Build Justification

The **build()** function is in the **Player** class.

The **isValidBuild()** function is used to check whether a move is valid. This function is in **Validator** class, or in a **God Card** class that has a special buildStrategy.

The whole build and check flow is as follows:

1. Determining if this player use a God Card that with special buildStrategy

- Responsibility: Player class
- Method: Build method
- Action: Check if the BuildStrategy instance is null or not
  - If BuildStrategy instance is null, use Validator.isValidBuild()
  - If BuildStrategy instance is not null, use isValidBuild() in God Card Class

2. Determining if a Build Action is Valid using Validator class (No God Card Game):

- Responsibility: Validator class
- Method: isValidBuild method
- Action: check the following things to valid a move
  - If the status of current player is “build”
  - If the grid is within the board boundaries
  - If the targeted grid’s height is less than 4
  - If the targeted grid is adjacent to the recently moved worker (which means worker can only move one grid to the new position)
  - If the targeted grid is unoccupied
  - If a worker has been moved in the current player’s turn

3. Determining if a Build Action is Valid using Demeter class (Use God Card Game):

- Responsibility: Demeter class
- Method: isValidBuild method
- Action: check the following things to valid a move
  - If the status of current player is “build”
  - If the grid is within the board boundaries
  - If the targeted grid’s height is less than 4
  - If the targeted grid is adjacent to the recently moved worker (which means worker can only move one grid to the new position)
  - If the targeted grid is unoccupied
  - If a worker has been moved in the current player’s turn
  - If the player hasn't already done a build in this turn, or, If the player try to build the second time in this turn, and the second targeted grid is in a different place from the first targeted grid.

4. Build a block on a certain grid:

- Responsibility: Board class
- Method: buildBlock method

- Action: Increase the height of the targeted grid by one

Justification:

Deciding to separate the default validation logic into the Validator class is based on the Single Responsibility Principle. This class solely focuses on validating actions, decoupling the validation logic from game states or player actions, enhancing the system's maintainability and extensibility while improving cohesion.

For instance, using the BuildStrategy interface and the Demeter class, the application of the strategy pattern aligns with essential design principles of flexibility, modularity, and maintainability. This method encapsulates the construction logic in distinct strategy classes, facilitating the addition or modification of building rules for various deities, like Demeter, without altering the core gameplay. This design choice reflects the principles of open/closed and single responsibility by concentrating each strategy class on a specific aspect of gameplay (building), allowing the introduction of new behaviors without changing existing code.

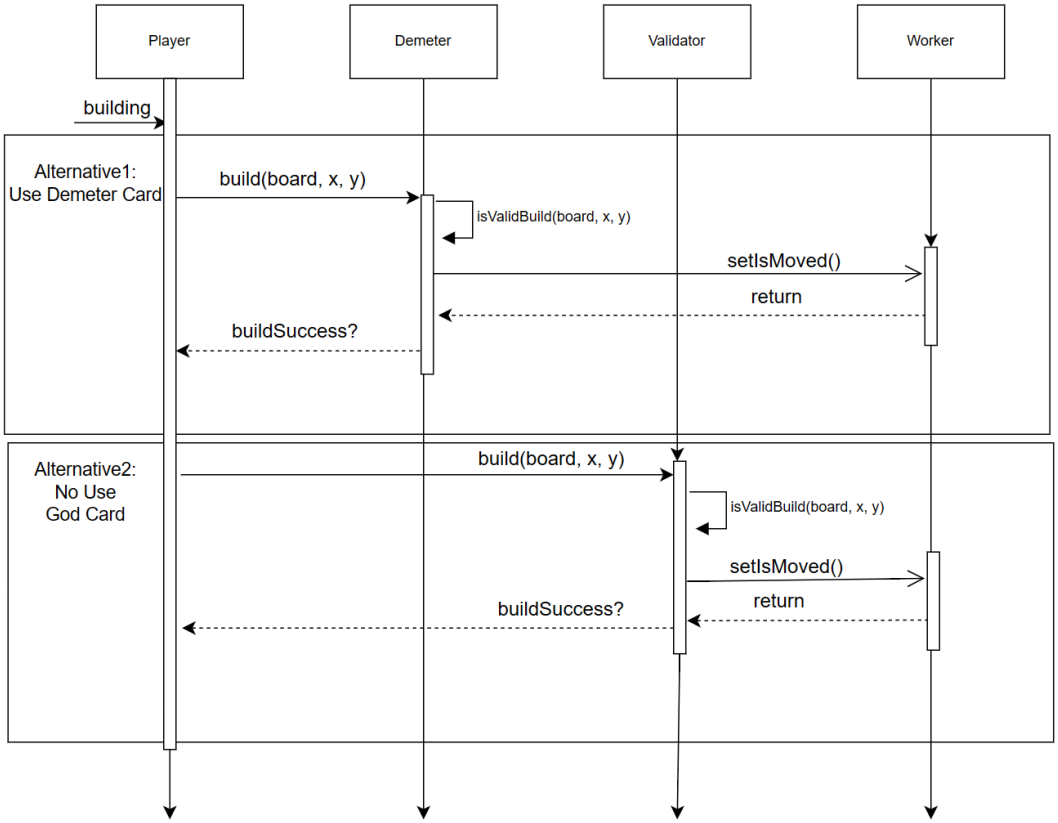
The Player class represents the actions a player can undertake, and by encapsulating the building logic within the build method, it adheres to the Encapsulation Principle, ensuring all related functionalities are contained within the class.

The Board class acts as the central authority for the game state. Assigning it the responsibility for direct state modifications (like building a block) ensures data integrity and cohesion. The internal state (grid) remains private, guaranteeing that changes to the state occur through controlled methods.

Alternatives:

Embedding validation logic within the Player or Board classes could simplify interactions but would violate the Single Responsibility Principle, resulting in larger, more complex classes. Additionally, the Player or Board classes would need to understand the specific abilities of each god card in order to correspond to different validation logics, making such a design difficult to maintain and extend.

Sequences:


Player
Demeter
Validator
Worker
building
Alternative1:
Use Demeter Card
build(board, x, y)
isValidBuild(board, x, y)
setIsMoved()
return
buildSuccess?
Alternative2:
No Use
God Card
build(board, x, y)
isValidBuild(board, x, y)
setIsMoved()
return
buildSuccess?