Class Diagram

HuaRongDaoGame

-levels: dictionary
-board: list
- history: list
-steps:int
-startTime:int
-currentLevel:str
-highScores: str
-pieceSizes: dictionary
- - - - - - - -
-setBoard(boardStr: list): Bool
-getBoardStr(): str
-movePiece(piece: Tile, dir: enum):tuple
-undo(): tuple
-is_solved(): Bool
-getState(): dictionary
-geneRandBoard(): str
-generateHint(): void

UI System

-painter: BoarPainter
-handler: ButtonHandler
- - - - - - - -
-update_UI(): void

BoardPainter

-board: list
-selected_id:int
-piece_colors: dictionary
-rect_map:dictionary
-piece_pixmaps:dictionary
-hint_info: tuple
-hintTimer: QTimer()
- - - - - - - -
-initboard(): void
-showHint(pid: char, dir:enum):void
-get_piece_span(pid): int
-clearHint():void
-mousePressEvent(event): void

ButtonHandler

-step: int
-elapsed: int
-currentLevel: str
-selectedPiece: str
-zmq: ZmqThread()
-infoLabel: Qlabel()
-timerLabel: Qlabel()
-stepLabel: Qlabel()
-recordLabel:Qlabel
-upButton: Button
-downButton: Button
-leftButton: Button
-rightButton: Button
-undoButton: Button
-resetButton: Button
-levelButton: Button
-hingButton: Button
-levelMenu: Menu
- - - - - - - -
-show_level_menu():void
-_onLevelSelected(level_name: str):void
-onPieceSelected(pid:str):void
-send_move(direction: str): void
-setLevel(levelName: str): void
-setRandomLevel(): void
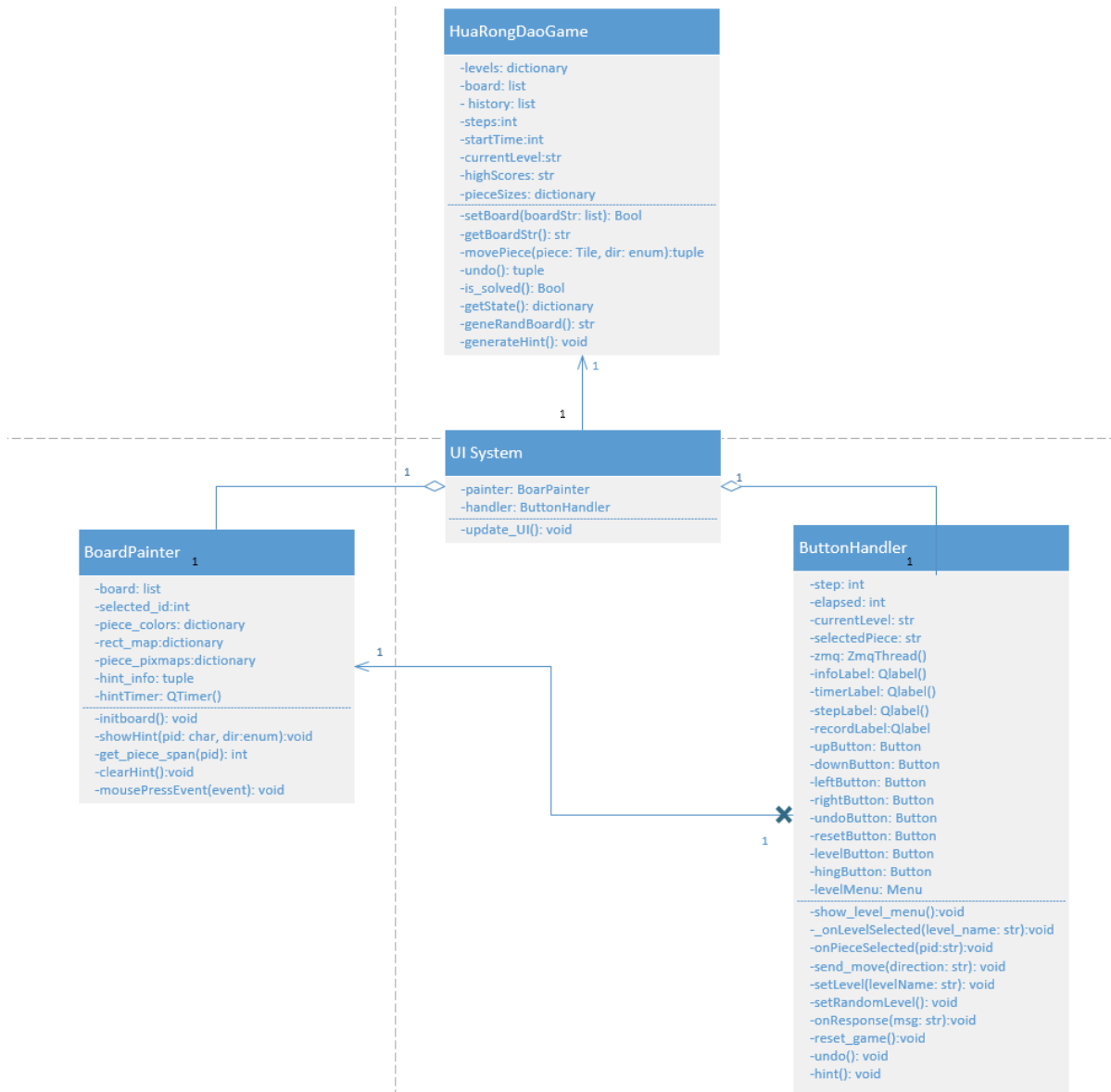-onResponse(msg: str):void
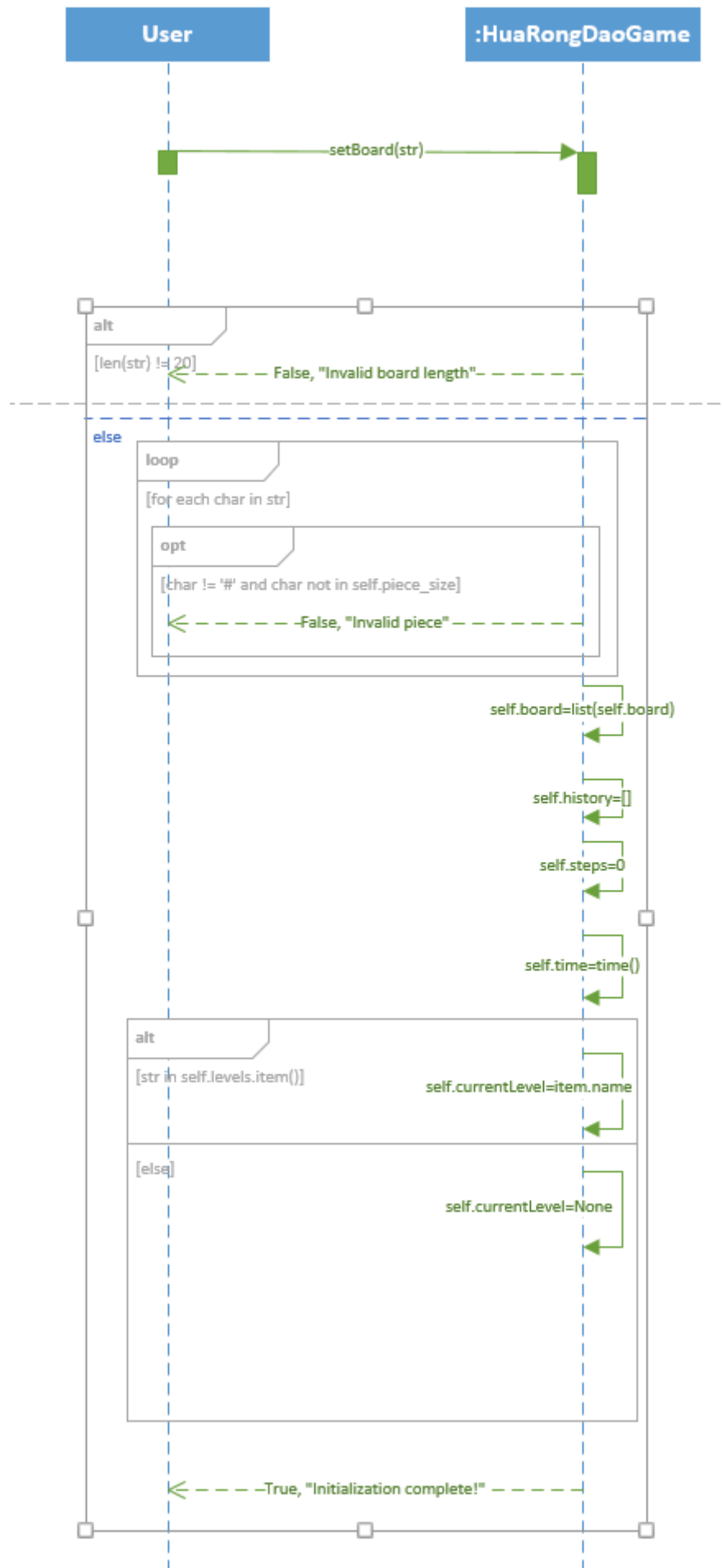-reset_game():void
-undo(): void
-hint(): void

Method descriptions

Class Method:

1. Class HuaRongDaoGame

1.1 setBoard(BoadrStr: list): Board

- input the board list(list that use numbers to represent a piece)
    - check if the board if valid, if valid, set current board to BoardStr; if not valid, return false

User → :HuaRongDaoGame

setBoard(str)

alt

[len(str) != 20]
False, "Invalid board length"

else

  loop
  [for each char in str]

    opt
    [char != '#' and char not in self.piece_size]
    False, "Invalid piece"

self.board=list(self.board)

self.history=[]

self.steps=0

self.time=time()

alt

[str in self.levels.item()]
self.currentLevel=item.name

[else]
self.currentLevel=None
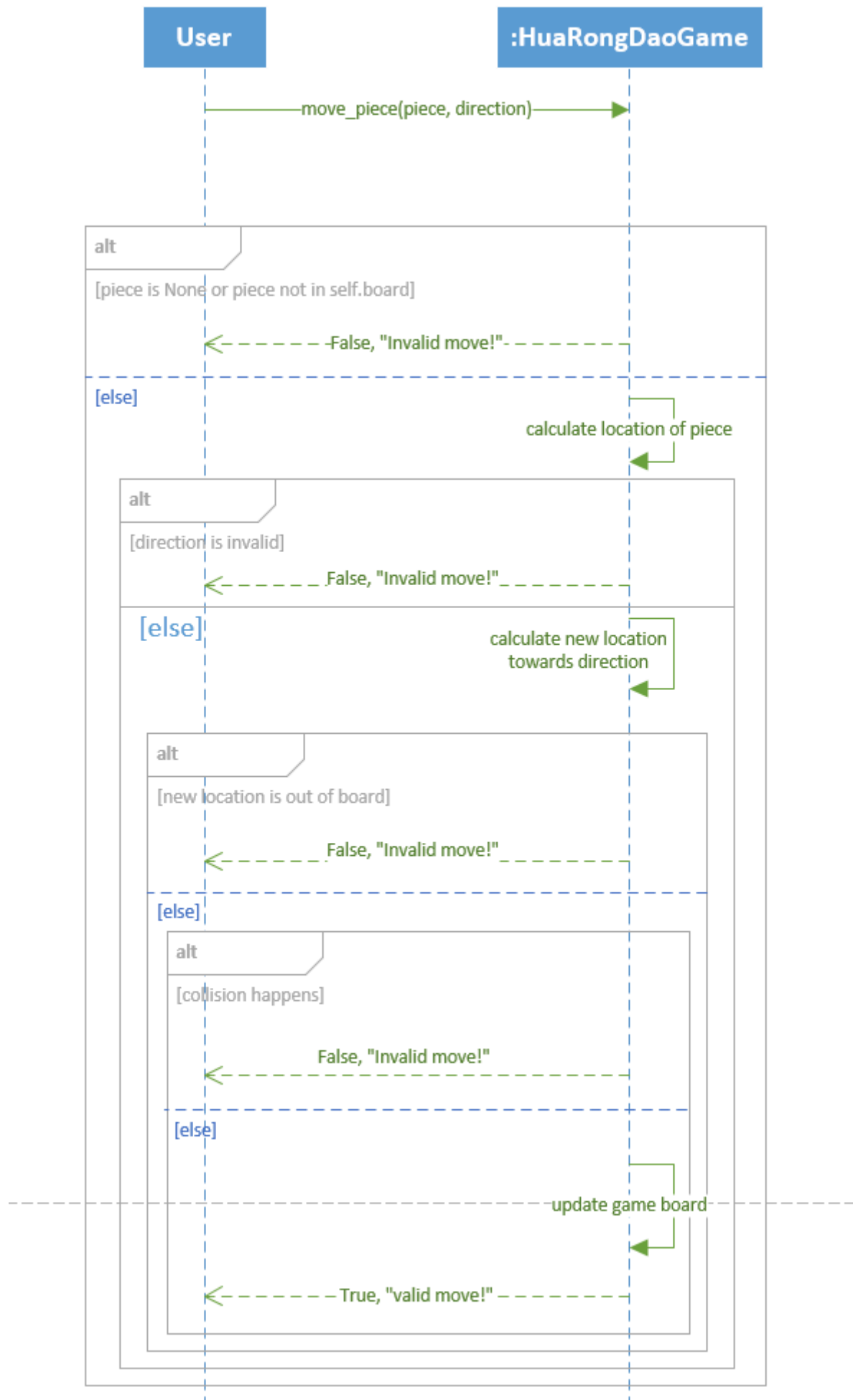
True, "Initialization complete!"

1.2 getBoard(): str

- return current board's board list.

1.3 movePiece(piece: Tile, dir:enum): tuple

- move the piece to the direction dir; if valid move, update board; if not valid, not update the board and return false

```
                  ┌──────────┐                        ┌──────────────────┐
                  │   User   │                        │  :HuaRongDaoGame │
                  └──────────┘                        └──────────────────┘
                       │         move_piece(piece, direction)         │
                       │─────────────────────────────────────────────▶│

  ┌─alt──────────────────────────────────────────────────────────────────────────┐
  │ [piece is None or piece not in self.board]                                     │
  │                    │◀────────False, "Invalid move!"────────────────│           │
  ├────────────────────────────────────────────────────────────────────────────────┤
  │ [else]                                                             │            │
  │                                               calculate location of piece       │
  │                                                                    ┌─┐          │
  │                                                                    └─┘◀         │
  │    ┌─alt──────────────────────────────────────────────────────────────┐        │
  │    │ [direction is invalid]                                            │        │
  │    │               │◀────────False, "Invalid move!"──────────│         │        │
  │    ├────────────────────────────────────────────────────────────────────┤       │
  │    │ [else]                                     calculate new location   │       │
  │    │                                              towards direction      │       │
  │    │                                                        ┌─┐          │       │
  │    │                                                        └─┘◀         │       │
  │    │   ┌─alt────────────────────────────────────────────────────┐       │       │
  │    │   │ [new location is out of board]                          │       │       │
  │    │   │           │◀──────False, "Invalid move!"──────│          │       │       │
  │    │   ├──────────────────────────────────────────────────────────┤      │       │
  │    │   │ [else]                                                    │      │       │
  │    │   │   ┌─alt──────────────────────────────────────────┐        │      │       │
  │    │   │   │ [collision happens]                           │        │      │       │
  │    │   │   │         │◀────False, "Invalid move!"────│       │        │      │       │
  │    │   │   ├────────────────────────────────────────────────┤       │      │       │
  │    │   │   │ [else]                                          │       │      │       │
  │    │   │   │                            update game board    │       │      │       │
  │    │   │   │                                          ┌─┐    │       │      │       │
  │    │   │   │                                          └─┘◀   │       │      │       │
  │    │   │   │     │◀──────True, "valid move!"────────│        │       │      │       │
  │    │   │   └────────────────────────────────────────────────┘       │      │       │
  │    │   └──────────────────────────────────────────────────────────────┘      │       │
  │    └────────────────────────────────────────────────────────────────────────┘       │
  └──────────────────────────────────────────────────────────────────────────────────┘
```

1.4 undo(): tuple

- turn the board to last state and update the log. Return "True, Undo complete"
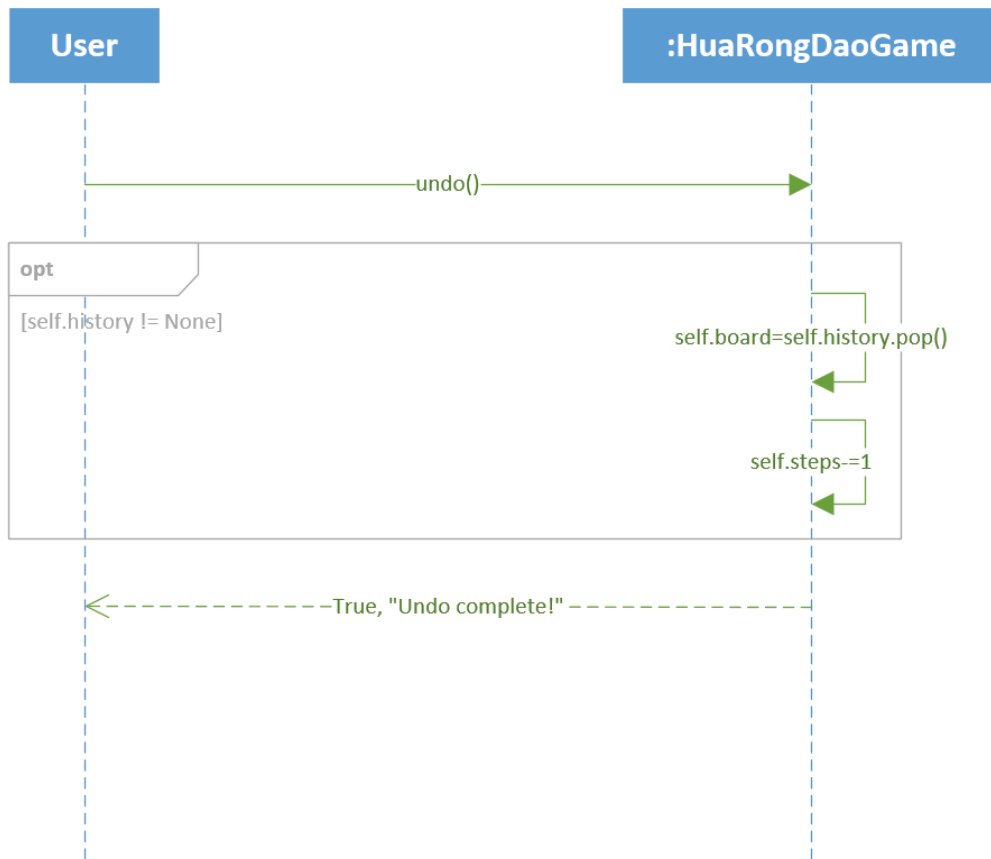
1.5 is_solved(): Bool

- check if the current level has been solved. True for solved, False for not

1.6 getState(): dictionary

- return current time and time used in a dictionary.

1.7 geneRandBoard(): str

- if the random board is not valid, return None

  if the random board is valid, turn current board to a level that is generated randomly.

1.8 generateHint(): void

  create subprocess to find the next step using UPPAAL, and send the result back to frontend.

2 Class BoardPainter

2.1 initboard(): void

- initial the board information.

2.2 showHint(pid: char, dir:enum): void

- show the hint UI according to the pid of the piece and dirction
  - draw an arrow, whose direction is defined by dirction, in the piece

2.3 get_piece_span(pid: char): int

- get the span of the piece pid.

2.4 clearHint():void

- clear the hint UI.

2.5 mousePressEvent(event): void

- respond User clicking and change the selected_id to the selected one

2.6 paintEvent(): void

- to paint the board(pieces location, selected piece color filling) according to current board information.

3 Class ButtonHandler

3.1 show_level_menu(): void

- show the level menu, used when the user click the level selection button.

3.2 _onLevelSelected(level_name:str):void

- change the board to the level_name that is clicked by user.

3.3 onPieceSelected(pid: str):void

- select the piece pid

3.4 send_move(direction: str):void

- send move signal to Server and to HuaRongDaoGame later.

3.5 setLevel(levelName: str):void

- set current level to the levelName

3.6 setRandomLevel(): void

- set current level randomly.

3.7 reset_game():void

- reset the game to the initial state according to the level chosen last time.

3.8 undo(): void

- send undo massage to backend

3.9 hint(): void

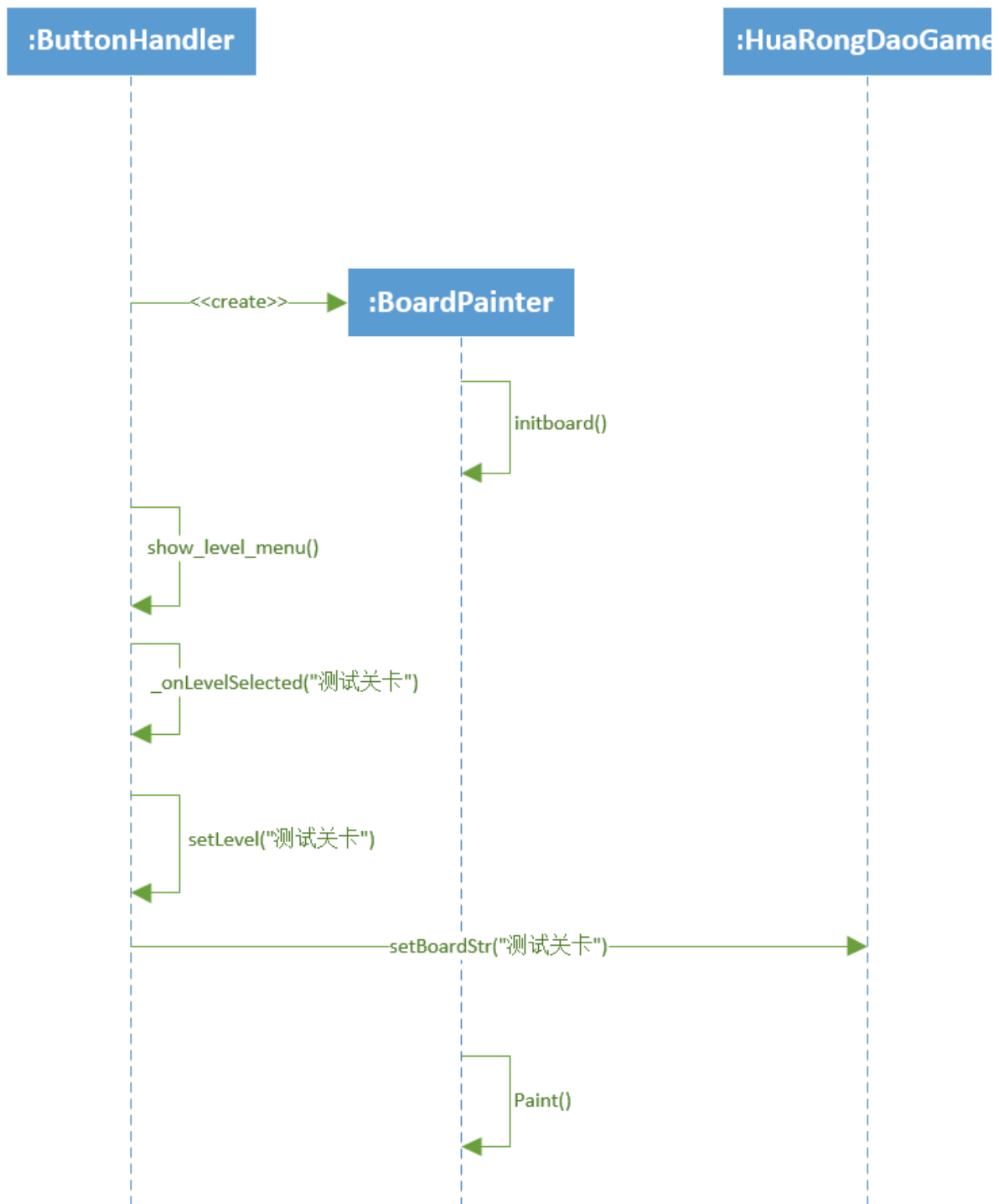- send hint request to backend

4. Class UI system

4.1 update_UI(): void

Update all the labels and painting on the response of backend.

The sequence Diagram:

1. initialize the game and choose the level "Testing level"

2. click a time and move it; Undo; click Hint and move the piece according to the hint.

```
         :ButtonHandler          :BoardPainter          :HuaRongDaoGame

              ┌─┐
    onPieceSelected(9)
              └→│
              │──────mousePressEvent()──────→│
              │                              ┌─┐
              │                          paintEvent()
              │                              └→│
              │←──────────paintDone──────────│
         ┌─┐
    send_move(Left)
         └→│
              │──────────movePiece(9, Left)──────────────────────→│
              │←──────────────moveDone───────────────────────────│
              │──────────────is_solved()─────────────────────────→│
                              │←────────is_solved=is_solved()─────│
                              ┌─┐
                          paintEvent()
                              └→│
              │←─────paintDone─────│
              │──────────────────Undo()──────────────────────────→│
                              │←────────────UndoNone──────────────│
                              ┌─┐
                          paintEvent()
                              └→│
              │←──────paintEventDone──────│
              │──────showHint(1, Down)──────→│
                              ┌─┐
                          paintEvent()
                              └→│
              │←──────paintEventDone──────│
              │──────────clearHint()────────→│
                              ┌─┐
                          paintEvent()
                              └→│
              │←──────paintEventDone──────│
              │──────────────────────────────→│
    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         ┌─┐
    onPieceSelected(9)
         └→│
              │──────mousePressEvent()──────→│
                              ┌─┐
                          paintEvent()
                              └→│
              │←──────────paintDone──────────│
         ┌─┐
    send_move(Left)
         └→│
              │──────────movePiece(9, Left)──────────────────────→│
              │←──────────────moveDone───────────────────────────│
              │──────────────is_solved()─────────────────────────→│
              │←────────────is_solved=is_solved()─────────────────│
         ┌─┐
    onResponse(is_solved)
         └→│
```