Explain two key classes that you have debated as a team

We have implemented two important classes, that is, BoardModelImpl and BoardImpl. Their interfaces are BoardModel Interface and Board Interface respectively.

Through these two classes, we have completed the separation of concerns for "Board". In the traditional vision, "Board" needs to save data and interact with the outside world. But in the current design, we classify all the data storage functions of the chessboard in BoardModelImpl. At the same time, BoardModelImpl will provide meta-operations for manipulating data. For example, adding, deleting, modifying and checking operations. The implementation of complex operations and logic will be implemented by BoardImpl.

BoardModelImpl only provides the most basic meta-operations, such as querying a Position, or querying a Token on a Position. Or place a Token in a Position. These operations are basic and involve no logic.

BoardImpl itself does not have data, it just provides an interface to operate BoardModelImpl. For example, when a player wants to place a Token at a certain Position on the board, BoardImpl will perform the following three steps: Step 1, check whether the coordinates entered by the player are legal, for example, the player is not allowed to place a Token on a line or place the Token Placed on a position off the board. The second step is to check that the Position is empty, and the player is not allowed to place Token again on the Position that already has Token. In the third step, BoardImpl calls the method of BoardModelImpl to complete the placement of Token. These three steps must succeed or fail at the same time, so BoardImpl will combine these three operations into a function.

Similarly, BoardImpl can combine the two meta-operations provided by two BoardModelImpls, delete the Token of a certain Position and add a Token to another Position to complete the movement of the chess pieces.

At the same time, BoardModelImpl abstracts data storage. It does not need to store data according to the true model of the chessboard. For example, the 9MM chessboard is a 2D chessboard. If we use a two-dimensional array to store Position data, it will cause a lot of data waste, and we need to plan its coordinates in two-dimensional space. Since we use BoardModelImpl for data abstraction, we will use a one-dimensional array to store all Positions.

BoardModelImpl does not care how the Positions stored in the one-dimensional array are rendered in the View, because the rendering function will be managed by the BoardView.

Through the separation of concerns, the operation of data and the storage of data will be completely separated. This effectively prevents "Board" from becoming a God class, and at the same time makes each class have a clearer division of responsibilities. At the same time, because the Board itself has too much complex data and operations, we divide the Board into several different classes instead of several functions.