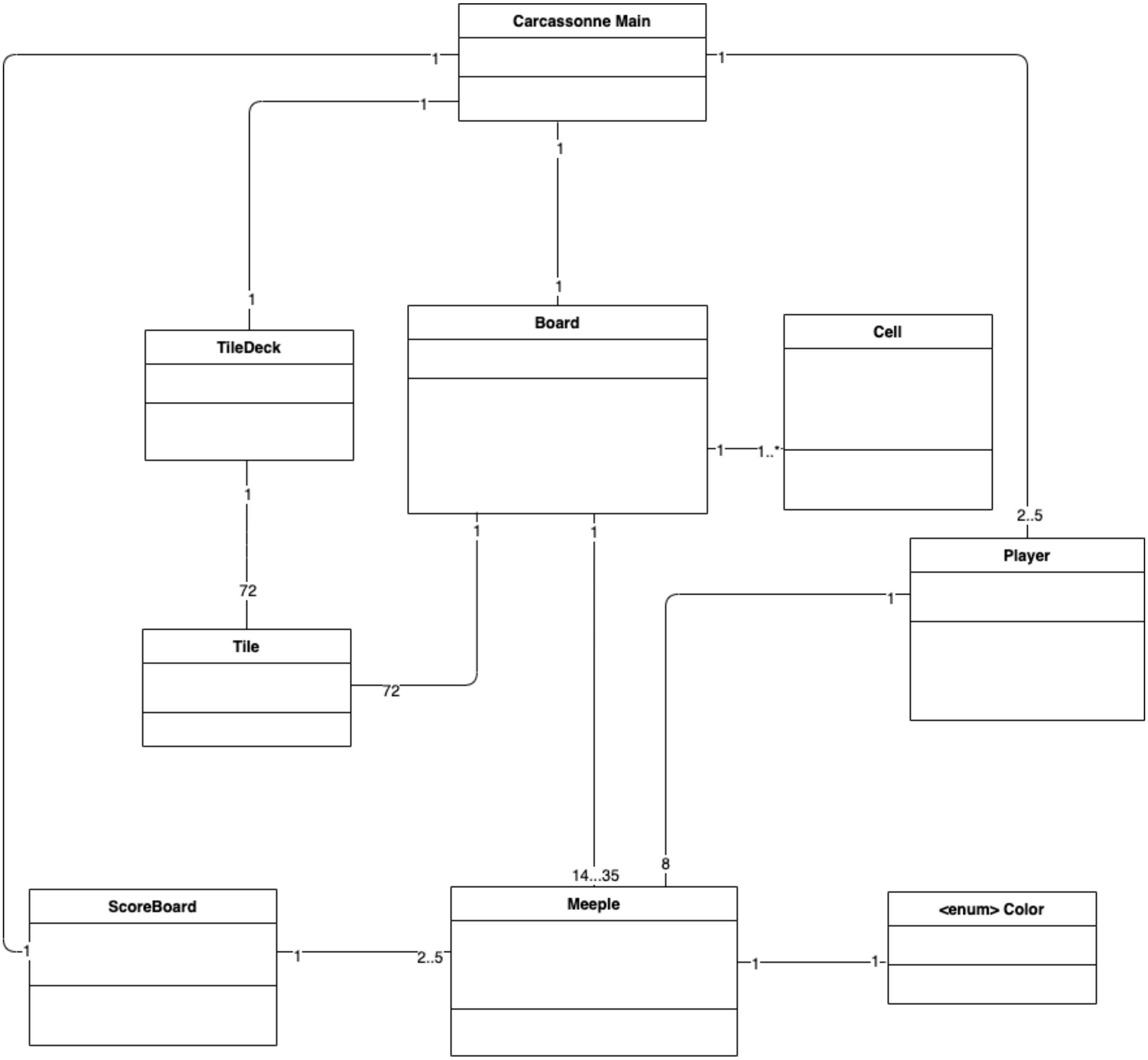
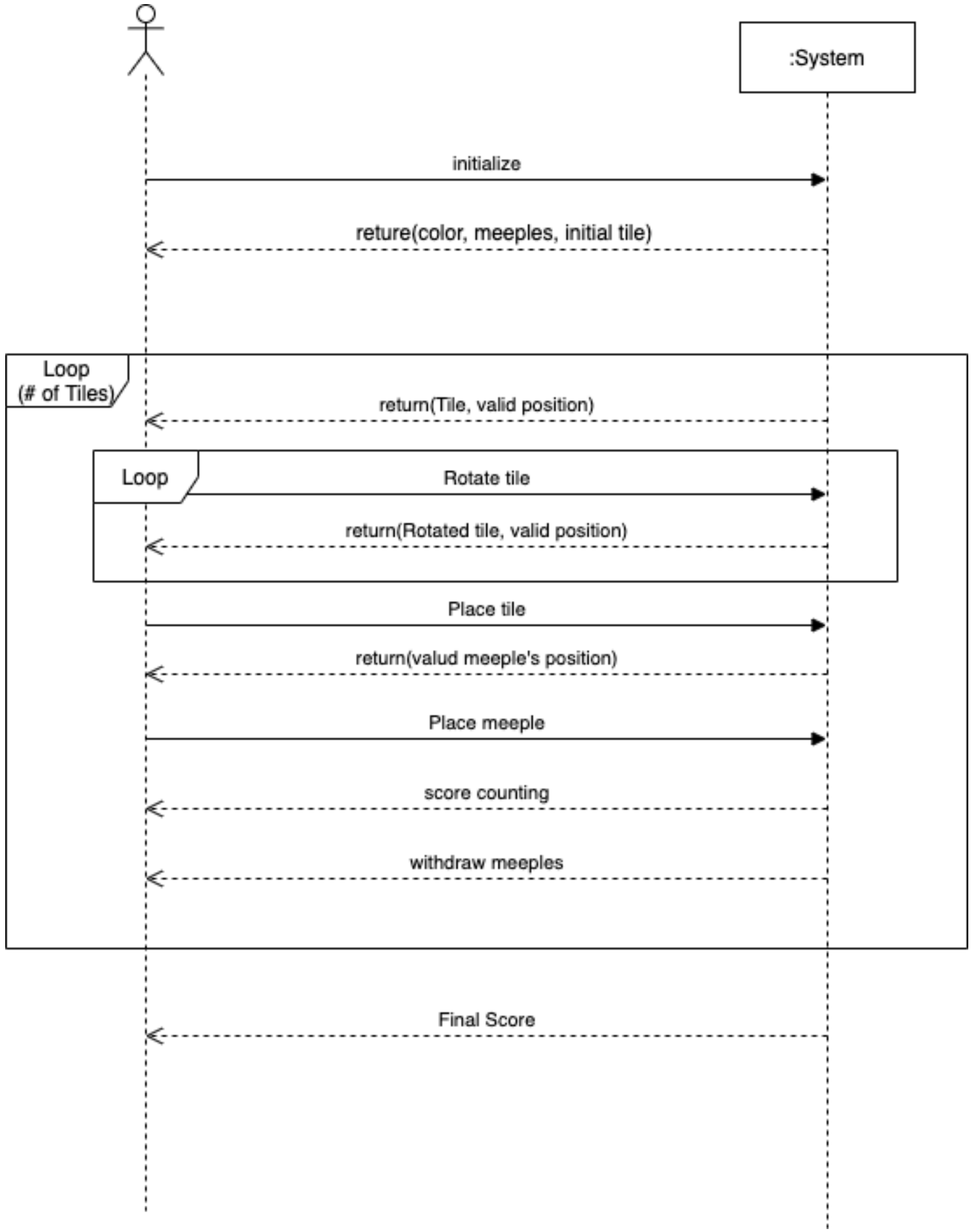
Domain Model



System Sequence


:System
initialize
reture(color, meeples, initial tile)
Loop
(# of Tiles)
return(Tile, valid position)
Loop
Rotate tile
return(Rotated tile, valid position)
Place tile
return(valud meeple's position)
Place meeple
score counting
withdraw meeples
Final Score

Behavioral contract

The user can play a tile but cannot place a meeple.

Operation: placeTile(List<int[]>)

Cross References: Use cases: it happens in each round for each player

Preconditions:

- There is at least one tile in the tile deck. The tile isn't a start tile.
- There is at least one valid place to put the tile

Postconditions:

- Return a valid coordinate
- A new tile is added to the board
- The tile deck pops the deck out

Operation: placeMeeple(Tile)

Cross References: Use cases: it happens after players place a tile.

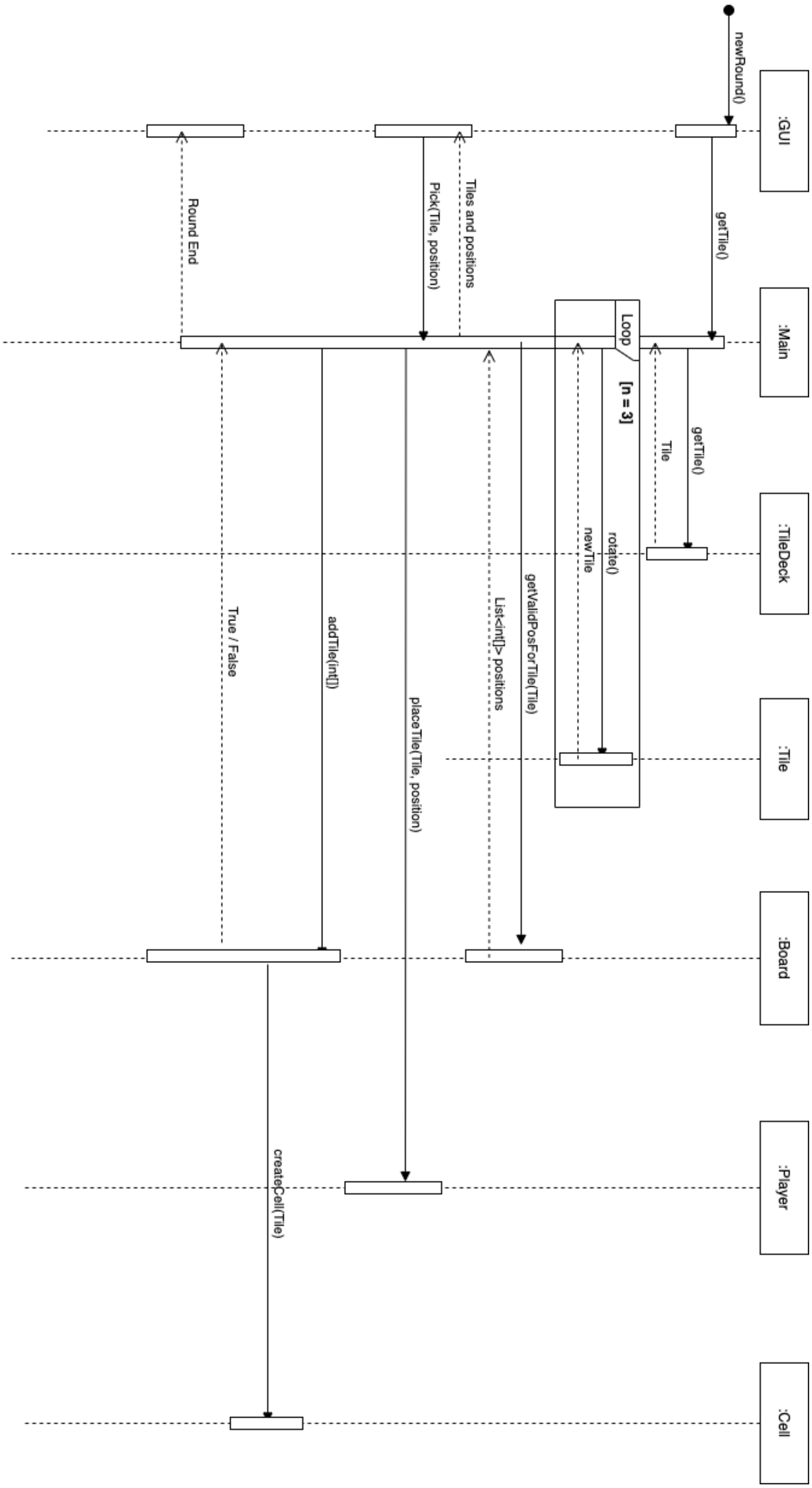
Preconditions:

- The player just placed a tile in the board
- There is at least one valid place in the tile. The valid place means it has to be a city, a road or a monastery that haven't been occupied by other players' meeples.
- The player owns at least one meeple

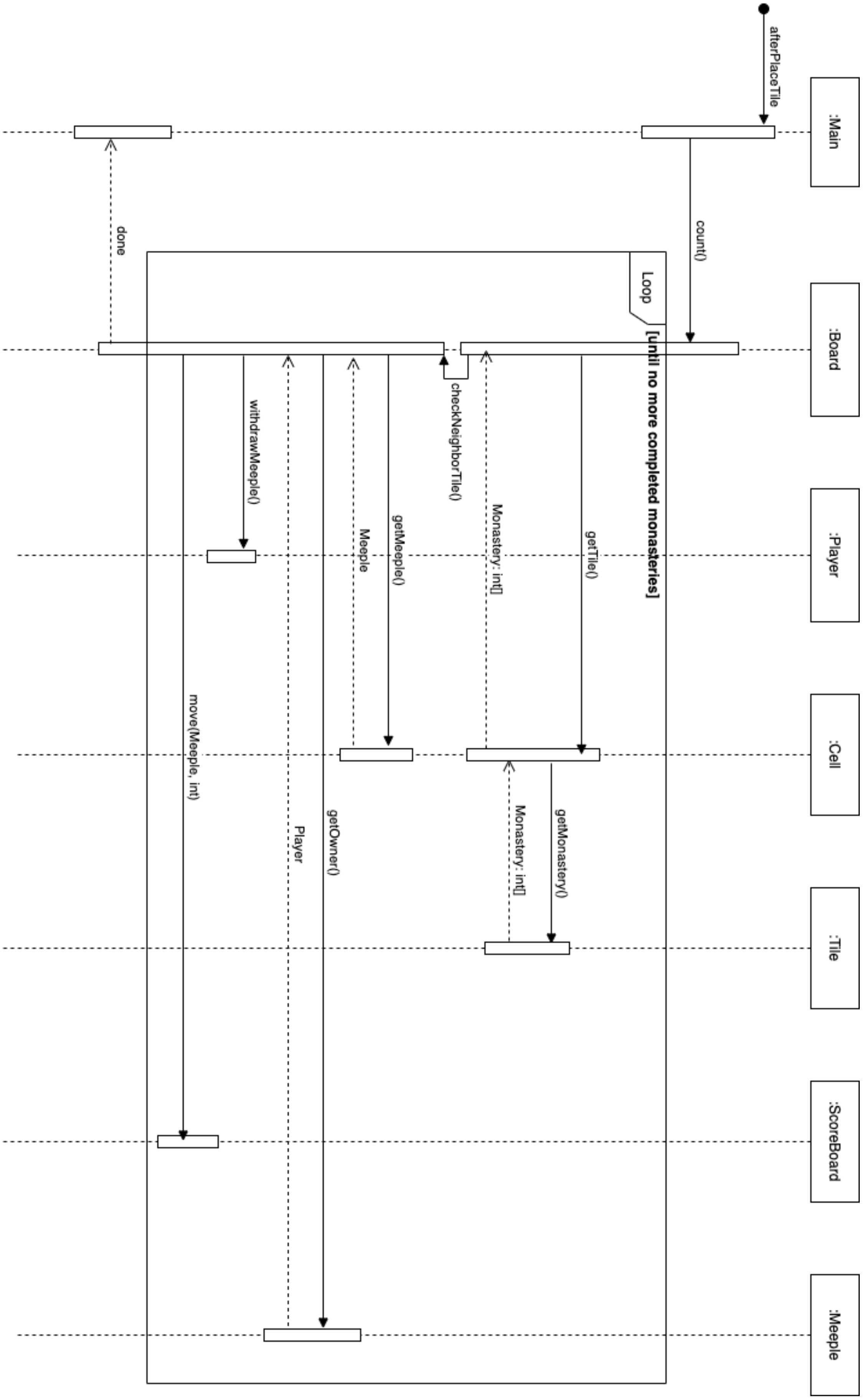
Postconditions:

- A new meeple is placed in the board
- Any meeples that are in a completed road, city or monastery should be counted and withdraw from the board
- Scoreboard should be updated

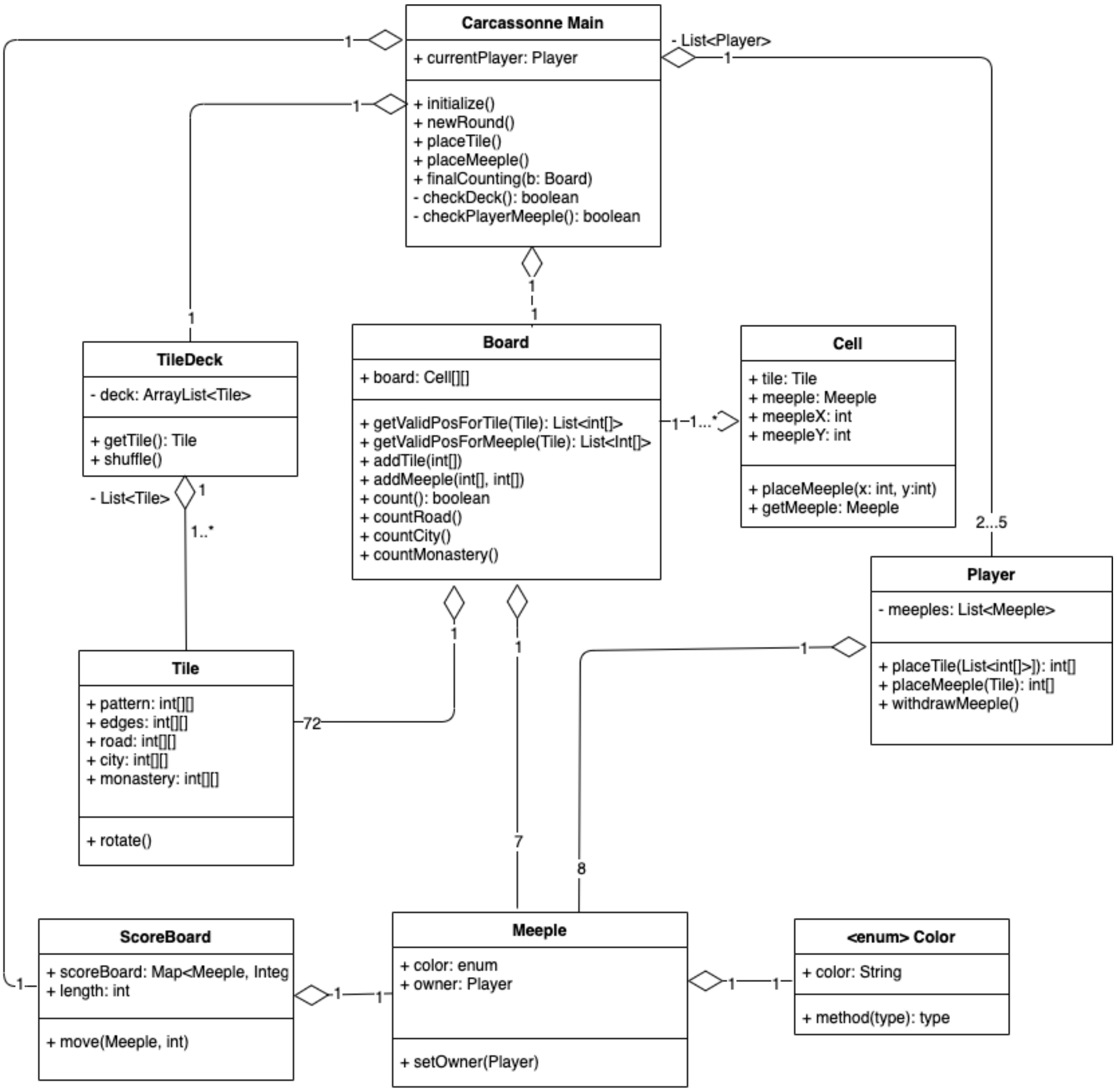
Tile Validation Interaction



Monastery Scoring Interaction



Object-level Interaction:



Rationale:

- How can a player interact with the game? What are the possible actions a player can perform?

A player interacts with the game through the buttons on the GUI. The player can perform four actions: start a game, rotate a tile, place a tile and place a meeple.

- How would someone start a game with multiple players? How would players share the same screen?

At the beginning of the game, players are asked to enter the number of players between 2 to 7. In each round, the players would take turns to play a tile. The screen would show the color of the current player so that players can decide who is going to play the current tile.

- How is a player's action of placing a tile with a meeple represented differently from placing a tile with no meeples?

In my solution, there is no difference when a player is placing a tile. After the player places a tile, the system would check automatically if the player owns at least one meeple. If the player does, the system would allow the player to place a meeple in the tile. If the player doesn't, the system just skips the meeple part and starts to count the scores.

- How are the current placements of tiles represented? How are tile rotations represented?

There is a class called Board. Board has a 2D array that represents the placements of tiles and meeples. Tile class has a function called rotate. Every time the function is called the pattern of the tile would be rotated and all the instance fields would be changed in the clockwise direction.

- When a new tile is placed, how does the game determine the newly completed features to be scored? How does the game determine if a meeple placement conflicts with a previously played meeple?

The Board would check the features on this tile and its neighbors. For example, if the tile has a road but its neighbors don't, the road cannot be a completed feature. Then the Board would check the specific coordinate of the possibly completed features to count the scores. The Board knows the position of each meeple on the board so it could inspect if the feature is occupied by a meeple by going through all the neighbors of the tile and the possible neighbors of the neighbors.

- How does the game support different scoring methods for different features? For example, monasteries are completed when they are surrounded by 8 tiles, but cities are completed when there is no tile placement that could expand the city.

There are three ways of counting in Board class. Each of them would implement a scoring method for one feature.

- How are invalid operations handled? To what extent are they processed internally vs. exposed to clients using the game? Are invalid operations exposed as exceptions, boolean or null return values, or other structures that represent the error?

In my solution, there would not be invalid operations because the players are allowed to pick the placement of tiles and meeples given by the system. The system would check if there is any valid position and return players the results. If there is an error, for example, the system will return a Boolean value to show that the operation fails.

- How will the game be represented in a GUI? What information does a GUI need to access from the game?

The GUI needs the patterns of tiles, the colors of players, the board information and the scoreboard.