**Information:**

1931 Salvo Edition:  
In the modern Milton Bradley rules for *Battleship*, *Salvo* is listed as a variation "for more experienced players", with the following rule(s)

* players target a specified number of squares at one time, and all squares are attacked simultaneously. The number of shots a player can fire each turn may either:
  + be equal to the number of ships that the firing player has remaining
  + be fixed at five for the whole game
  + be equal to the size of the player's largest undamaged ship
* The receiving player may either call
  + the result of each shot in turn
  + or simply announce the number of hits and misses (ex: "two hits and three misses"), leaving their opponent to work out the consequences of the salvo
* One variant of Battleship allows players to decline to announce that a ship has been sunk (so the opponent takes further shots to confirm that an area is clear)
* Another house rule allows a player to move 1 of their ships to a new, UNCALLED location every fourth or fifth move

**Variables Received:**

gameType = 1 or 2 (1 = able to fire at only 1 target per turn, 2 = able to fire at multiple)

gameOver = 0, 1, or 2 (0 = game in progress, 1 = player1 won, 2 = player2 or computer won)

attackLoc[5][2] = 5 rows (max num of attacks), 2 columns (X/Y Grid)

primaryGrid & trackingGrid data (explained below)

**Variables Returned:**

primaryGrid data

trackingGrid data

**PsuedoCode:**

Class player:

shipHp[2 – ?(max = 6)]:

0 = num of times player can fire (controlled by gameType and set at beginning of game)

1, 2, …, ? = the status/health of ship’s individual coordinate hp (initialized as 1 at the beginning of the game)

If any of the ship health equals zero, Then the ship is sunk & the user is shown the ship that was sunk, Else no change

primaryGrid[10][10] = shows where the player’s ships are and where the opponent has fired shots at

trackingGrid[10][10] = shows where the player thinks the opponents ships are

ship?[2][?] = where the player’s ships are, 2 columns (X/Y Location)

Functions:

def init(self) = initialize the variables and grids to their initial value or None

def translate(self, s (string) ) = translate inputs received to be used for logic computing

def inputFireLoc(self, fireInputs[] (list) ) = get inputs on where the player wants to fire shots at, translate the inputs into firelock

def setShips(self) =

translate inputs received

use inputs to set the player’s ships in primary grid (called once at beginning of the game)

set gameType depending on gameMode variable received (salvo == 1, classic == 0)

initialize the number of shots player can fire depending on gameMode(salvo = 5, classic = 1)

def fire(self, object) =

fires a shot (once per FOR loop, repeated up to how many times the player can fire at that point in the game)

check if shot is a hit or a miss

check if opponent ship has been sunk

update the player’s tracking grid and the opponent’s primary grid

update how many times the opponent can fire(if salvo)

def isSunk(self, object) =

check how many ships has been sunk and count them

IF the gameType == 1, Then update the number of times the opponent can fire shots to 5 – count