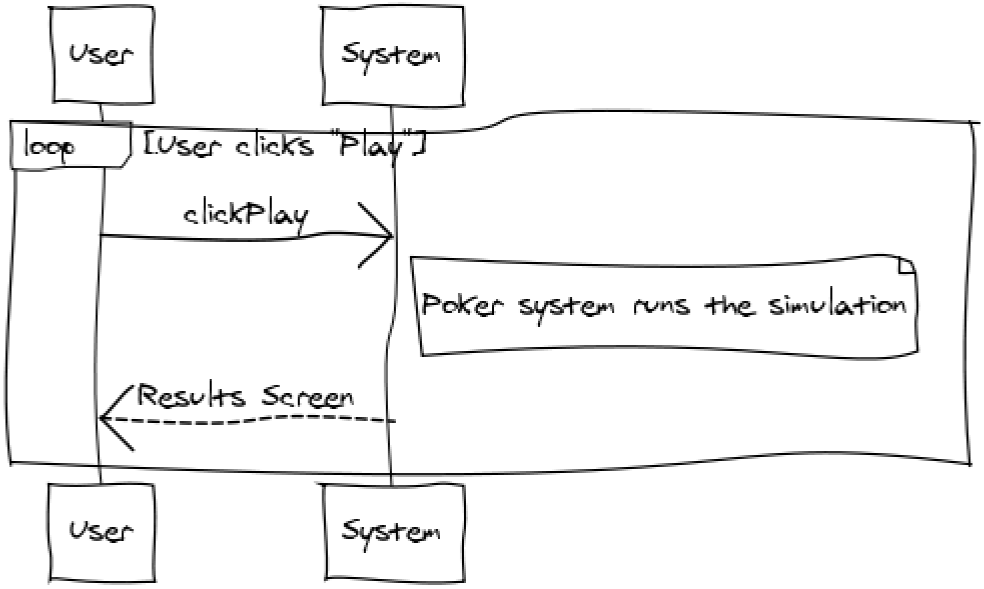
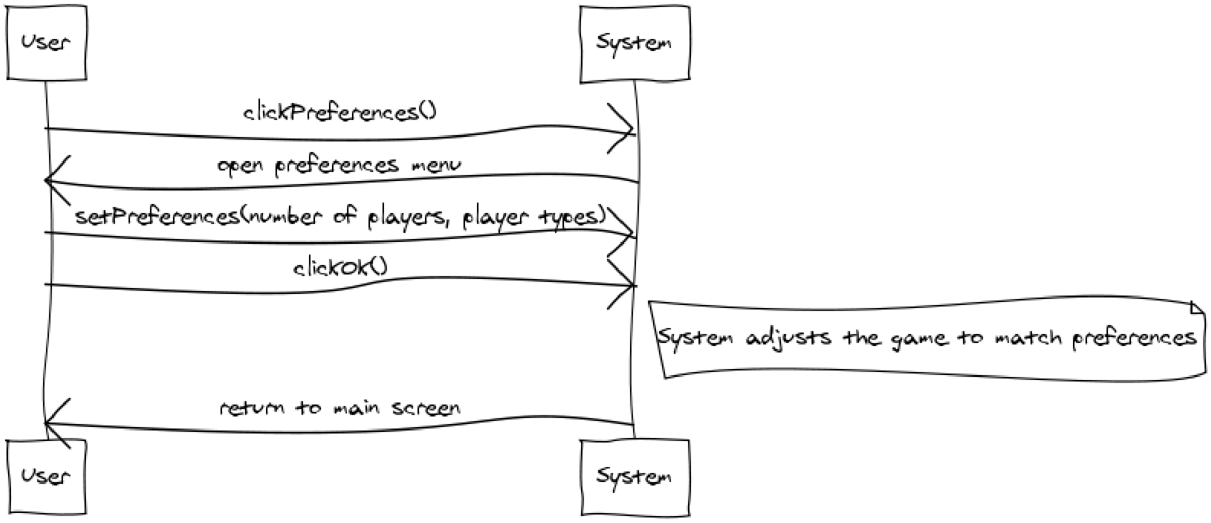
SSD for use case 1

```
loop User clicks "Play"
    User->System: clickPlay
    note right of System: Poker system runs the simulation
    System-->User: Results Screen
End
```



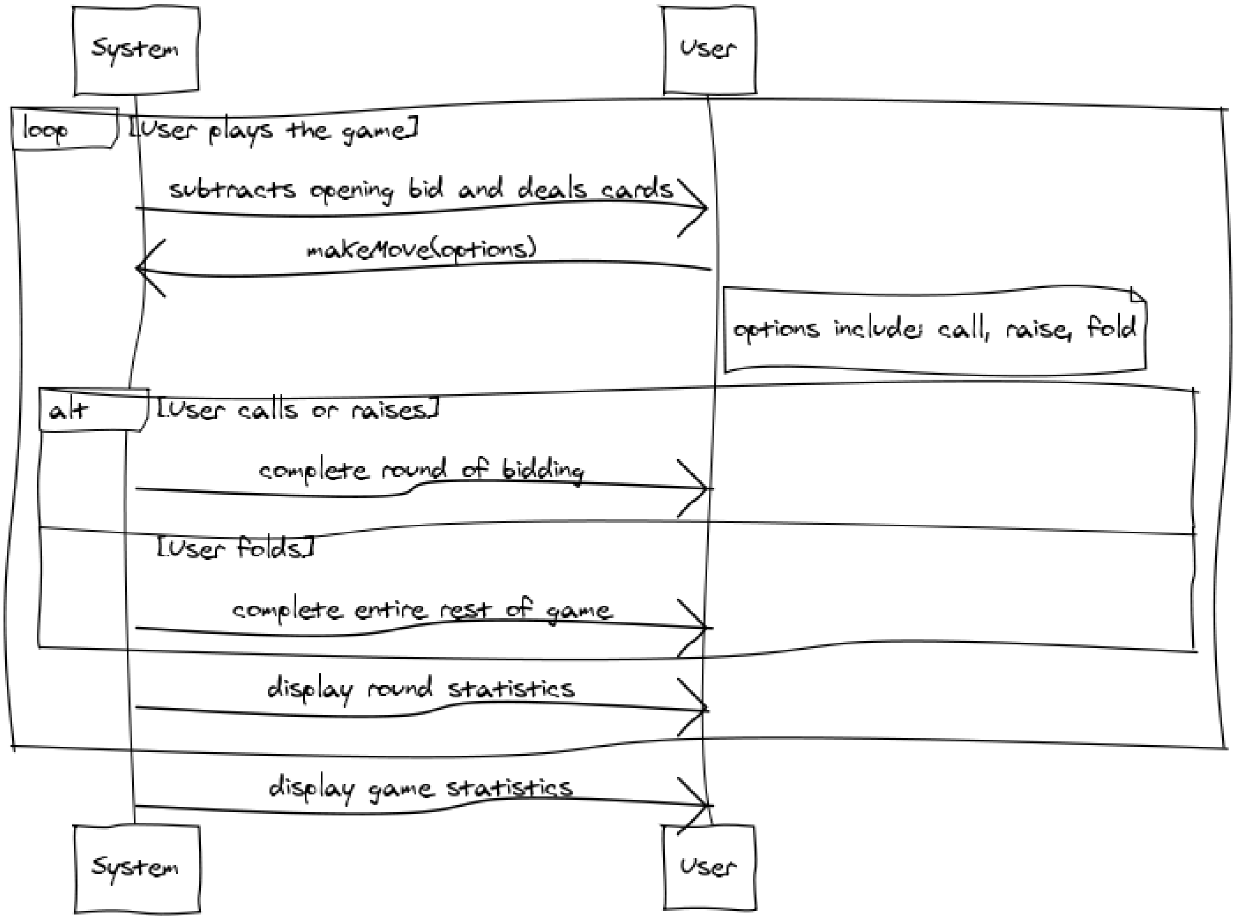
SSD for use case 2

User->System: clickPreferences()
System->User: open preferences menu
User->System: setPreferences(number_of_players, player_types)
note right of System: System adjusts the game to match preferences
System->User: return to main screen


User
System
clickPreferences()
open preferences menu
setPreferences(number of players, player types)
clickOK()
System adjusts the game to match preferences
return to main screen
User
System

SSD for use case 3

```
loop User plays the game
System->User: subtracts opening bid and deals cards
User->System: makeMove(options)
note right of User: options include: call, raise, fold
alt User calls or raises
   System->User: complete round of bidding
else User folds
   System->User: complete entire rest of game
end
System->User: display round statistics
end
System->User: display game statistics
```



Contract CO1: clickPlay

Operation:	clickPlay()
Cross References:	Use Cases: Starting the Game
Preconditions:	User has set player options
Postconditions:	The user watches the simulation take place

Contract CO2: setPreferences

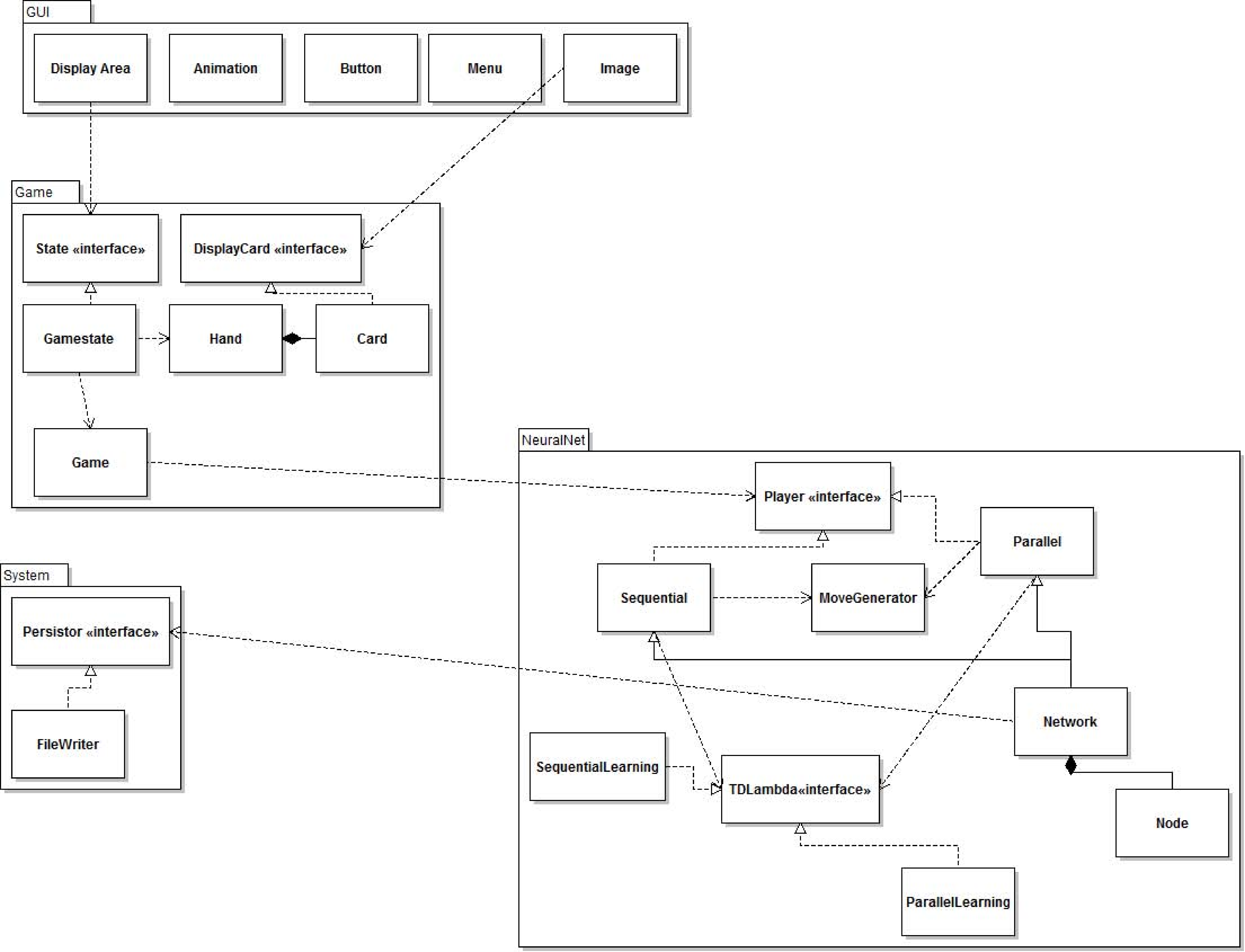
Operation:	setPreferences()
Cross References:	Use Cases: Setting Player Options
Preconditions:	The application has initialized properly Parameter menus are functioning properly
Postconditions:	The types of players have changed based on what the user has selected.

Contract CO3: playGame

Operation:	playGame()
Cross References:	Use Cases: N/A
Preconditions:	User has set player options to include human player User has clicked "Start" button
Postconditions:	User views game statistics and/or plays another game

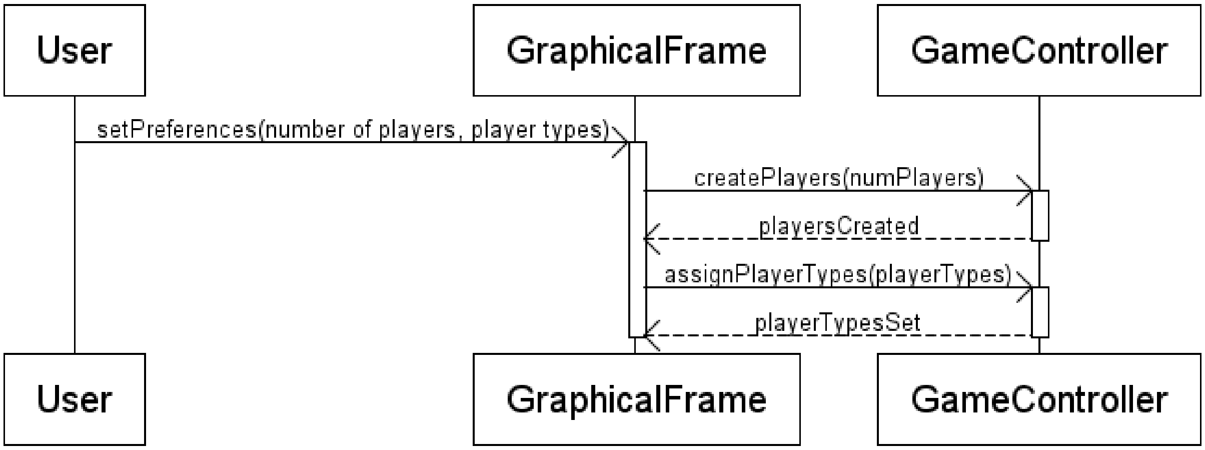
Contract CO4: makeMove

Operation:	makeMove()
Cross References:	Use Cases: N/A
Preconditions:	The game is in play It is the user's turn in the game
Postconditions:	The appropriate amount of chips move to the pot The user's stack is deducted the appropriate amount The game continues on to the next turn


GUI
Display Area
Animation
Button
Menu
Image
Game
State «interface»
DisplayCard «interface»
Gamestate
Hand
Card
Game
NeuralNet
Player «interface»
Parallel
Sequential
MoveGenerator
Network
SequentialLearning
TDLambda«interface»
Node
ParallelLearning
System
Persistor «interface»
FileWriter

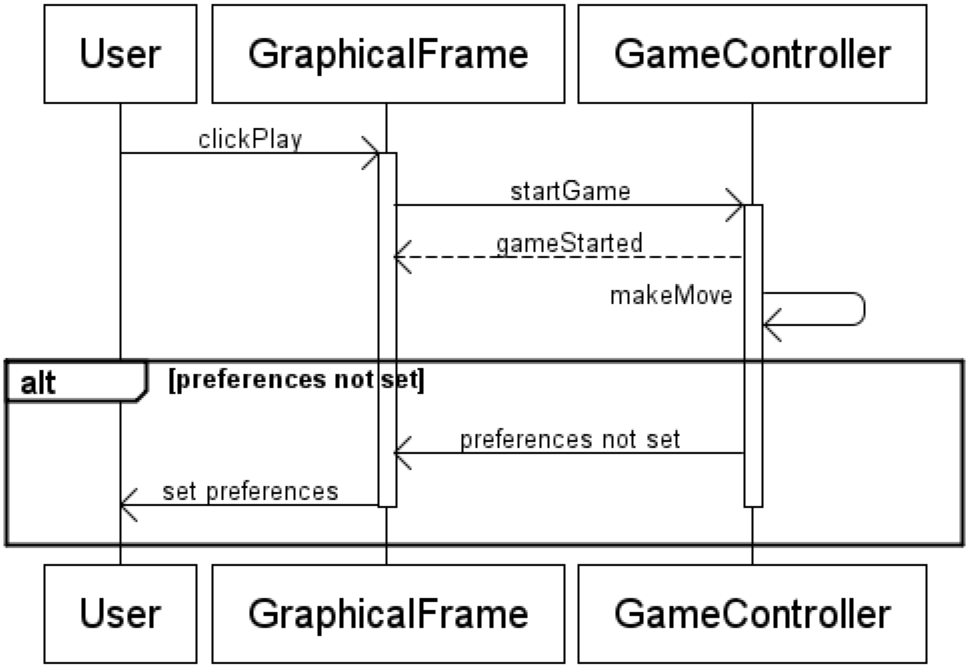
Set Preferences Operation

```
participant User
participant GraphicalFrame
participant GameController
User -> GraphicalFrame: setPreferences(number of players, player types)
activate GraphicalFrame
GraphicalFrame -> GameController: createPlayers(numPlayers)
activate GameController
GameController --> GraphicalFrame: playersCreated
deactivate GameController
GraphicalFrame -> GameController: assignPlayerTypes(playerTypes)
activate GameController
GameController --> GraphicalFrame: playerTypesSet
deactivate GameController
deactivate GraphicalFrame
```


User
GraphicalFrame
GameController
setPreferences(number of players, player types)
createPlayers(numPlayers)
playersCreated
assignPlayerTypes(playerTypes)
playerTypesSet
User
GraphicalFrame
GameController

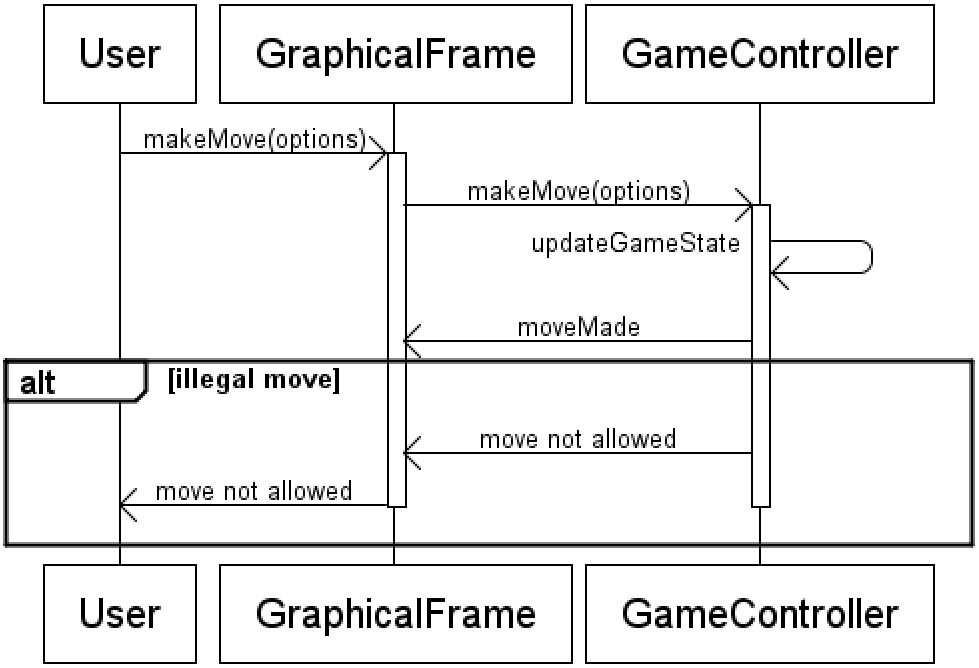
Begin Game Operation

```
participant User
participant GraphicalFrame
participant GameController
User -> GraphicalFrame: clickPlay
activate GraphicalFrame
GraphicalFrame -> GameController: startGame
activate GameController
GameController --> GraphicalFrame: gameStarted
GameController -> GameController: makeMove
alt preferences not set
GameController -> GraphicalFrame: preferences not set
GraphicalFrame -> User: set preferences
deactivate GameController
deactivate GraphicalFrame
```



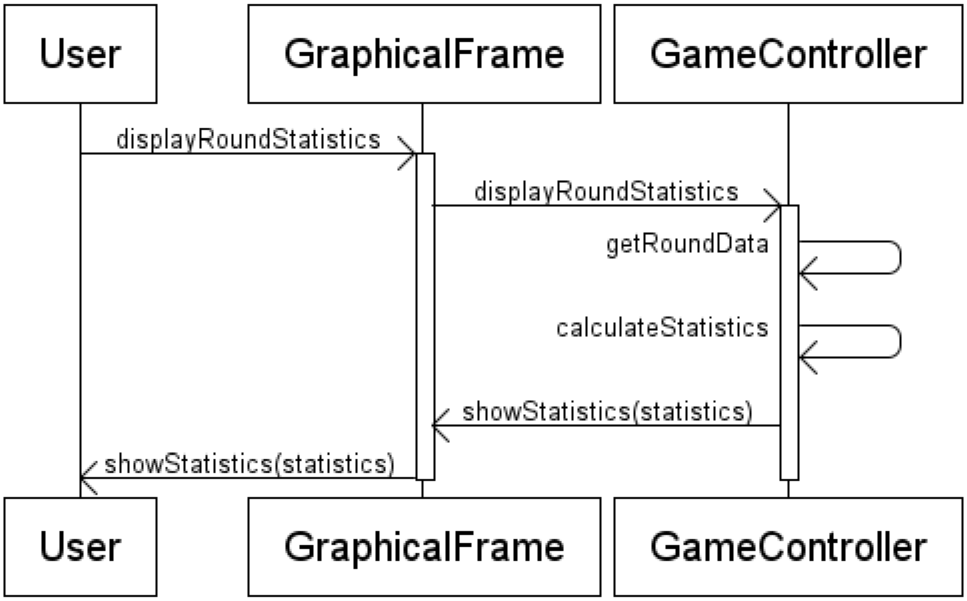
Make Move Operation

```
participant User
participant GraphicalFrame
participant GameController
User -> GraphicalFrame: makeMove(options)
activate GraphicalFrame
GraphicalFrame -> GameController: makeMove(options)
activate GameController
GameController -> GameController: updateGameState
GameController -> GraphicalFrame: moveMade
alt illegal move
GameController -> GraphicalFrame: move not allowed
GraphicalFrame -> User: move not allowed
deactivate GameController
deactivate GraphicalFrame
```



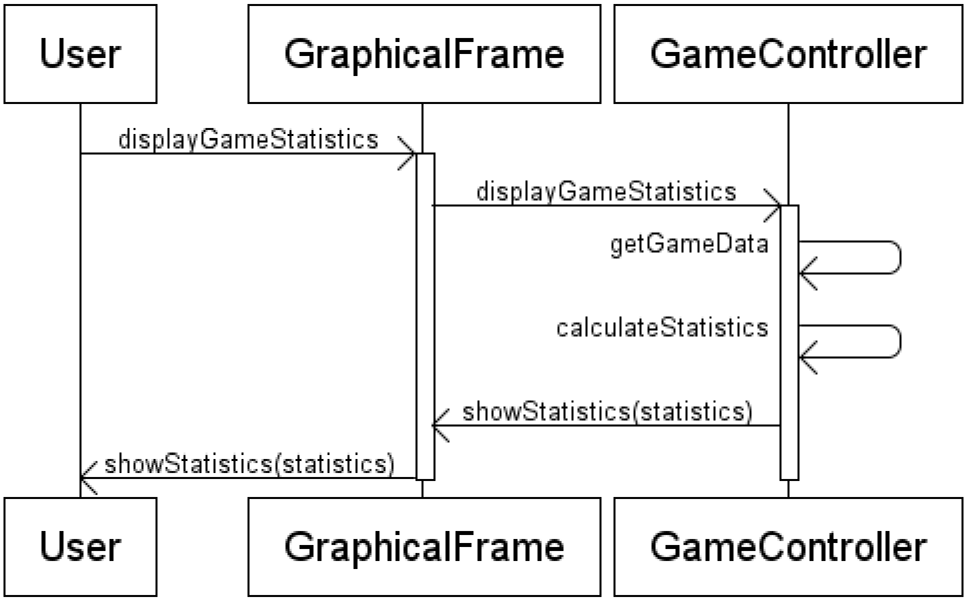
Display Round Statistics Operation

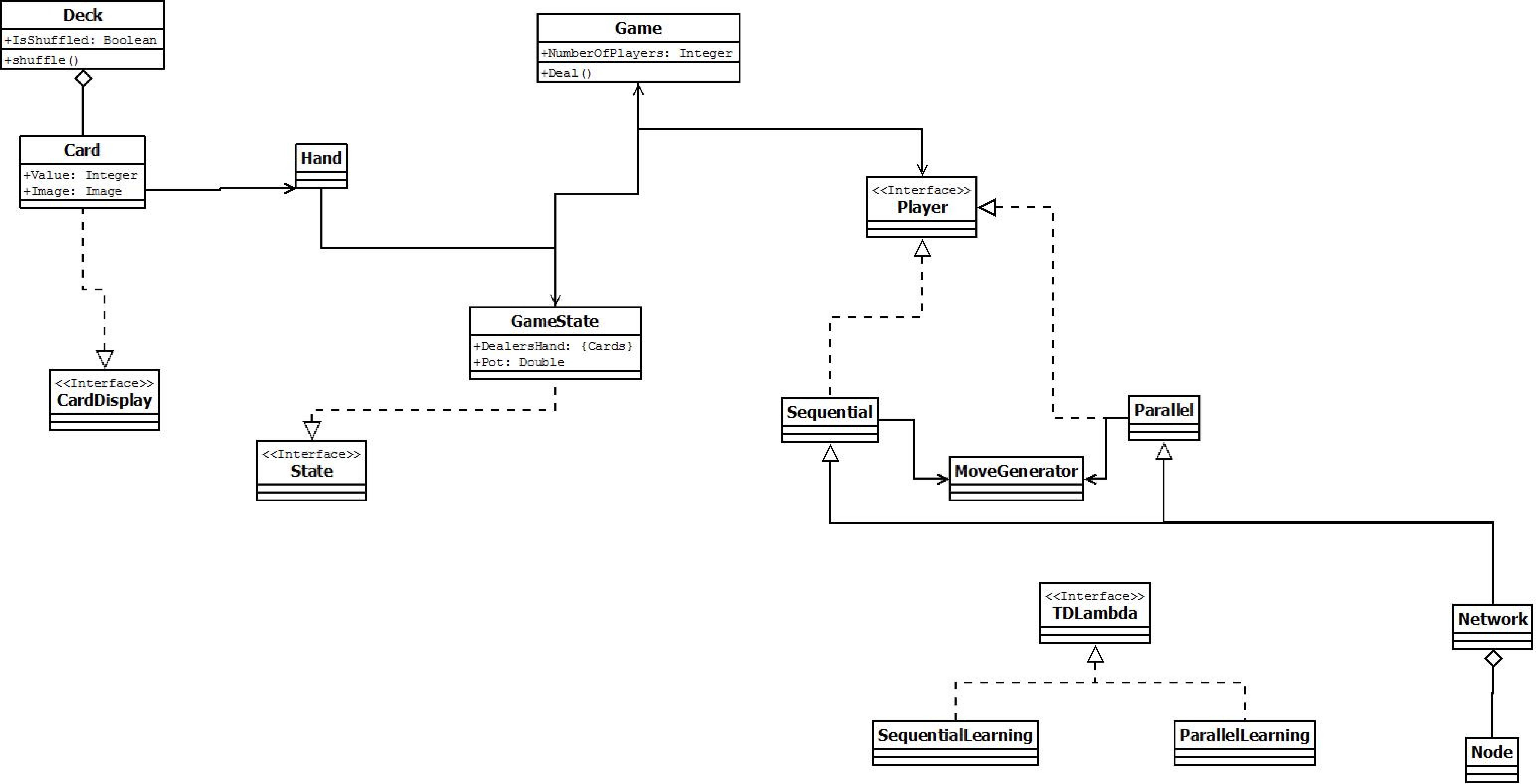
```
participant User
participant GraphicalFrame
participant GameController
User -> GraphicalFrame: displayRoundStatistics
activate GraphicalFrame
GraphicalFrame -> GameController: displayRoundStatistics
activate GameController
GameController -> GameController: getRoundData
GameController -> GameController: calculateStatistics
GameController -> GraphicalFrame: showStatistics(statistics)
GraphicalFrame -> User: showStatistics(statistics)
deactivate GameController
deactivate GraphicalFrame
```



Display Game Statistics Operation

```
participant User
participant GraphicalFrame
participant GameController
User -> GraphicalFrame: displayGameStatistics
activate GraphicalFrame
GraphicalFrame -> GameController: displayGameStatistics
activate GameController
GameController -> GameController: getGameData
GameController -> GameController: calculateStatistics
GameController -> GraphicalFrame: showStatistics(statistics)
GraphicalFrame -> User: showStatistics(statistics)
deactivate GameController
deactivate GraphicalFrame
```




Deck
+IsShuffled: Boolean
+shuffle()
Game
+NumberOfPlayers: Integer
+Deal()
Card
+Value: Integer
+Image: Image
Hand
<<Interface>>
Player
GameState
+DealersHand: {Cards}
+Pot: Double
<<Interface>>
CardDisplay
<<Interface>>
State
Sequential
Parallel
MoveGenerator
<<Interface>>
TDLambda
Network
SequentialLearning
ParallelLearning
Node