Battleships Technical Specification

Group Project

Git Repo: <https://github.com/DrGrimshaw/BattleshipsJavaGame>

**Ship**

State

* Length
* Health
* Starting Position
* Orientation

Behaviours

* Construct with length, starting position and orientation
* Get length
* Get starting co-ordinates
* Get ending co-ordinates
* Get an array of all co-ordinates
* Get orientation
* Check whether ship exists at co-ordinates
* Take a hit
* Check whether the ship is sunk

**Cell State**

* Nothing
* Miss
* Ship Hit
* Ship not hit
* Ship Sunk

**Player Board**

State

* A board has a grid of cell states
* A board has a list of ships
* A board has a width and height

Behaviours

* Create a board with a width and height
* Get width
* Get height
* Place a ship on to board
* Get cell state at co-ordinates
* Check whether co-ordinates have been guessed before
* Take a hit at co-ordinates, returning the new cell state
* Get ship at co-ordinates if any
* Get number of ships remaining
* Get an unmodifiable list of ships
* Check for game over

**Enemy Board**

State

* Board has a grid of cell states
* Board has width
* Board has height
* Number of ships remaining

Behaviour

* Construct an enemy board with… width, height, number of ships
* Update state of cell at co-ordinates – if battleship sunk, reduce number of ships remaining by one
* Get number of ships remaining
* Check whether co-ordinates have been guessed before
* Get cell state at co-ordinates, returning “nothing” in place of “ship not hit” and “ship hit” in place of “ship sunk”
* Get Width
* Get Height

**Game Over Message**

- You won

- You lost

- You resigned

- Your opponent resigned

**View**

Behaviours

* Print the cell state of the player and enemy boards to the user
* Print welcome message/game setup message to the user
* Print game over message
* Print who has won
* Print instructions to user
* Print result of the move that has been made
* Print types of ships left to place (during setup)

**Player**

State

* Player board
* Enemy board

Behaviours

* Construct a player (...)
* Choose a position and orientation for a ship
* Choose coordinates to fire on (if player has resigned throw player resigned exception)
* Take a hit at coordinates returning new cell state
* Check whether game is over
* Check whether co-ordinates have been guessed before by this player
* View this player’s state with a view
* Report the result of a played move, to update the enemy board

**TextView** *implements View*

**Human Player** *implements Player, using console I/O*

**Computer Player** *implements Player, choosing randomly*

**Game**

State

* Two players

Behaviour

* Get the next player
* Play
  + *While there is no winner:* get the next player, ask the player to make a move, check with other player whether hit/miss/sunk, report result back to player

**Program**

**Network Program Protocol**

Place ship on player board