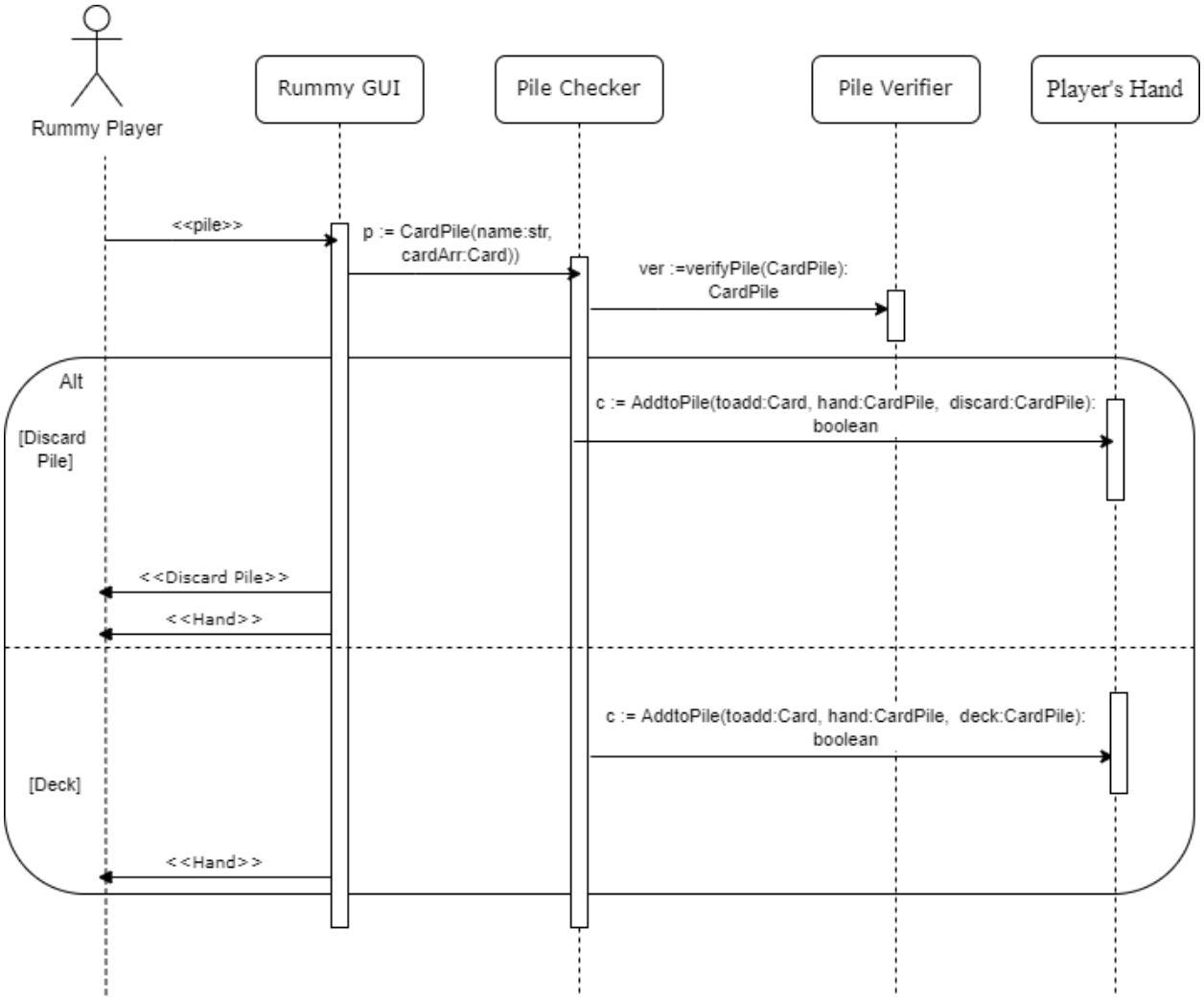
Sequence Diagrams:

7:2)

#	Subject	Action of Subject	Other Data/Objects	Object Acted Upon
1	Rummy Player	taps	pile	Rummy GUI
2	Rummy GUI	sends	pile	Pile Checker
2.1	Pile Checker	verifies	Selected pile	Pile verifier
2.2	Pile verifier	returns	discard/deck pile selected	Pile Checker
2.3	If discard pile was selected then			
2.3.1	Pile Checker	adds	Top card in pile	Player's hand
2.3.2	Pile Checker	returns	Player's hand	Rummy GUI
2.3.3	Pile Checker	updates	Next visible card	Discard pile
2.3.4	Pile Checker	returns	Discard pile	Rummy GUI
2.3.5	Rummy GUI	displays	Discard pile	Rummy Player
2.3.6	Rummy GUI	displays	Player's hand	Rummy Player
2.4	else			
2.4.1	Pile Checker	adds	Top card in pile	Player's hand
2.4.2	Pile Checker	returns	Player's hand	Rummy GUI
2.4.3	Rummy GUI	displays	Player's hand	Rummy Player


Rummy Player
Rummy GUI
Pile Checker
Pile Verifier
Player's hand
discard pile
<<pile>>
send pile
verify selected pile
return discard/deck pile selected
Alt
[if discard pile selected]
Add top card in pile to player's hand
update next visible card in pile
<<discard pile>>
<<discard pile>>
[else]
Add top card in pile to player's hand
<< Player's hand>>
<< Player's hand>>

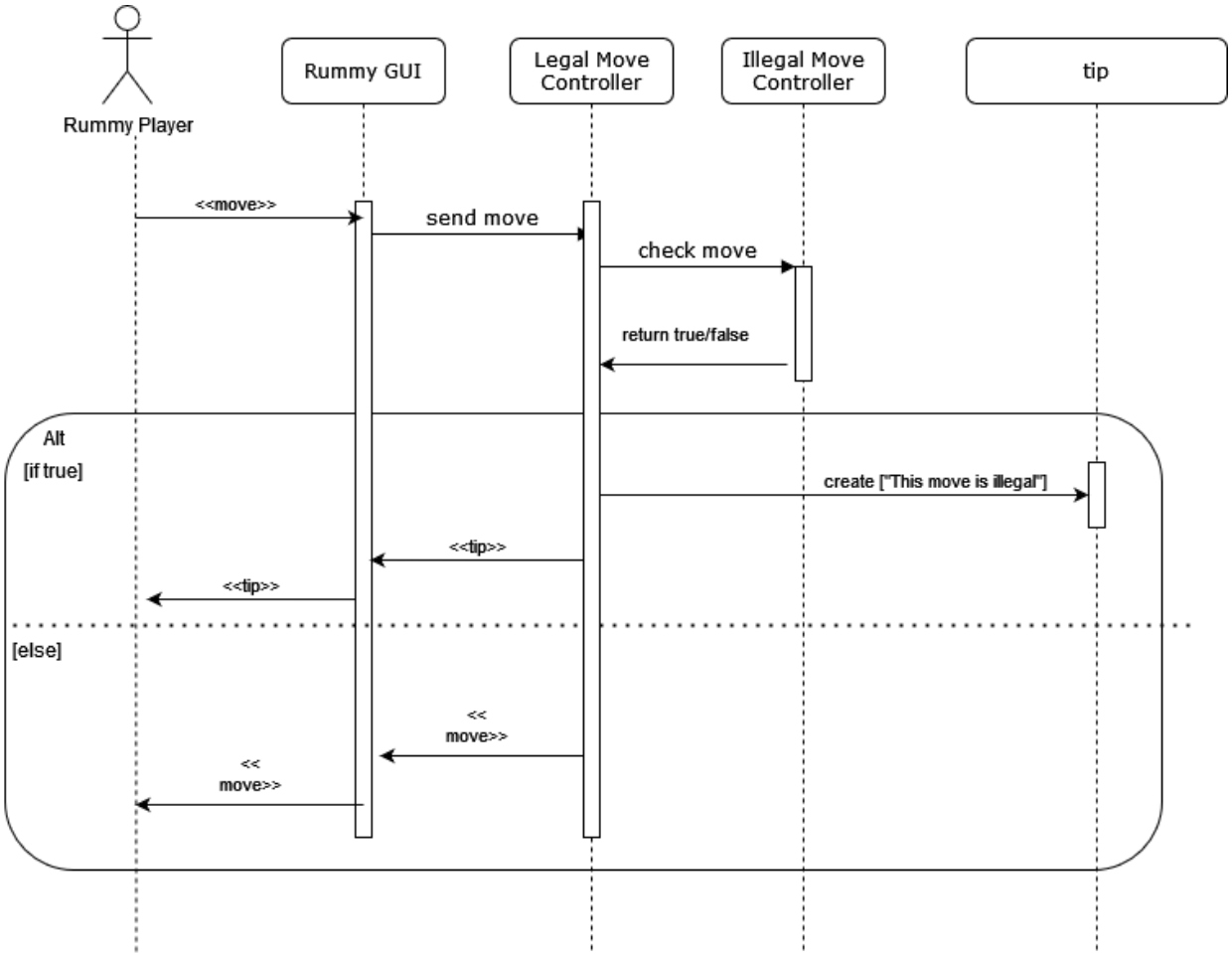
7:2 Design)



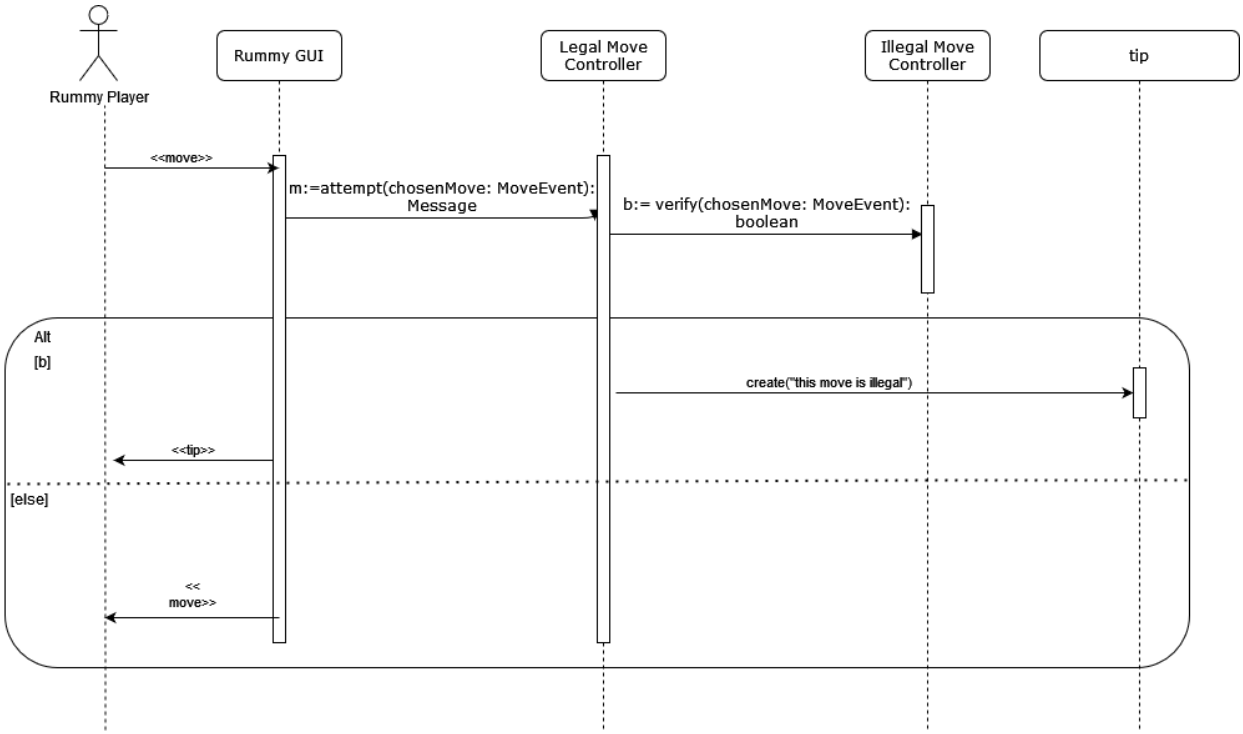
8:2)

#	**Subject**	**Action of Subject**	**Other Data/Objects**	**Object Acted Upon**
1	User	attempts	move	Rummy GUI
2	Rummy GUI	sends	move	Legal Move Controller
2.1	Legal Move Controller	checks	move	Illegal Move Controller

2.2	Illegal Move Controller	returns	true/false	Legal Move Controller
2.2.1	If true, then			
2.2.2	Legal Move Controller	creates	“This move is illegal”	tip
	Legal Move Controller	returns	tip	Rummy GUI
	Rummy GUI	displays	tip	Rummy Player
2.3	else			
2.3.1	Legal Move Controller	sends	move	Rummy GUI
2.3.2	Rummy GUI	displays	move	Rummy Player



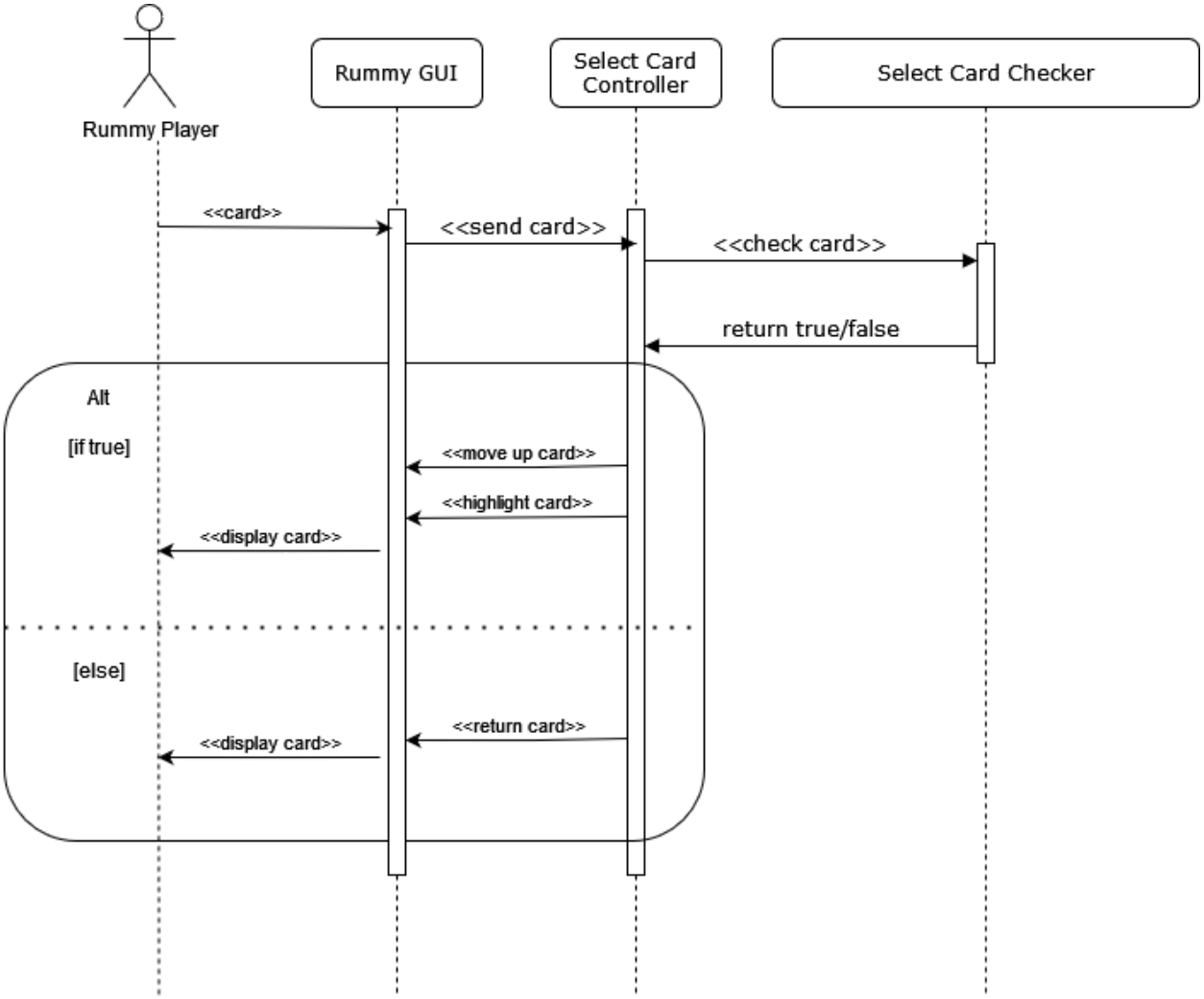
8:2) Design:

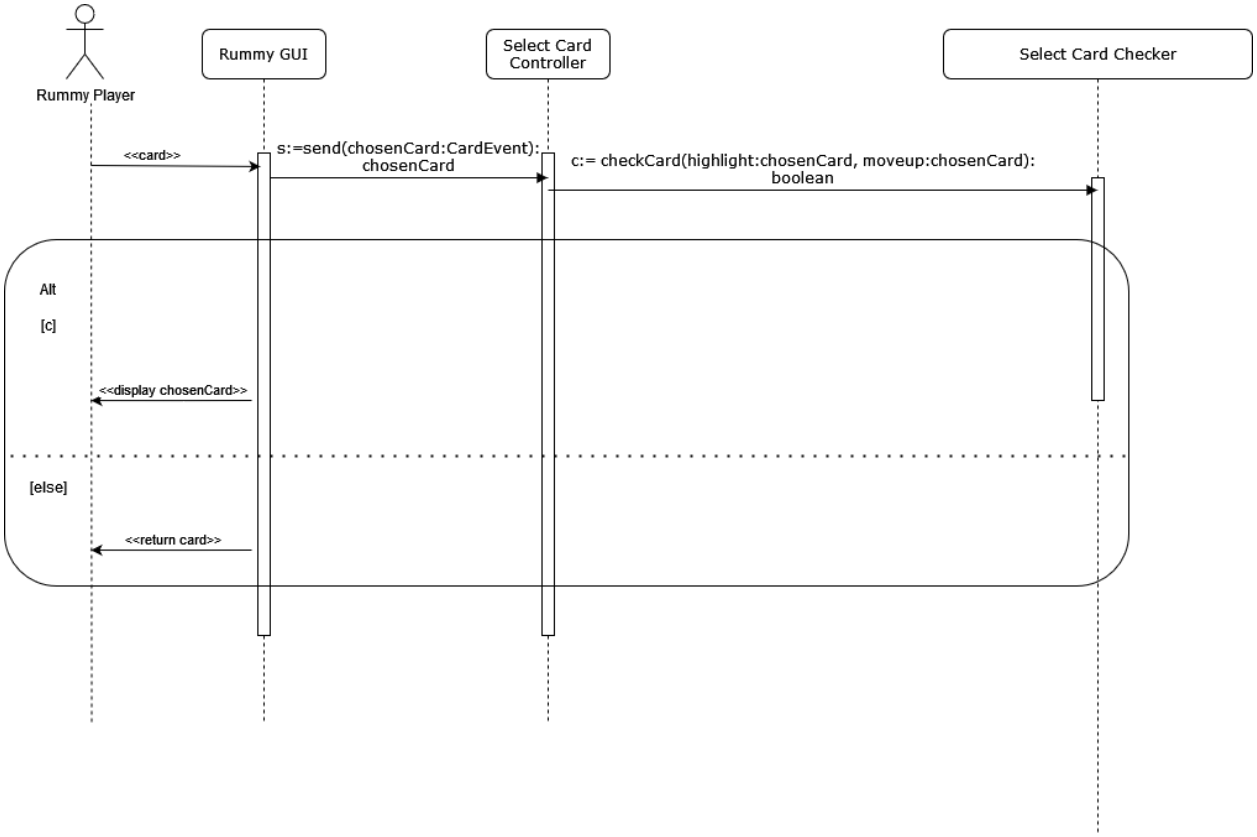


11:2)

#	**Subject**	**Action of Subject**	**Other Data/Objects**	**Object Acted Upon**
1	Rummy Player	taps	card	Rummy GUI
2	Rummy GUI	sends	card	Select Card Controller
2.1	Select Card Controller	checks	card	Select Card Checker

2.2	Select Card Checker	returns	true/false	Select Card Controller
2.3	If true…			
2.3.1	Select Card Controller	highlights	card	Rummy GUI
2.3.2	Select Card Controller	moves up	card	Rummy GUI
2.3.3	Rummy GUI	displays	card	Rummy Player
2.4	else…			
2.4.1	Select Card Controller	returns	card	Rummy GUI
2.4.2	Rummy GUI	displays	card	Rummy Player

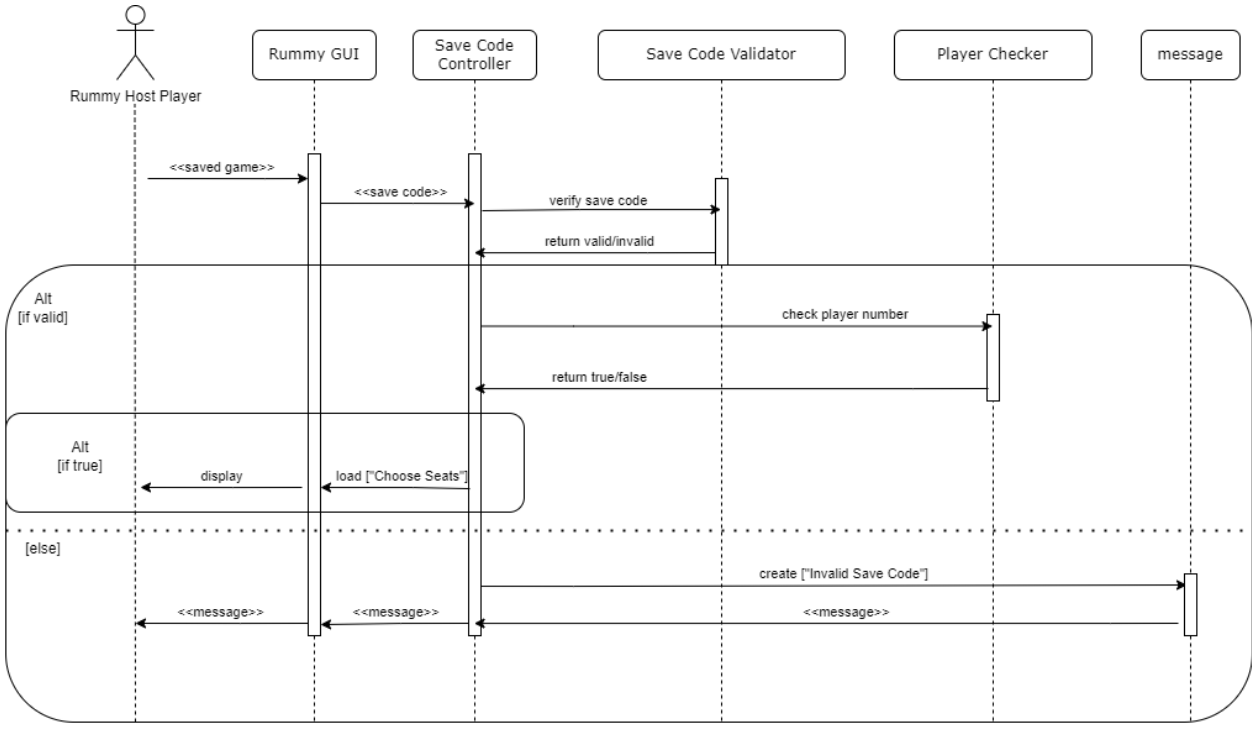

Rummy Player
Rummy GUI
Select Card Controller
Select Card Checker
<<card>>
<<send card>>
<<check card>>
return true/false
Alt
[if true]
<<move up card>>
<<highlight card>>
<<display card>>
[else]
<<return card>>
<<display card>>



5:2)

#	**Subject**	**Action of Subject**	**Other Data/Objects**	**Object Acted Upon**
1	Rummy Host Player	selects	saved game	Rummy GUI
2	Rummy GUI	sends	save code	Save Code Controller
2.1	Save Code Controller	verifies	save code	Save Code Validator
2.2	Save Code Validator	returns	valid/invalid	Save Code Controller
2.3	If valid then…			
2.3.1	Save Code Controller	checks	number of players in lobby == saved number players in lobby	Player Checker

2.3.2	Player Checker	returns	true/false	Save Code Controller
2.3.3	If true then…			
2.3.3.1	Save Code Controller	loads	“Choose Seats” screen	Rummy GUI
2.3.3.2	Rummy GUI	displays	“Choose Seats” screen	Rummy Host Player
2.4	else…			
2.4.1	Save Code Controller	creates	“Invalid Save Code”	message
2.4.2	Save Code Controller	returns	message	Rummy GUI
2.4.3	Rummy GUI	displays	message	Rummy Host Player

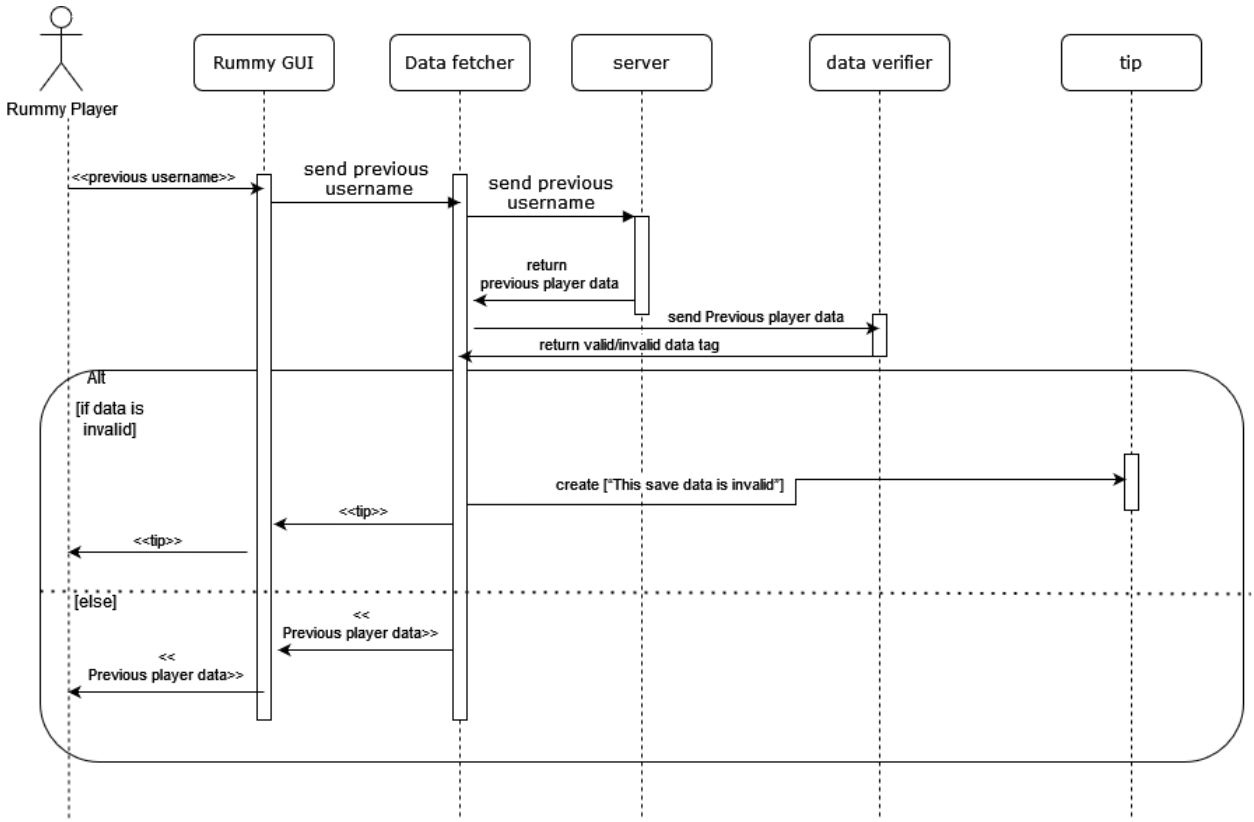


Description: Professor Ghanavati wanted me to leave a short description on this to help during grading. We added a nested “if” statement in the diagram and table to check if the number of

players in the lobby matches the number of players when the game was saved. Professor said it looked fine and no “else” statement was needed in the nested “if”.

5: 4.)

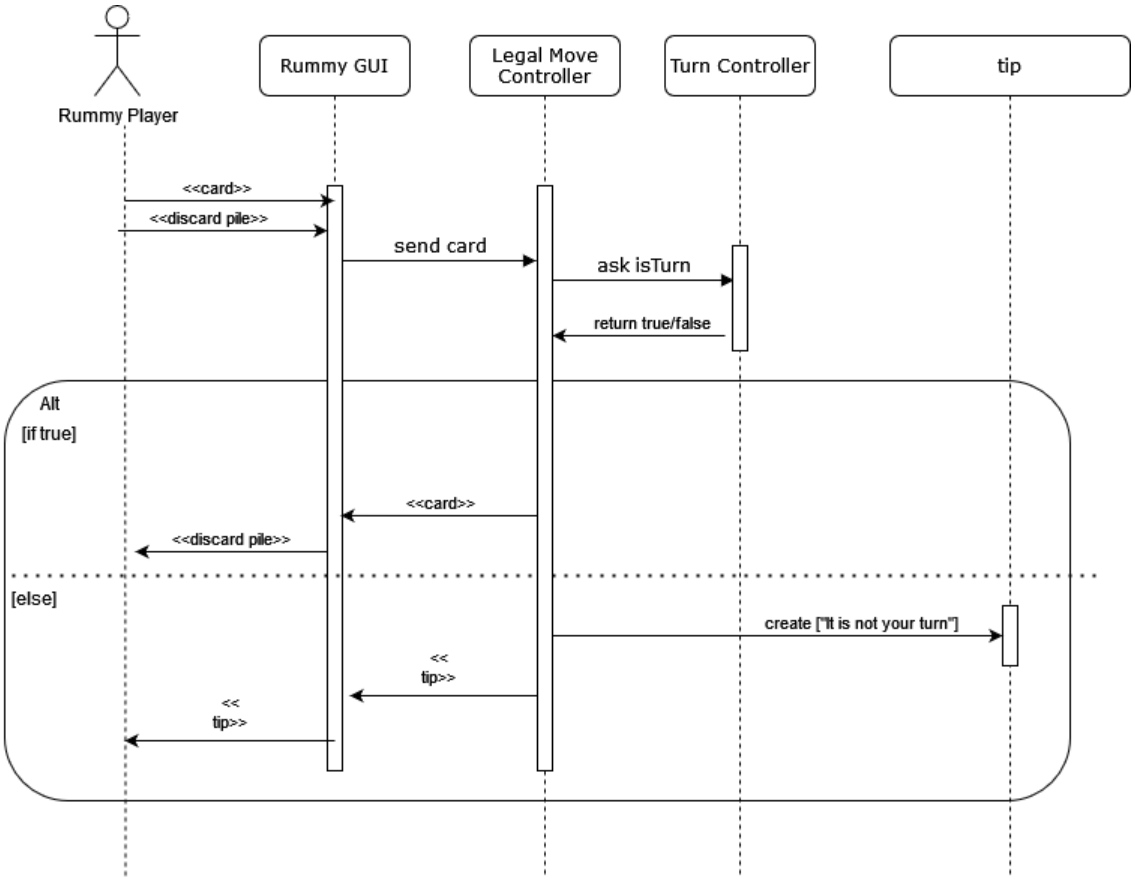
#	**Subject**	**Action of Subject**	**Other Data/Objects**	**Object Acted Upon**
1	Rummy Player	selects	Previous username	Rummy GUI
2	Rummy GUI	sends	Previous username	Data fetcher
2.1	Data fetcher	sends	Previous username	server
2.2	server	returns	Previous player data	Data fetcher
2.3	Data fetcher	sends	Previous player data	Data verifier
2.4	Data verifier	returns	valid/invalid data tag	Data fetcher
2.4.1	If data is invalid			
2.4.2	Data fetcher	creates	“This save data is invalid”	tip
2.4.3	Data fetcher	returns	tip	Rummy GUI
2.4.4	Rummy GUI	displays	tip	Rummy Player
2.5	else			
2.5.1	Data fetcher	returns	Previous player data	Rummy GUI
2.5.2	Rummy GUI	displays	Previous player data	Rummy Player



12: 2.)

#	Subject	Action of Subject	Other Data/Objects	Object Acted Upon
1	Rummy Player	selects	card	Rummy GUI
2	Rummy Player	Selects	Discard pile	Rummy GUI
3	Rummy GUI	sends	card	Legal Move Controller
4	Legal Move Controller	asks	isTurn	Turn Controller
4.1	Turn Controller	Returns	True/False	Legal Move Controller
4.2	If true, then			
4.2.1	Legal Move Controller	returns	card	Rummy GUI
4.2.2	Rummy GUI	updates	Discard pile	Rummy Player

4.3	else			
4.3.1	Legal Move Controller	creates	“It is not your turn”	tip
4.3.2	Legal Move Controller	returns	tip	Rummy GUI
4.3.3	Rummy GUI	displays	tip	Rummy Player

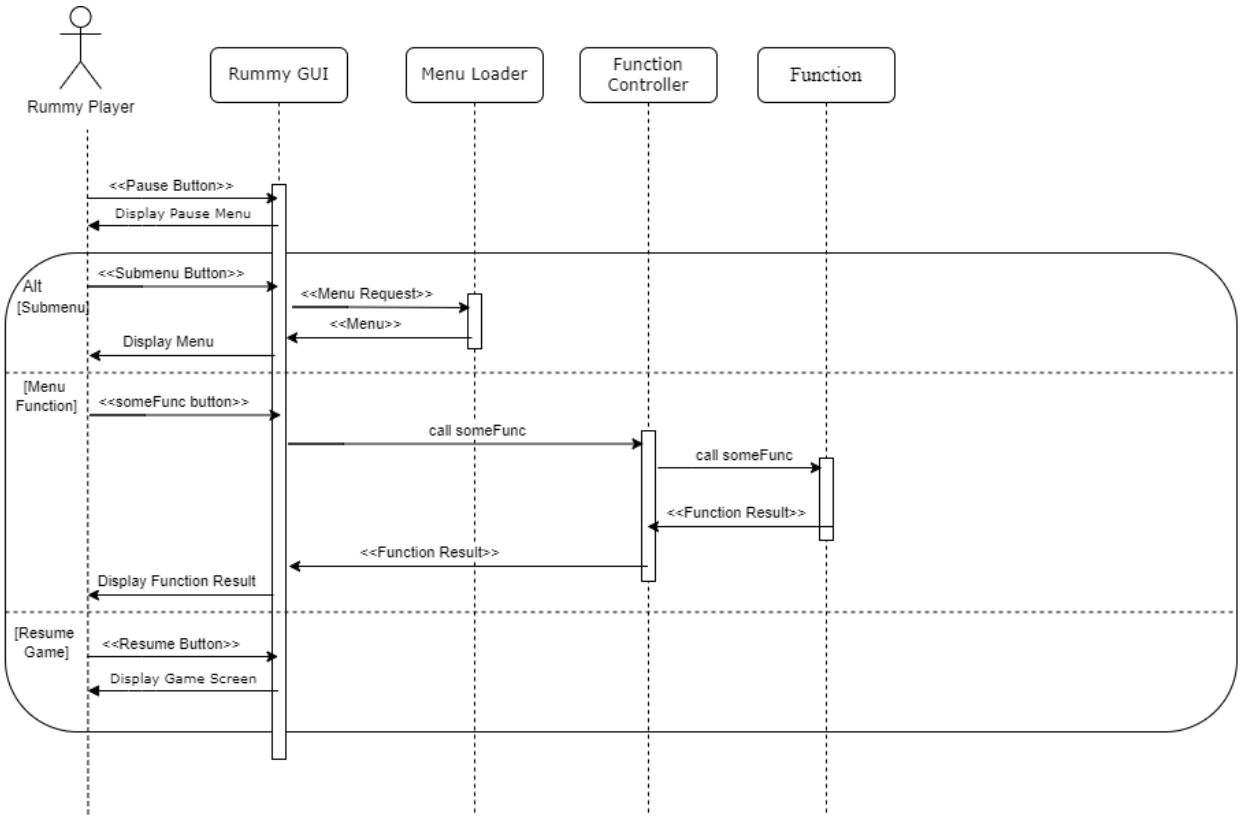


14: 4.) Super Rummy’s next actions depend on users choice, however, base case would be to provide the user with a “resume play” input display

The “User Choices” are just different menus/functionalities to go to from the pause menu.

#	**Subject**	**Action of Subject**	**Other Data/Objects**	**Object Acted Upon**

1	Rummy Player	selects	Pause Button	Rummy GUI
2	Rummy GUI	displays	Pause Menu	Rummy GUI
3	Rummy Player	selects	Menu option	Rummy GUI
3.1	If player selects submenu	I.e. “Settings”		
3.1.1	Rummy Player	selects	Submenu button	Rummy GUI
3.1.2	Rummy GUI	requests	menu	Menu Loader
3.1.3	Menu Loader	returns	Menu	Rummy GUI
3.1.3	Rummy GUI	displays	Menu	Rummy Player
3.2	Elif player selects menu function	I.e. “Save”		
3.2.1	Rummy Player	selects	someFunc button	Rummy GUI
3.2.1	Rummy GUI	calls	someFunc	Function Controller
3.2.2	Function Controller	performs	someFunc	Rummy GUI
3.2.3	Rummy GUI	displays	Function Result	Rummy Player
3.3	Player exits pause menu			
3.3.1	Rummy Player	selects	“Resume Game” button	Rummy GUI
3.3.2	Rummy GUI	displays	Game Screen	Rummy Player


Rummy Player
Rummy GUI
Menu Loader
Function Controller
Function
<<Pause Button>>
Display Pause Menu
Alt
[Submenu]
<<Submenu Button>>
<<Menu Request>>
<<Menu>>
Display Menu
[Menu Function]
<<someFunc button>>
call someFunc
call someFunc
<<Function Result>>
<<Function Result>>
Display Function Result
[Resume Game]
<<Resume Button>>
Display Game Screen