Valid Build Determination

The validity of a build can be determined by function canBuildToCell (cell), which is based on the following criteria:

1. **Neighbor block Check**: A block can only be built if it is a The target cell has to be in boundaries of the board (5*5)

    b a neighbor of the worker's current position. That is, the target block is 1 cell adjacent in both x-axis and y-axis.

    c The target cell cannot be the worker's current position.

    d The height difference between the target cell and current cell can be maximum 1.

    e If there is already a worker or a dome on a target cell, then this block cannot be further built on.

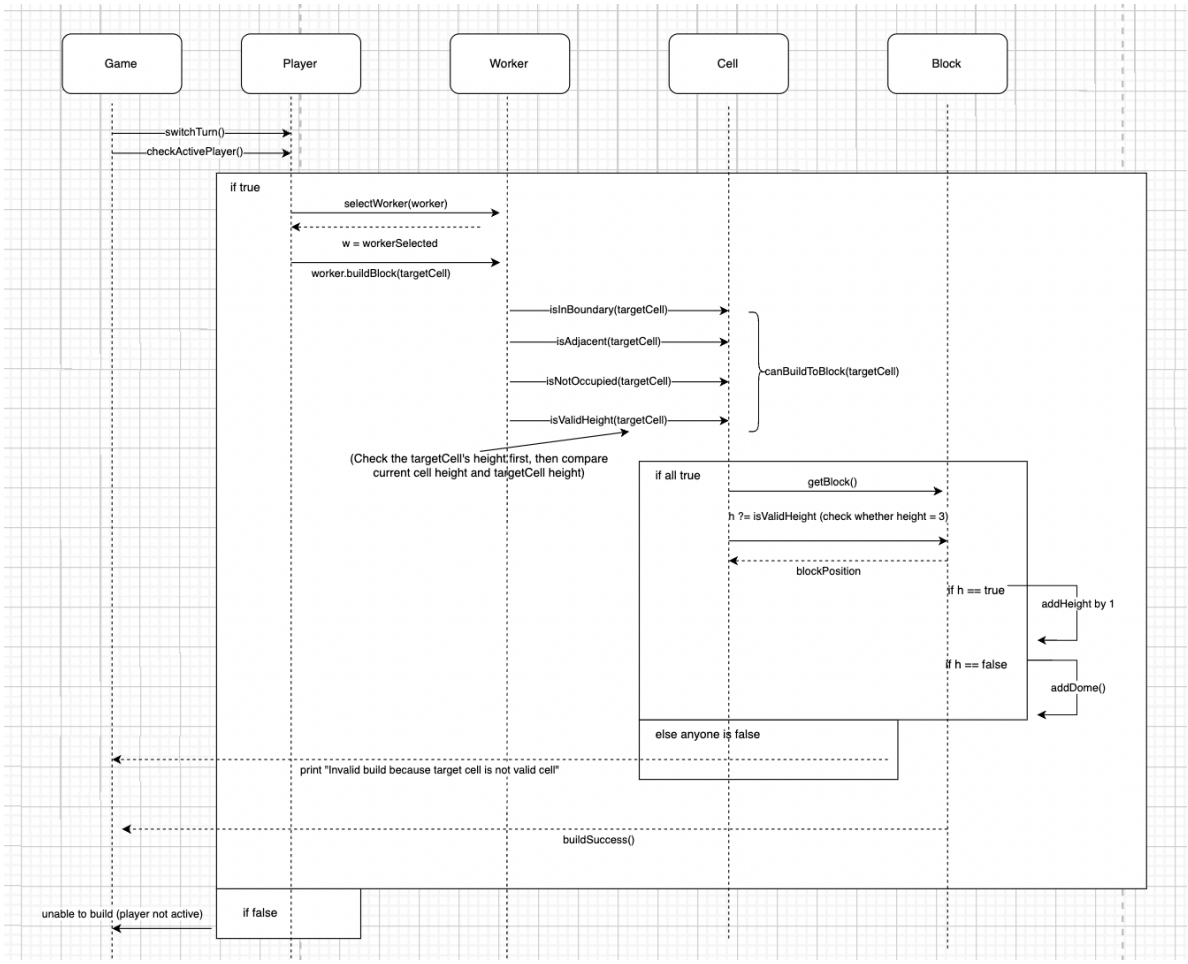
2. **Height Limit**: A normal block can only be added if the current height is less than 3.

3. **Dome Addition**: If the height= 3, a dome can be added instead of further blocks.

Perform Block Build and Update of State

The worker calls the function buildBlock (cell). After this buildBlock passes the canBuildToCell(cell), the target cell will call the function getBlock() to get the current block state on that cell. Then, the block will call the function: block.buildBlock(). If the

current block's height is lower than 3, then this block's height will be updated to increase by 1. If the current block's height is equal to 3 (i.e. MAXIMUM_HEIGHT), then the system will add a dome to that block instead of increasing height. In this case, the "hasDome" Boolean will be updated to true automatically. Afterall, this build action is finished. If the canBuildToCell(cell) is not passes, i.e. the target cell is not valid, the system will print "Invalid Build because block" and tell player to choose another block to build on.



Responsible Objects and Methods

1. **Block Class**: Responsible for managing the state of individual blocks, including height and whether a dome is present.

- buildBlock(cell Cell): Increases the height or adds a dome based on current state.
- addDome(cell Cell): Changes the state of the block to indicate a dome is present.

2. **Worker Class:** Coordinates the action of building for a specific player.
   - buildBlock(Cell targetCell): the worker performs the action to initiate the building process on a specific cell

Justification of Design Choices

1. **Encapsulation**: The Block class encapsulates all logic related to block states and modifications. This adheres to the principle of **Encapsulation**, keeping concerns separated and enhancing maintainability.
2. **Single Responsibility Principle (SRP)**: Each class has a specific responsibility:
   - The Block class manages its own state and update block height & hasDome or not.
   - The Worker class handles worker actions.
3. **Clarity and Simplicity**: The design is straightforward, allowing for easy understanding and modifications. Each method is focused on a single action, which enhances readability.

Alternatives Considered and Trade-offs

- **Validation Logic in Game Class**: One alternative was to place all validation logic within the Game class. However, this would lead to a more complex class and

make it harder to manage the building logic. Keeping the validation tied to the Block class provides clarity and reduces the complexity of the Game class.