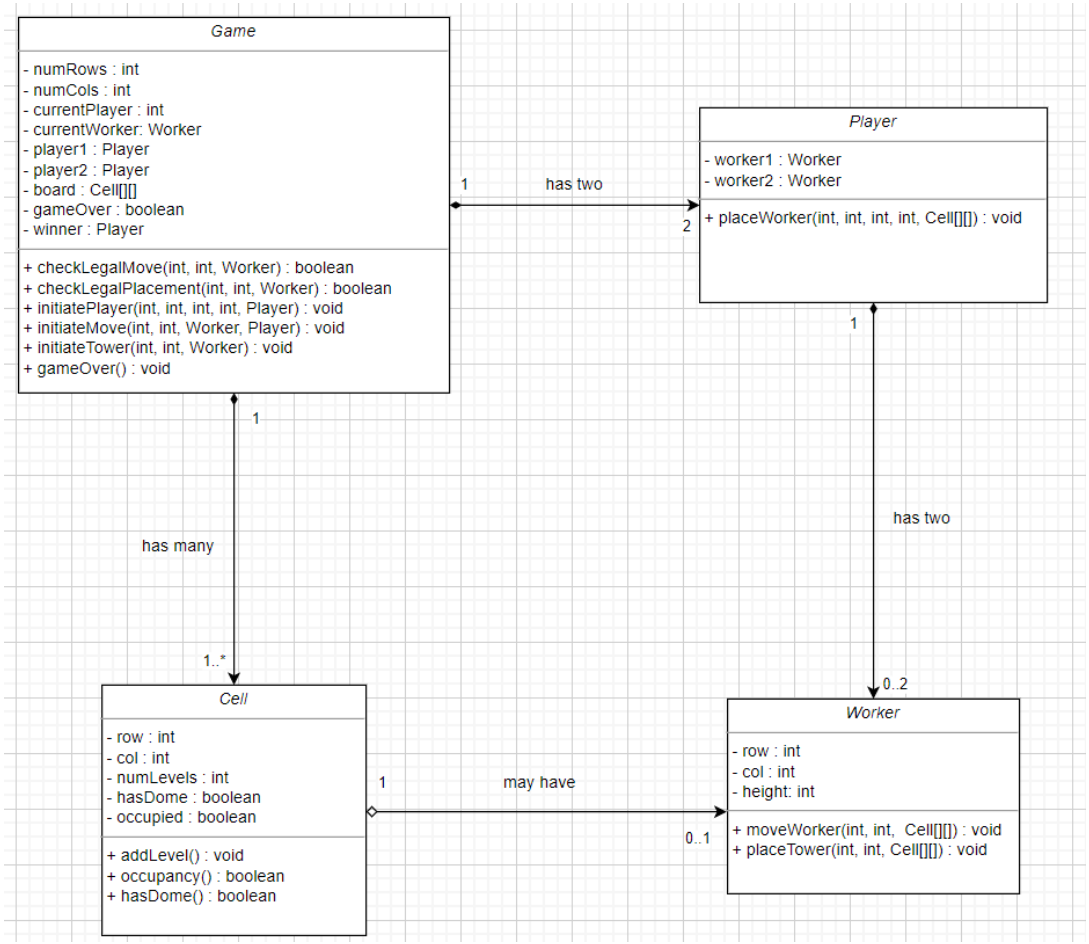
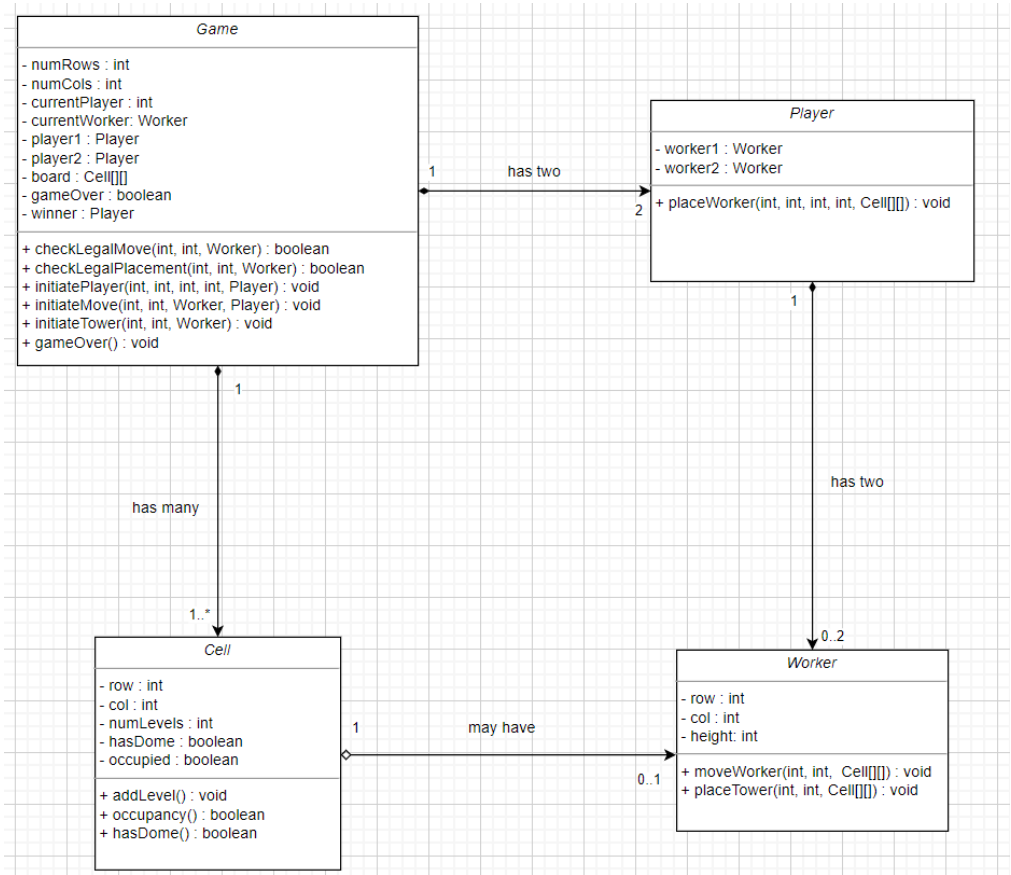
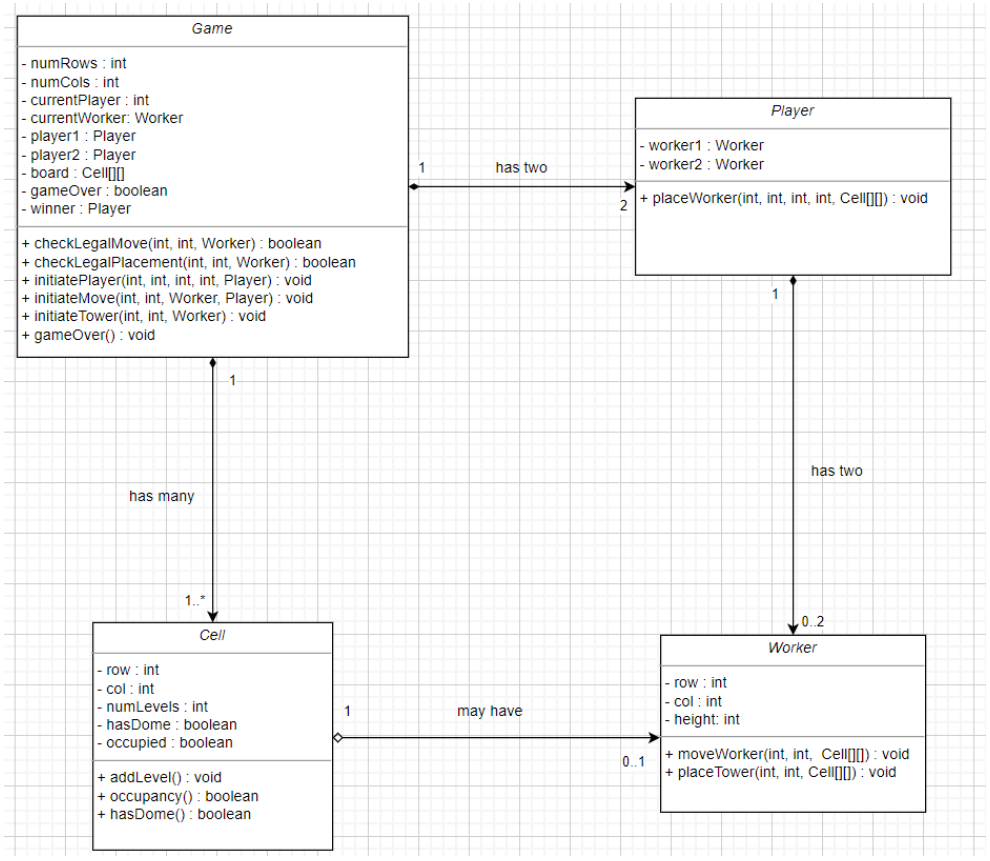
1. How can a player interact with the game? What are the possible actions? Please include necessary parts of the Object Model to explain.
In my implementation, the way that the player interacts with the game is through the Game class. My Game class does not have much, in terms of functions, but it serves as a medium to “connect” everything; it acts as a controller for the rest of my classes (cell, worker, player). I decided that the path of a Game controller, which the user interacts with to access all of the core functionality of playing the game is the most ideal in my opinion since allows me to keep the rest of my classes fairly independent from each other and group functions that are closely connected which follows the principles of high cohesion and low coupling. In Game, you can initiate each player using initiatePlayer which keeps track of their two workers. Once these initial players are created, the game can begin asking each player to take turns moving their worker using initiateMove, and placing towers using initiateTower. The game ends when one of the players’ workers has a height of 3 (when they stand on a 3 level high tower). My initiate functions check the validity of a move, so given a row and a col, it determines whether the resulting location is valid or not according to the rules (so I throw exceptions for things like out of bounds, trying to place a worker or tower on top of an existing worker location, etc.)

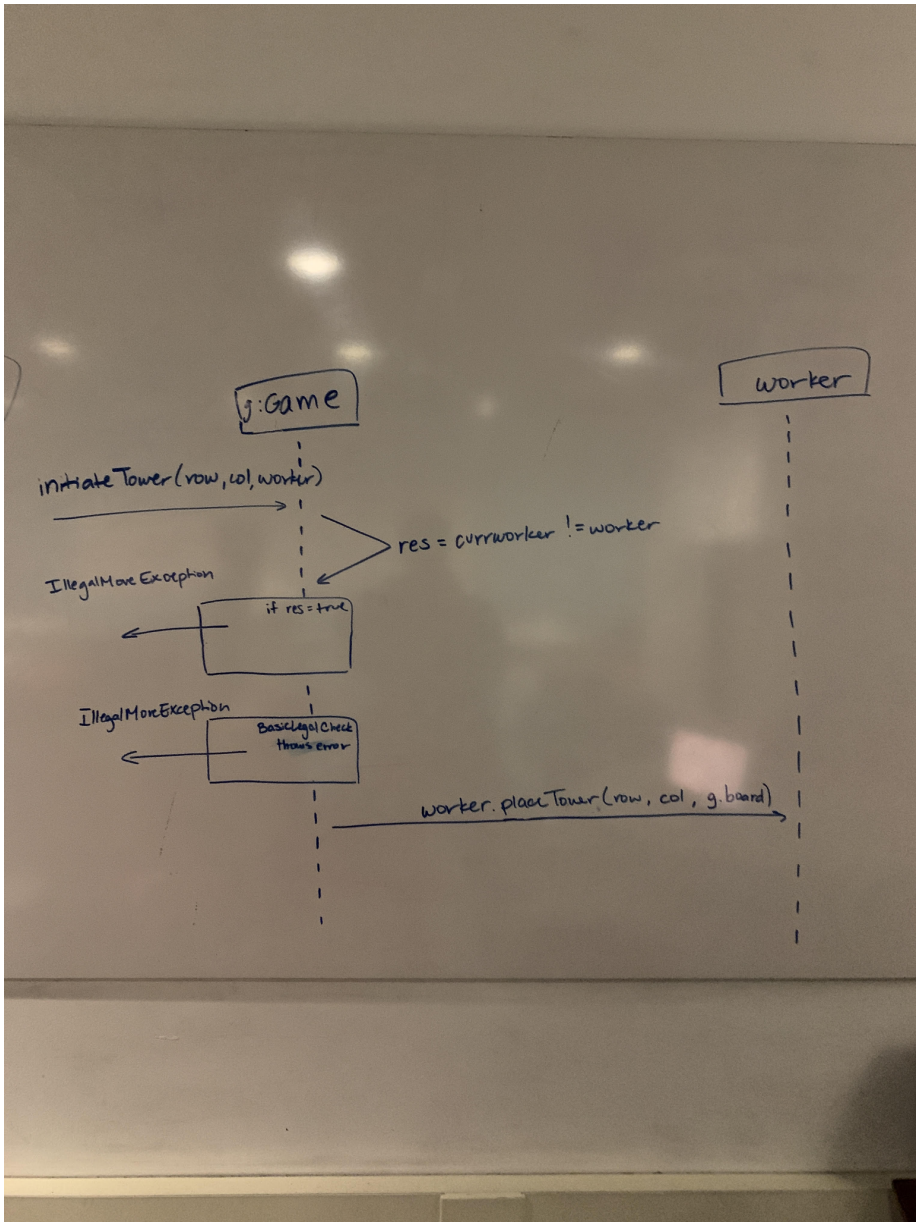


2. What state does the game need to store? And where should is be stored? Please include necessary parts of the Object Model to explain.
My Game stores a lot of information, mainly because it is the central point of the whole program. While it doesn't have implementations of core functionality of the cell, worker, or player class, it delegates the work to these classes. I keep track of a lot of private variables in my Game class such as currentPlayer, player1, player2, currentWorker, gameOver, winner, and the board. I use currentPlayer to keep track of whose turn it is and currentWorker is used to keep track of which worker was moved (for tower placement since we can only place towers adjacent to a recently moved worker). I also keep track of the board since this allows me to access specific values on the board. Lastly, I have my winner and gameover which helps me determine a winner and end the game. These are all located in my Game class, however I also keep track of some things in my other classes. In my cell class I keep track of the row, col, numLevels, hasDome, and occupancy which are really helpful in keeping everything organized as well as keeping track of important information of a cell (especially for checking validity later on). In my worker, I keep track of row, col, and height so that I can call it later to check if the worker is in a winning position (height). And last, I store the workers in the player class so that I can keep track of which worker belongs to which player.



3. How does the game determine what is a valid build (either a normal block or a dome) and how does the game perform the build? Please include necessary parts of an object-level Interaction Diagram and the Object Model to explain.
The game determines what is a valid build based on functions that check legality of a move. So checkLegalMove checks if the provided row and col are adjacent to the worker if the row and col are either the same height or their difference is less than 2. It also checks if the row and cols are in bounds and if there are workers/domes on the cell (which is checked by worker.occupancy and worker.hasDome - any height > 3). The game performs the build when I call initiateTower which does the series of checks and if it passes all legality checks, then we call the worker function placeTower which directly modifies the game board.




g:Game
worker
initiateTower(row, col, worker)
res = currworker != worker
IllegalMoveException
if res = true
IllegalMoveException
BasicLegalCheck
throws error
worker.placeTower(row, col, g.board)