To: Jason^2
From: Jason
Date: September 20, 2018
Subject: Santorini

The Santorini Game will all be controlled within a TCP server that represents a tournament. It will have a list of current Game objects and a map of all Games played to the winning Player to determine an overall winner in the end. Clients are able to connect to the server and each client gets a Player object tied to their Socket. The server will keep score of each client to assign an overall winner. Inside each Game, there are references to 2 Player objects. Clients will send requests to their Player to be parsed and regulated within the server. Each client will not be able to directly change their current Game object.

The Game object will regulate which player's turn it is. When it's a Player's turn, the Game calls *reqMove()* on the Player. Inside the Player, the client is then prompted for a request which *reqMove()* will handle the parsing and turn it into an IAction object. An IAction object can be a Move or Build request. This object is returned to the Game object and Game will execute the given object. Here, the game will first check if it is a valid IAction. If it is not, the Game will prompt the user for a new action. If it is, the Game will execute the action and the action will update the Game.

A Game will have 2 Players, a 2D Array of Squares, an Array of Workers, and a boolean flag representing whose turn it is. Each square represents a cell in the grid. Each has a height; when the height is 0, this represents no building on the cell. From 0, each Square can add one floor at a time until the height is 4. Each Player will have reference to their 2 Worker objects so that they can be included in the IAction objects.

The following page describes specific implementation necessities for each class.

Worker(Square square):

- *void Move(Square):* moves worker to given Square
- *boolean canMove(Square):* is true if Square is adjacent, is only 1 height higher, less than 4 floors, and no worker is there
- *void addFloor(Square):* adds floor to given Square
- *boolean canAddFloor(Square):* checks if can add a floor to Square

Square(int row, int col, int height):

- *void addFloor():* adds a floor to this Square
- *boolean canAddFloor(Worker):* checks if this square is adjacent to worker and if it has less than 4 floors

Player(Socket socket, Array<Worker> workers):

- *IAction reqMove():* Returns an IAction object, prompts the client to send a request and will pass the client input to parseRequest(request) within the Player class and returns the output
- *IAction parseRequest(String request):* parses the given string and returns the appropriate IAction object, either a Move or Build object

Move(Worker w, Player p, Square target) implements IAction:

- *void execute(Game game):* checks if the given worker is able to move into the given target square. If it can, updates the Game , if it can't, throws an error and the Player is prompted for another move

Build(Worker w, Player p, Square target) implements IAction:

- *void execute(Game game):* checks if the given square can have a floor added given the worker. If it can, updates the game, if it can't, throws an error and the Player is prompted for another move

Game(Player p1, Player p2, boolean turn, Array<Array<Square> board, Array<Worker> workers):

- *boolean gameOver():* if a worker is on a 3rd floor, if a worker cannot move, if a worker cannot build, or if a player disconnected: returns true

- *Player determineWinner():* evaluates the board to return the winner