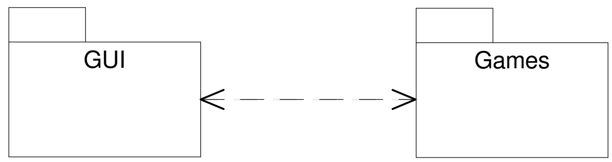
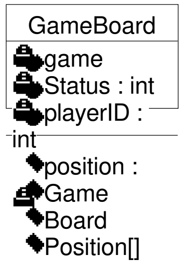
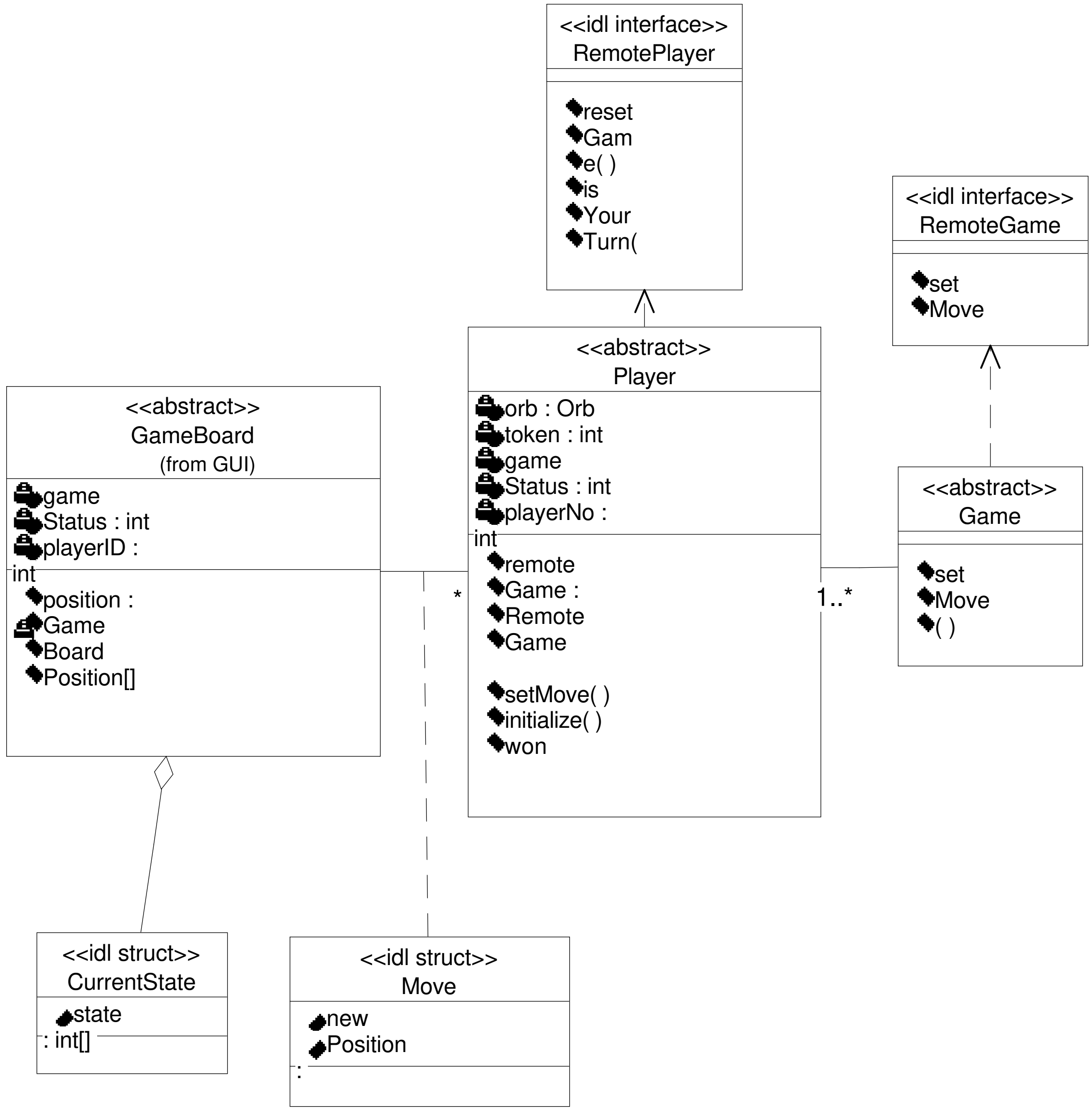
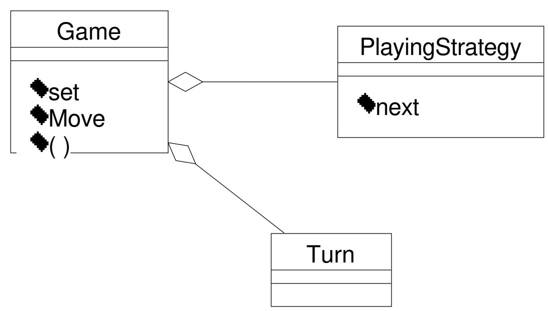
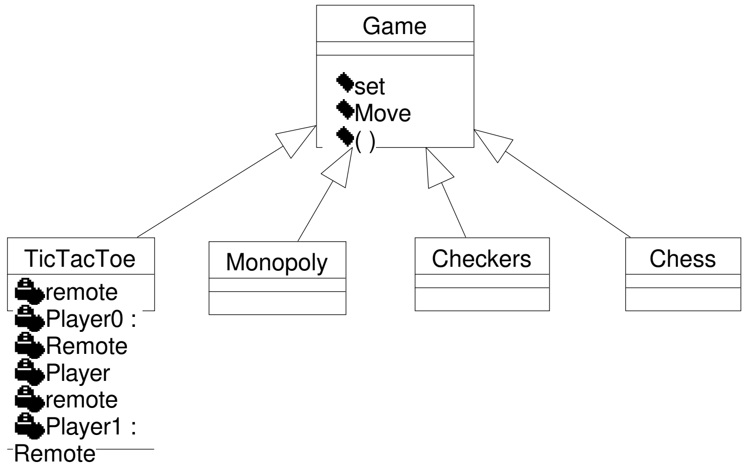

GUI
Games

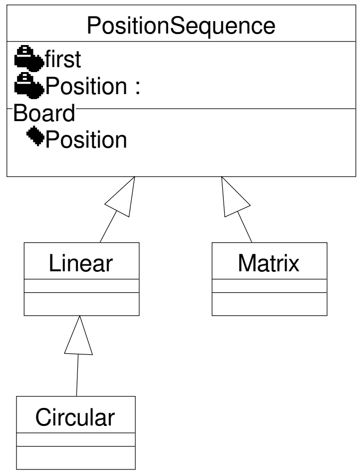

GameBoard
game
Status : int
playerID :
int
position :
Game
Board
Position[]

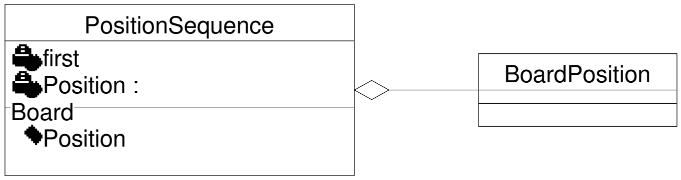


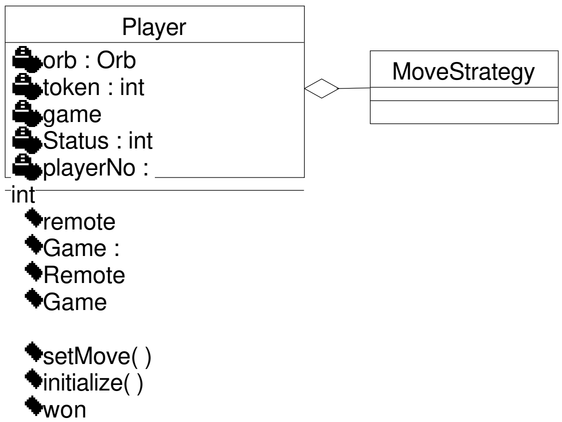
<<idl interface>>
RemotePlayer
resetGame( )
isYourTurn(
<<idl interface>>
RemoteGame
setMove
<<abstract>>
GameBoard
(from GUI)
gameStatus : int
playerID : int
position : Game
Board
Position[]
<<abstract>>
Player
orb : Orb
token : int
gameStatus : int
playerNo : int
remoteGame : RemoteGame
setMove( )
initialize( )
won
<<abstract>>
Game
setMove( )
*
1..*
<<idl struct>>
CurrentState
state : int[]
<<idl struct>>
Move
newPosition :


Game
set
Move
( )
PlayingStrategy
next
Turn

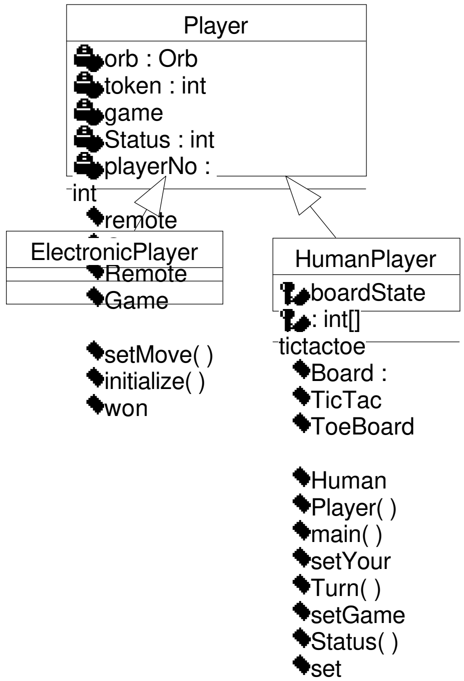

Game
set
Move
( )
TicTacToe
remote
Player0 :
Remote
Player
remote
Player1 :
Remote
Monopoly
Checkers
Chess

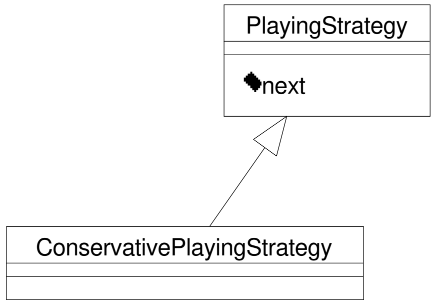

PositionSequence
first
Position :
Board
Position
Linear
Matrix
Circular

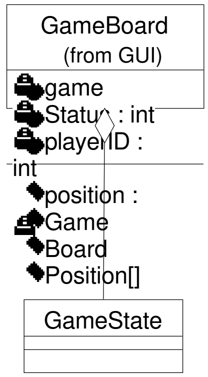

PositionSequence
first
Position :
Board
Position
BoardPosition

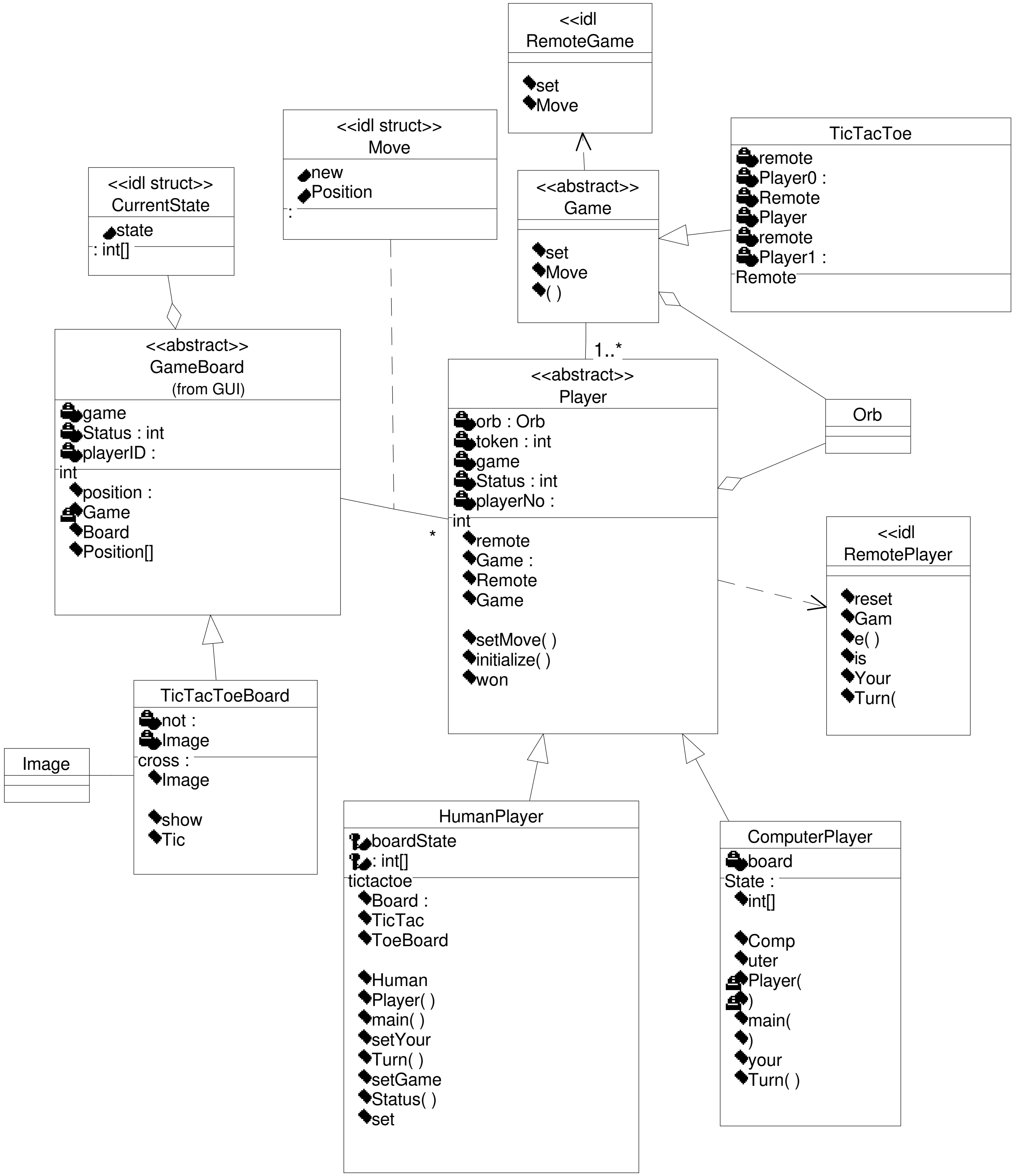


Player
orb : Orb
token : int
game
Status : int
playerNo : int
remote
Game :
Remote
Game
setMove( )
initialize( )
won
MoveStrategy

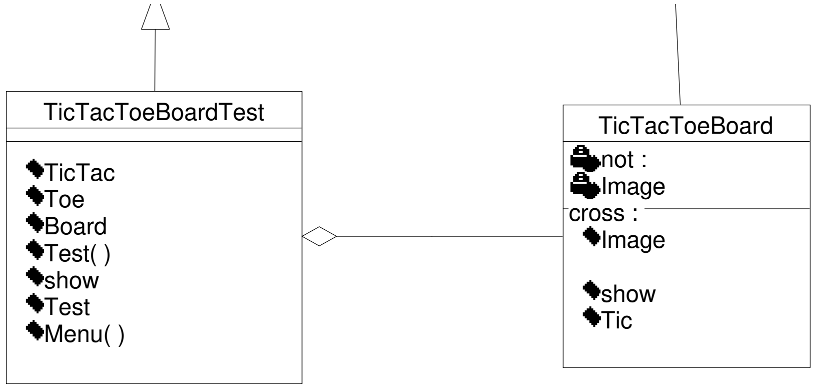

Player
orb : Orb
token : int
game
Status : int
playerNo :
int
remote
Remote
Game
setMove( )
initialize( )
won
ElectronicPlayer
HumanPlayer
boardState
: int[]
tictactoe
Board :
TicTac
ToeBoard
Human
Player( )
main( )
setYour
Turn( )
setGame
Status( )
set

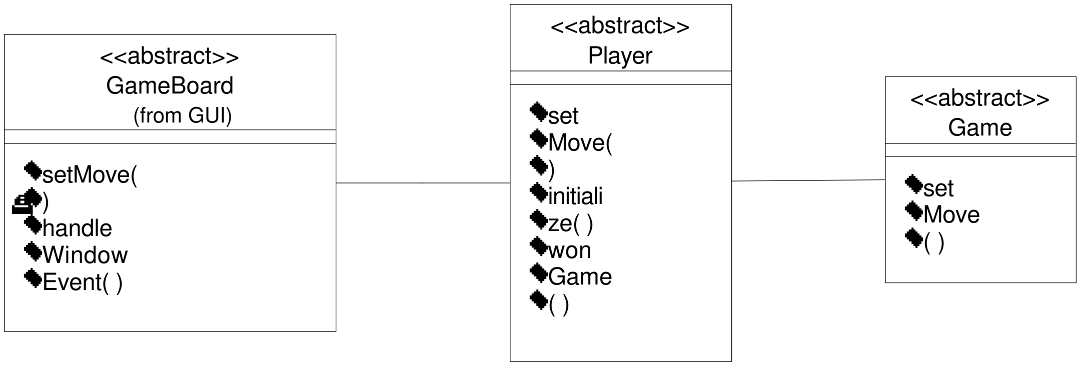

PlayingStrategy
next
ConservativePlayingStrategy

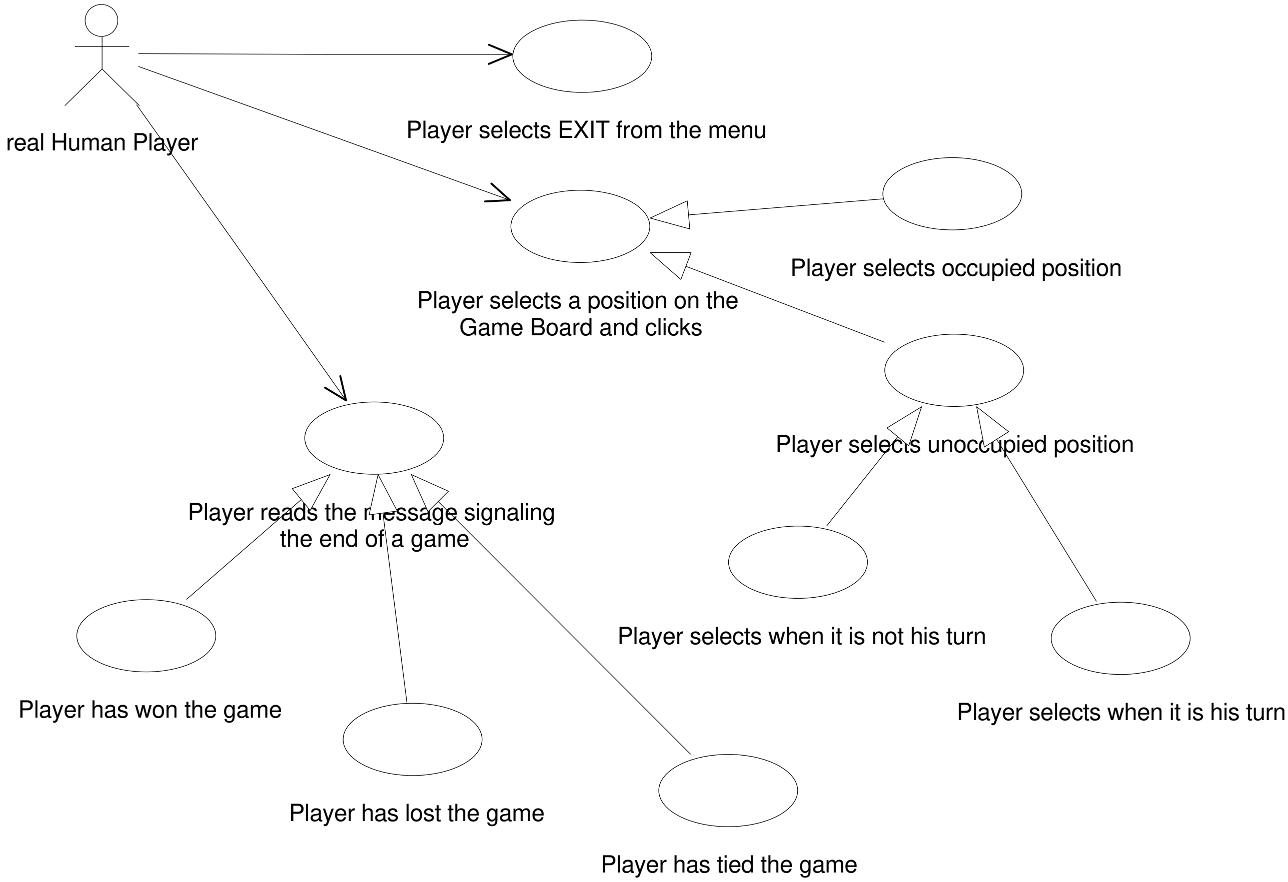

GameBoard
(from GUI)
game
Status : int
playerID : int
position :
Game
Board
Position[]
GameState

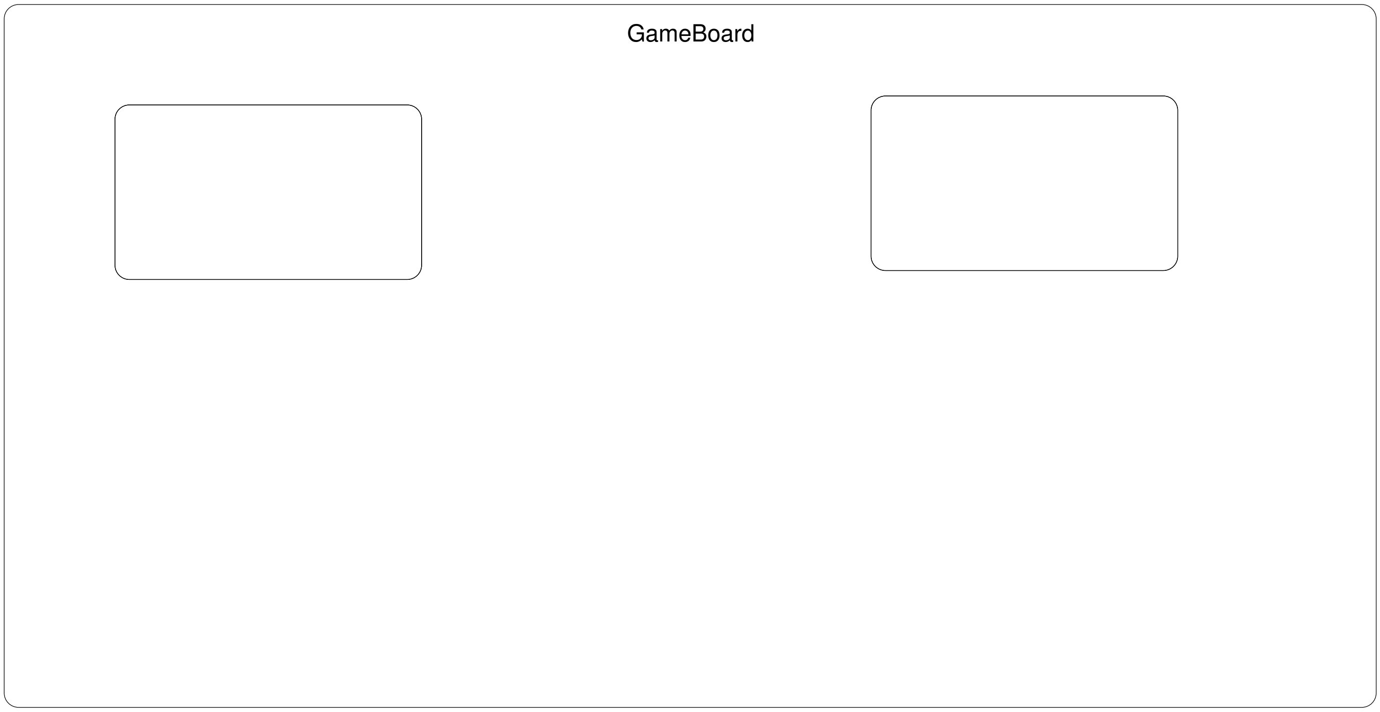

<<idl
RemoteGame
set
Move
<<idl struct>>
Move
new
Position
:
TicTacToe
remote
Player0 :
Remote
Player
remote
Player1 :
Remote
<<idl struct>>
CurrentState
state
: int[]
<<abstract>>
Game
set
Move
( )
1..*
<<abstract>>
GameBoard
(from GUI)
game
Status : int
playerID :
int
position :
Game
Board
Position[]
*
<<abstract>>
Player
orb : Orb
token : int
game
Status : int
playerNo :
int
remote
Game :
Remote
Game
setMove( )
initialize( )
won
Orb
<<idl
RemotePlayer
reset
Gam
e( )
is
Your
Turn(
TicTacToeBoard
not :
Image
cross :
Image
show
Tic
Image
HumanPlayer
boardState
: int[]
tictactoe
Board :
TicTac
ToeBoard
Human
Player( )
main( )
setYour
Turn( )
setGame
Status( )
set
ComputerPlayer
board
State :
int[]
Comp
uter
Player(
)
main(
)
your
Turn( )

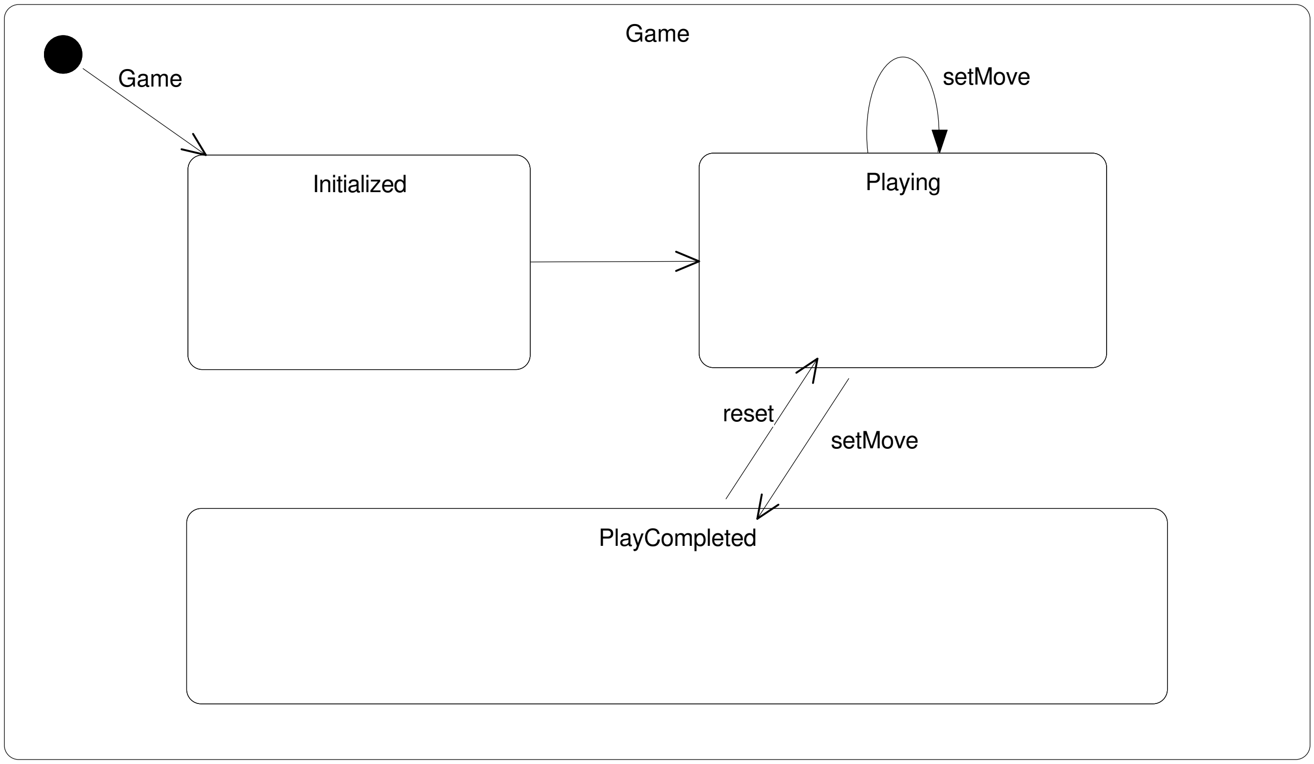

<<abstract>>
AbstractTest
function
alSuite(
)
structur
alSuite(
)
interacti
on
Suite( )
regress
ion
Suite( )
base
Line
Suite( )
class
Invarian
t( )
<<abstract>>
AbstractBeanTest
Abstract
BeanTest(
)
add
Vetoable
Change
<<abstract>>
GameBoardTest
show
Test
Menu( )
test
Dispatc
h( )
function
alSuite(
)
structura
lSuite( )
interacti
on
Suite( )
base
Line
<<abstract>>
GameBoard
(from GUI)
game
Status : int
playerID :
int
position :
Game
Board
Position[]

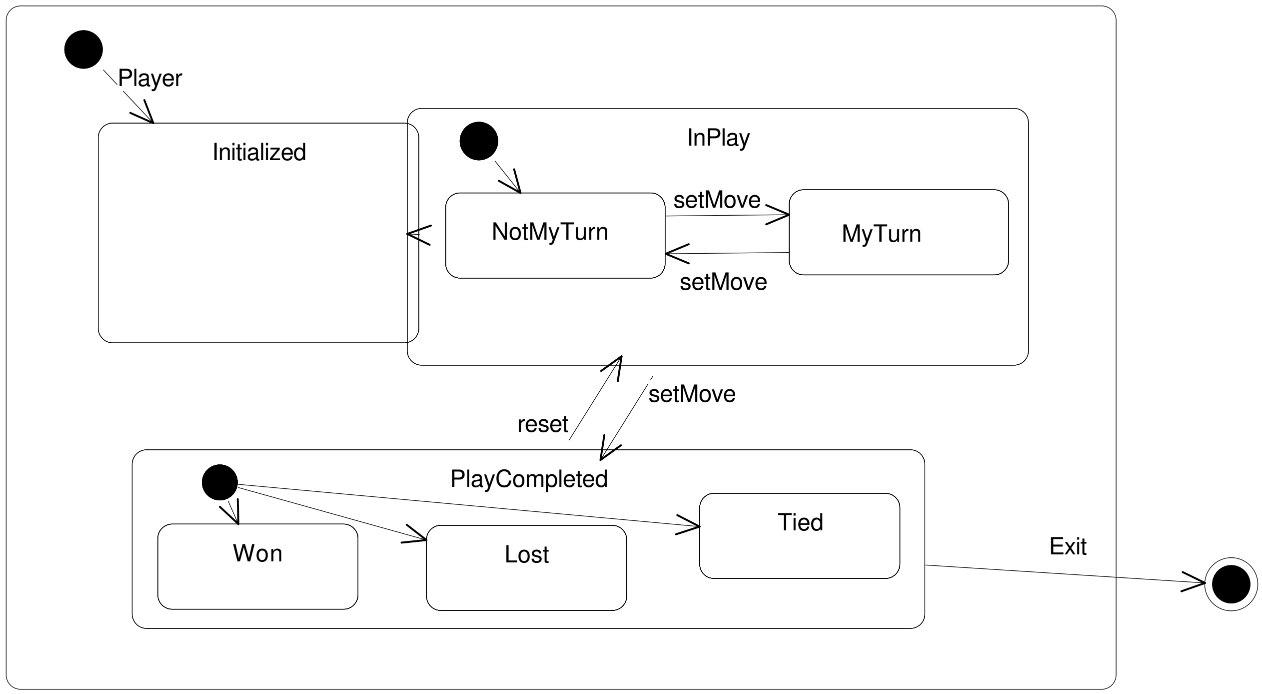

TicTacToeBoardTest
TicTac
Toe
Board
Test( )
show
Test
Menu( )
TicTacToeBoard
not :
Image
cross :
Image
show
Tic

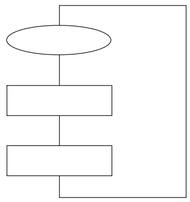

<<abstract>>
GameBoard
(from GUI)
setMove( )
handleWindowEvent( )
<<abstract>>
Player
setMove( )
initialize( )
wonGame( )
<<abstract>>
Game
setMove( )

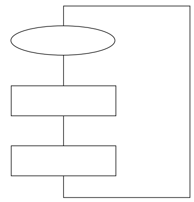

real Human Player
Player selects EXIT from the menu
Player selects a position on the Game Board and clicks
Player selects occupied position
Player selects unoccupied position
Player reads the message signaling the end of a game
Player selects when it is not his turn
Player selects when it is his turn
Player has won the game
Player has lost the game
Player has tied the game

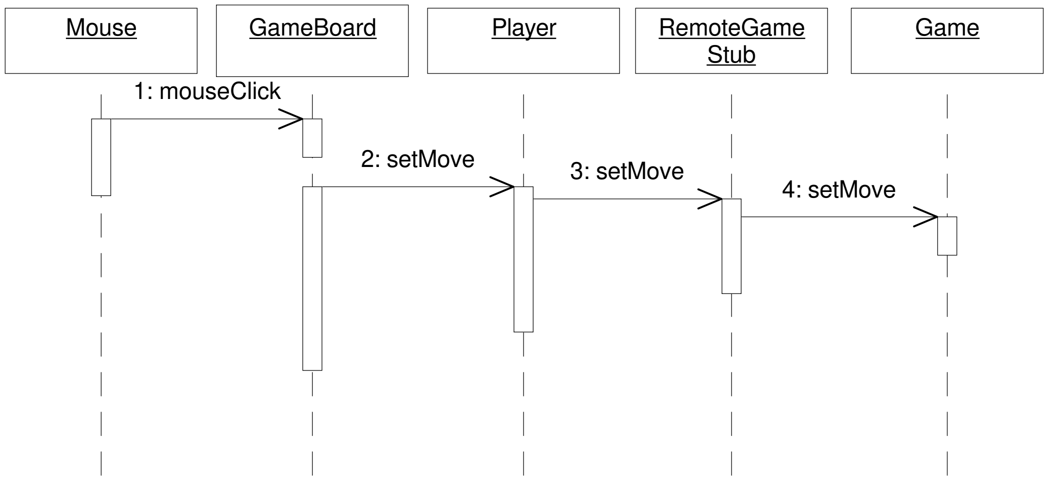

GameBoard

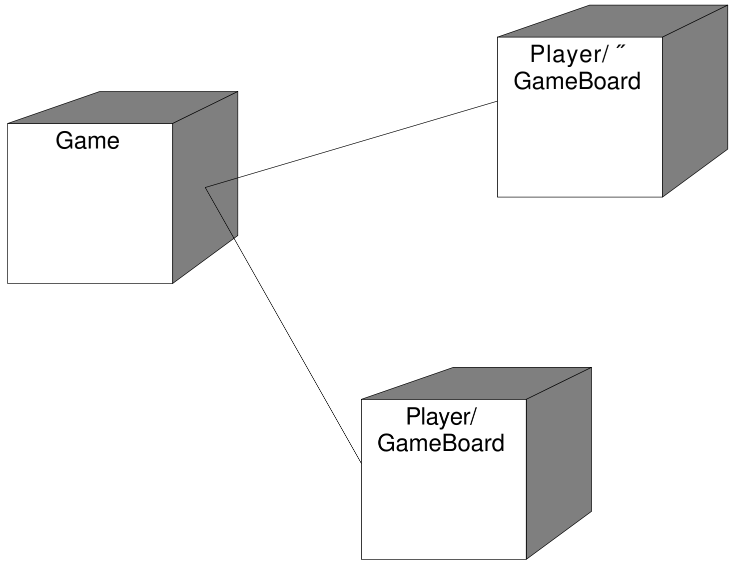

Game
Game
setMove
Initialized
Playing
reset
setMove
PlayCompleted


Player
Initialized
InPlay
NotMyTurn
setMove
MyTurn
setMove
reset
setMove
PlayCompleted
Won
Lost
Tied
Exit






Mouse
GameBoard
Player
RemoteGame Stub
Game
1: mouseClick
2: setMove
3: setMove
4: setMove


Player/ ˝
GameBoard
Game
Player/
GameBoard