Classes

**Board**

*Variables*

Pits (array of integers) [14]

*Methods*

**Distribute** (pit # {int}, board {class}, player)

extraTurn = False  
Creates a new Board, with initial values = the values of the input board. Sets the value of the pit pit # to 0, then sets each subsequent pit after for x pits, where x is the number pit # started with, to their value +1. If it ends in a Khalana, extraTurn = True. Returns (new Board, extraTurn)

**CheckVictory** (board {class})

If each value for 1 player on the input board are 0, then totals the values for each player (0-6 for player 1, 7-13 for player 2). If player 1 wins, returns (True, 1). If player 2 wins, returns (True, 2). If neither wins, returns (False, none). If it is a tie, returns (True, 3).

**NextTurn** (board {class}, player {int}, extra turn {Boolean}, invalid {Boolean})

Prints each value of the board, in this format:

“ Player 1

0 1 2 3 4 5

6 13

7 8 9 10 11 12

Player 2 ”

If not extra turn + not invalid: Prints “Player [player], enter the pit you would like to move from.”

If extra turn + not invalid: Prints “You ended in your khalana. Take an extra turn. Enter another pit.”

If invalid: “Invalid move. Please enter the number of a pit that has stones on it on your side.”

player = player

user\_input = player.Move(new board identical to the old one)

Checks user\_input validity (long set of if’s that check it is on the right player’s side and it is a number and the chosen pit has stones in it. If invalid, returns (None, None, True)

(newBoard, newTurn) = Board.distribute(user\_input, new board identical to input board, player)

Return (newBoard, newTurn, False)

**Player (Superclass)**

*Variables:*

Number = integer

*Subclasses:*

HumanPlayer, AIPlayer

*Methods:*

Move

**HumanPlayer**

*Variables:*

*Methods:*

**Move** (Board)

pit = user input

return pit

**AIPlayer (Not in use yet)**

Program Overview

* MainBoard = a new Board object with 3 as the value for each non-6, non-13 value of pits, 0 for 6 and 13
* Player = 0
* ExtraTurn = False
* Invalid = False
* gameOver = False
* While not gameOver:
  + (newBoard, extraTurn, invalidMove) = Board.NewTurn(MainBoard, Player, ExtraTurn, Invalid)
  + ExtraTurn = extraTurn
  + Invalid = invalidMove
  + If invalid == true: continue
  + MainBoard = newBoard
  + If extraTurn == true: continue
  + (game\_over, victor) = Board.CheckVictory(Main Board)
  + If game\_over == true and victor == 3: prints “The game ends in a tie.”
  + Elif game\_over == true and victor == 1: prints “The game is taken by player 1, %d to %d.” %player 1 score (value of MainBoard.pits[6]), player 2 score (value of MainBoard.pits[13])
  + Elif game\_over == true and victor == 2: prints “The game is taken by player 2, %d to %d.” %player 2 score (value of MainBoard.pits[13]), player 1 score (value of MainBoard.pits[6])