To: Matthias Felleisen
From: Gino Jacob and Emily Miller-McGlone
Subject: Santorini

A game of Santorini consists of two players matched according to number of games played and their victory rate (wins / total games). The player with the lower win rate places and moves their workers first. Otherwise, they are randomly selected.

Communication between Santorini and the players is one sided. The former holds the state of the game, and the latter simply inform their strategies—where to move/build—when prompted and given the current state of the game. Santorini can execute a player's move, and tracks each cell on the board, where each player's workers are, whose turn it is, and what rules are to be imposed. Players know their ids, win to total games ratio, the ids of their workers, and what move to make given the state of the current game.

Santorini understands when to invoke invalid move exceptions to the players. Players know how to respond when re-prompted for a move given the context of the exception.

Key pieces of the game include Santorini, Player, MoveError, BuildError, and Cell, which can represent either a Floor or Worker. Every Santorini game is automated and isolated to two players interacting with each other. A SantoriniServer is responsible for matching players up in games and running them concurrently.

Santorini Zero-indexed list of (*Cell*, level: number)	*Cell* := *Floor* | *Worker*
Worker id: Int coordinates: (x, y) player: *Player*	*Player* id: Int wins: Int total_games: Int prompt(state: 2D list, worker1: (x, y), worker2: (x,y)) except(error: *Error*, state: 2D list, worker1: (x, y), worker2: (x,y))
Floor is an empty class	*Error* := *MoveError* | *BuildError*
MoveError := *BlockingWorker* | *FloorUnreachable* | *OutOfBounds* | *CoordinatesUnreachable* | *StandingOnCoordinates* | *FloorFour*	*BuildException* := *BlockingWorker* | *FloorUnreachable* | *OutOfBounds* | *CoordinatesUnreachable* | *StandingOnCoordinates* | *FloorFour*

Player's *prompt* and *except* return the player's worker number they want to move, an enum: *Direction*, and enum: build position. The method except returns the player's worker number they want to move, an enum: direction, and enum: build position. The *Direction* and build position enumeration has members: N, S, W, E, NW, NE, SW, and SE.