justification

1. How can a player interact with the game? What are the possible actions?

A player could have a series of actions to interact with the game: set, move, build. A player have 2 workers which are stored in player's array.At the beginning of the game, each player needs to set their workers on the available space on the board in turn. At every turn, the player **must** move one of workers and then build the tower (the same worker), which could be block or the dome(but according to the rules, dome only can be built on the third floor).

In summary, there are 3 possible actions for a player during this game: set, move and build.

2. What state does the game need to store? Where is it stored? Include the necessary parts of an object model to support your answer.

The game needs to store the players, current players, worker locations, towers, winners, round numbers, grids.

Game state:restore the **currentPlayer**,the **round numbe**r of the game as well if the game is still in process(**ifEnd**).

Player state: the player id, 2 workers associated with him. If all his workers are not **movable** anymore, the **winning status**.

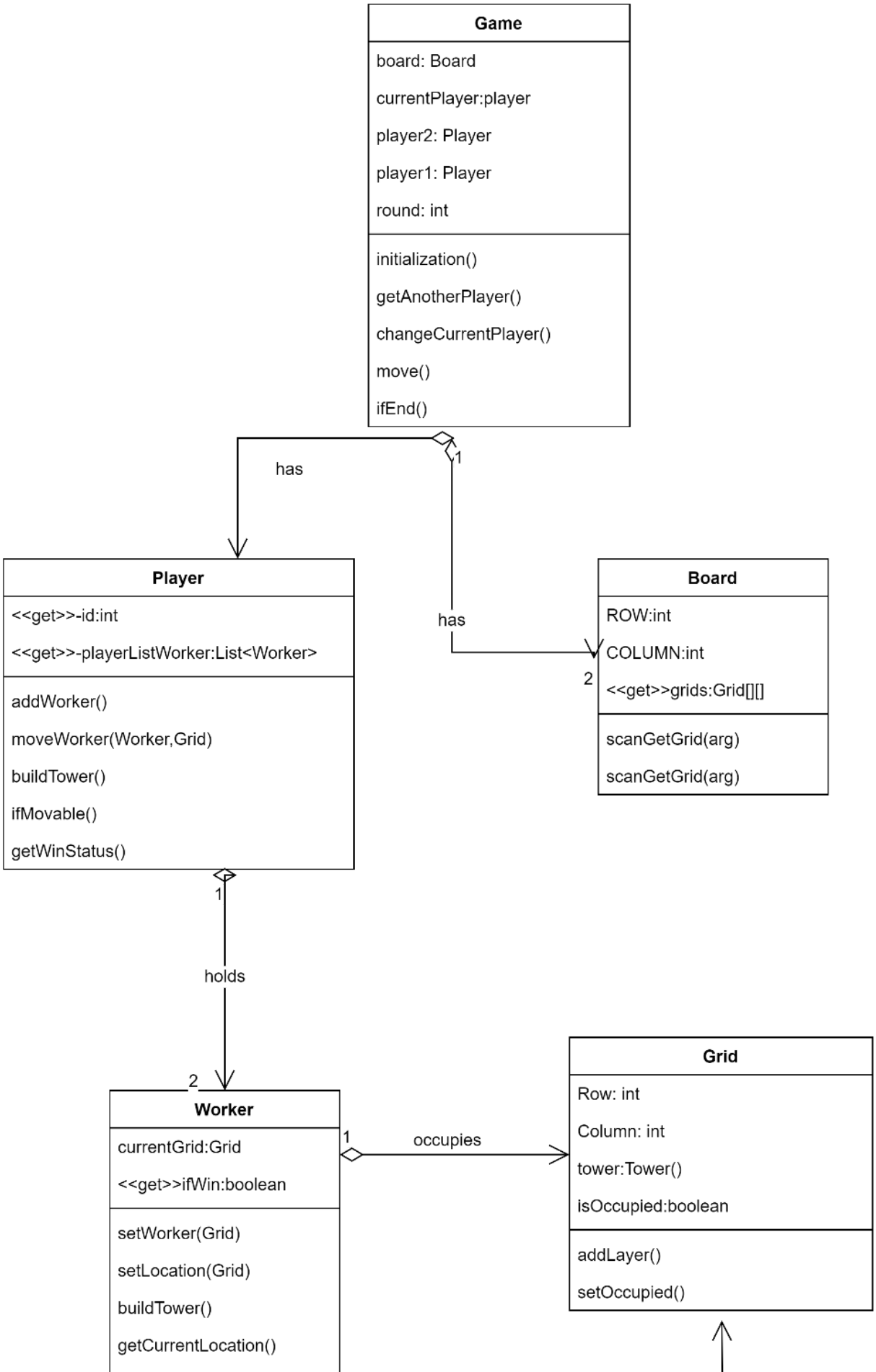
Worker state: the grid the worker is located. The available grid for the worker to move.

Winning status, movable grids

Board: grids on the board. A grid map.

Grid: **location,is occupied,towers**

Tower**: layers, ifDome**


Game
board: Board
currentPlayer:player
player2: Player
player1: Player
round: int
initialization()
getAnotherPlayer()
changeCurrentPlayer()
move()
ifEnd()
has
1
Player
<<get>>-id:int
<<get>>-playerListWorker:List<Worker>
addWorker()
moveWorker(Worker,Grid)
buildTower()
ifMovable()
getWinStatus()
has
Board
ROW:int
COLUMN:int
<<get>>grids:Grid[][]
scanGetGrid(arg)
scanGetGrid(arg)
2
1
holds
2
Worker
currentGrid:Grid
<<get>>ifWin:boolean
setWorker(Grid)
setLocation(Grid)
buildTower()
getCurrentLocation()
1
occupies
Grid
Row: int
Column: int
tower:Tower()
isOccupied:boolean
addLayer()
setOccupied()

Other alternatives:

In order to achieve the balance of low coupling and high cohesion, I have these alternatives and made some trade-offs between them.

At first, I store the tower easily in the grid as layer and test if build in grid, but finally I found this makes grid too many functions and responsibilities and is not proper.

Actually, the UI part in my program took a lot of responsibility at first, but then I found this makes the implementation difficult and useless. This is more like a simple main method to put all things into that and makes it hard to test. Then I divided them into different objects.

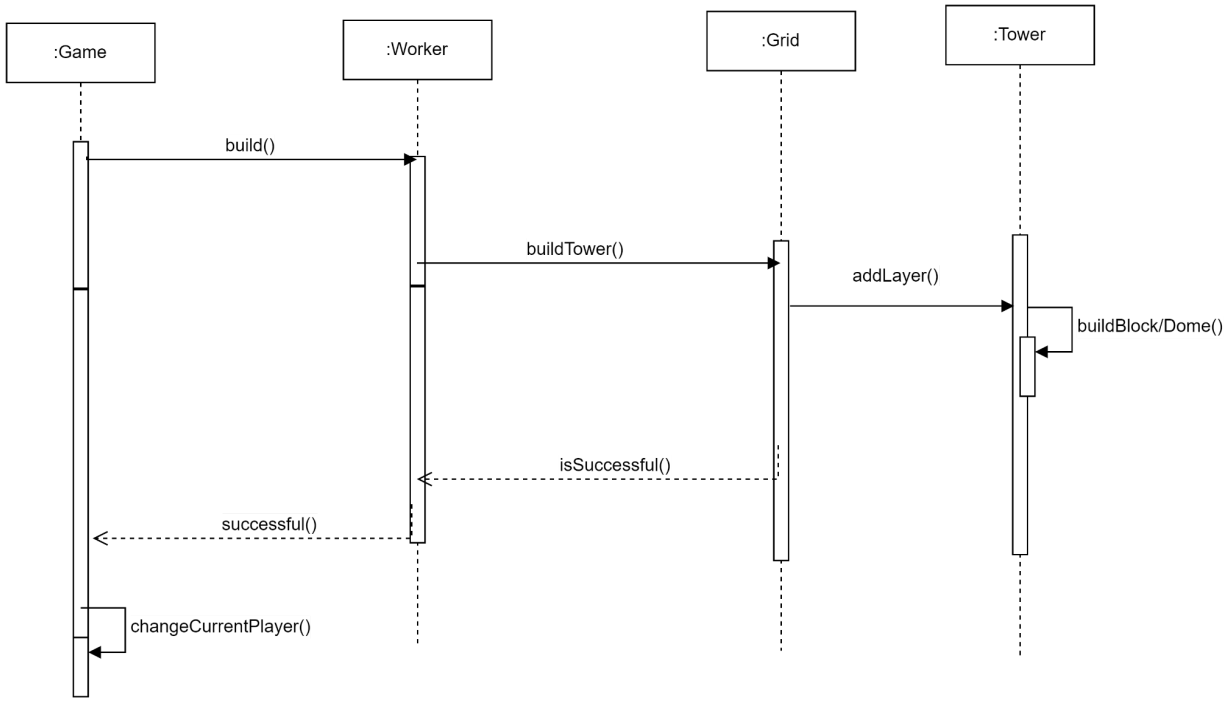
Since the UI required system.in and System.out, it is hard to divide them into different parts.

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? Include the necessary parts of an object-level interaction diagram (using planned method names and calls) to support your answer.

Valid building blocks: checkIfValidBuild(Grid grid, Grid current grid): determine if the block location is a valid location(which means the (row of the block - row of the moved worker) <=1 and >= -1 and (column of the block - column of the moved worker) <=1 and >= -1. What's more, the layer should be less than 3, which means there is no dome on it. And thus, this is a valid building action, this grid could not be occupied as well.

Since the interactive part is aiming at providing the process-flow, this is related to converting information between objects.

When player is going to build a tower, the worker need to be the worker who moved so this is picked in the previous steps. Worker would build tower on the grid, and grid would call Tower to build on it/ whether a block or a dome. Then, the tower started to return the building status to the grid and then return to the worker, the game and then decide the next step of the game.


:Game
:Worker
:Grid
:Tower
build()
buildTower()
addLayer()
buildBlock/Dome()
isSuccessful()
successful()
changeCurrentPlayer()