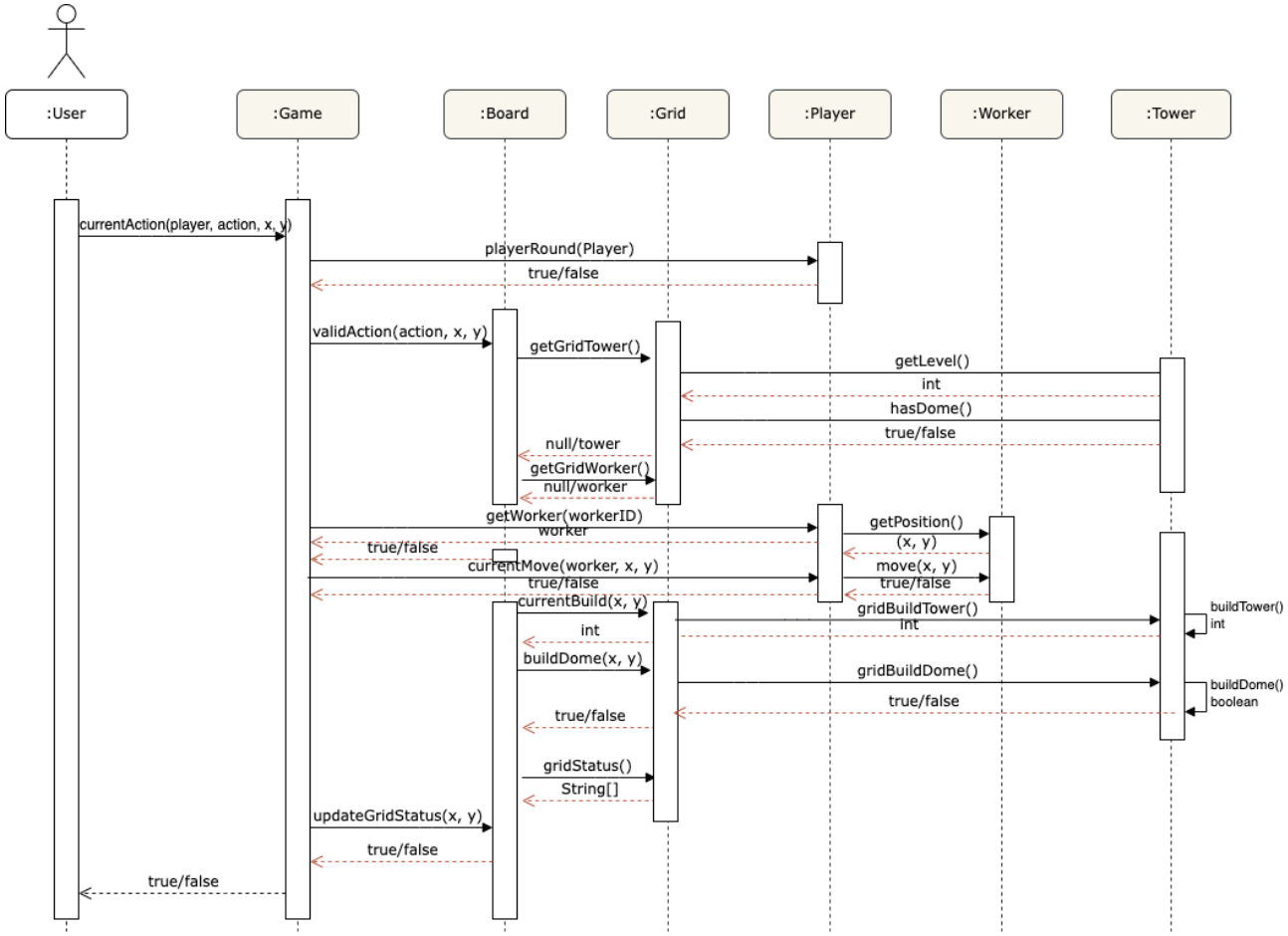
Deliverable 4: Justification. Write a short report that answers the following questions to justify your design choices. To receive full credit, each of your answers to these questions must both:

- Refer to design goals, principles, and patterns where appropriate.
- Discuss the alternatives you had considered and the trade-offs they entailed that led you to choose this particular design (essentially, your design process).

Questions:

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

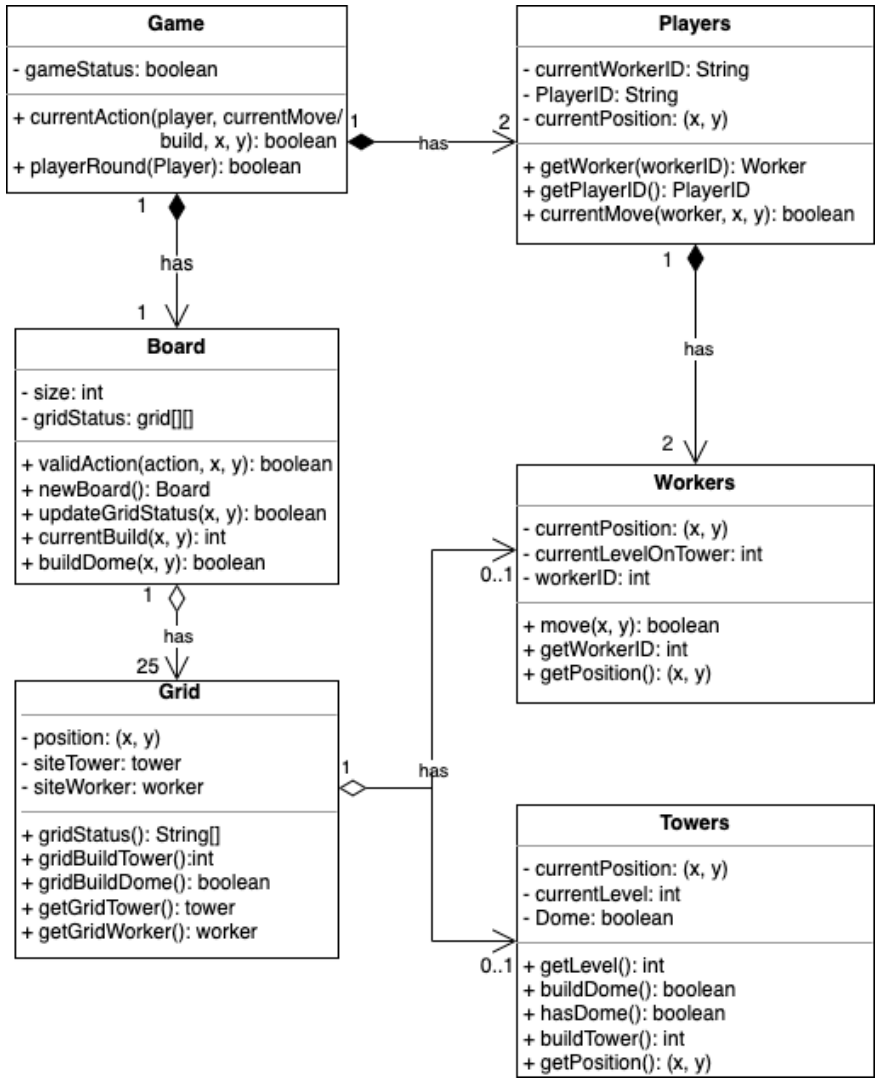


The users input attributes to currentAction functions. The action attribute can be ether build a tower, add a dome or move a worker. Different actions will trigger different boolean judgement inside this function. I will implement certain boolean inside the validAction rather than separate them into different function, since they are short. Then different parts of the game are called to modify the conditions of workers and towers.

I considered about the control to workers from both players and the grid. They Both can get workers from certain functions, which is a kind of coupling I wanted to avoid. But in one grid, the control to worker is more like save the worker's ID. So I saved this design for now.

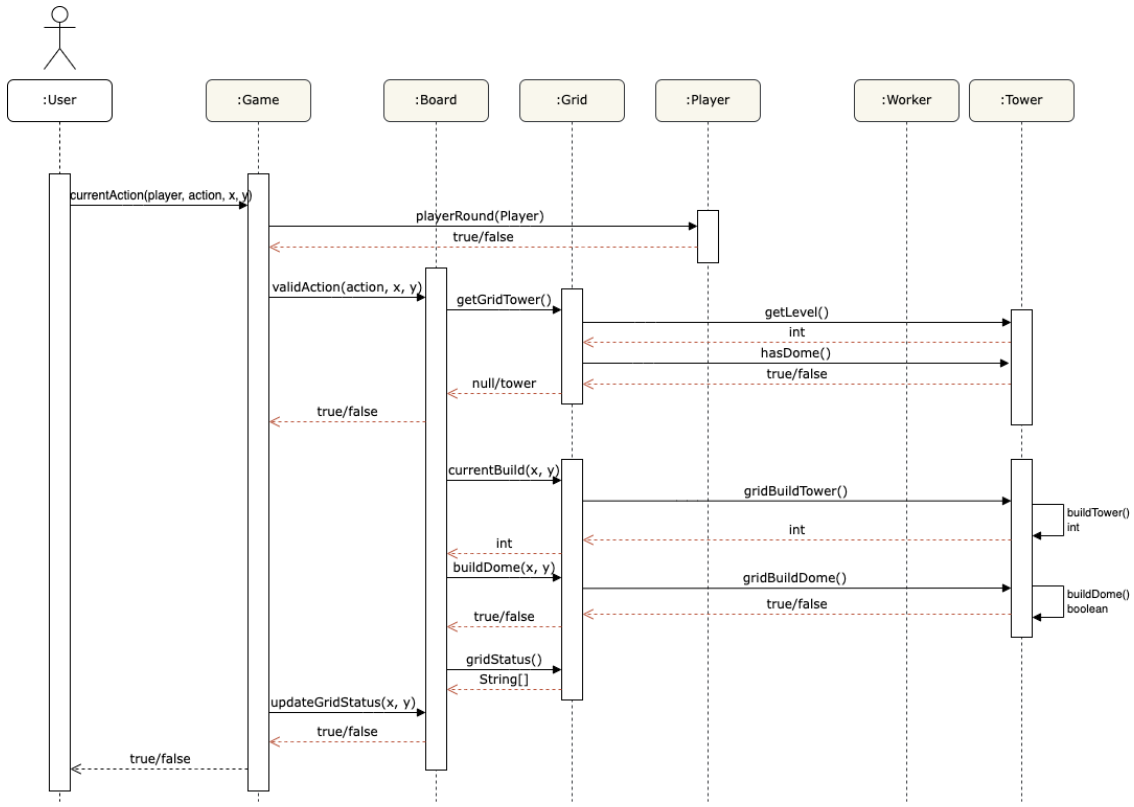
I also thought about the trade-off between cohesion on the Board class and coupling on different functions. There was one way that the Player class only store workers' ID and players' ID, but soon I found out that the class of Player is barely used, and Board would have too many cohesion with building functions. Thus, I decided to put moving workers function into the Player, since the building functions are more changing the grid status, or board's character, which is more intuitive.

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.



I think mostly states needed to be stored are the status of grids, whether there are towers or workers on them, and also the tower status of dome and levels. What's more, the game needs to store which player should play for each round. The Tower and Worker should store their own character of position.

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.



I think I will firstly check whether the build/add dome action is valid for certain grid. I used validAction(x, y) to test certain grid. If the input action is build, I expected there is no workerID or a tower with more than 2 level inside the the return string array, or else it will return false for function validAction. For adding dome, I need to check whether the returning tower has a dome or not. After the validAction return true, the program will return the next step.