Mini-Project report

Design and implementation of the traditional game O an quan

Ha Viet Hung – 20194435

Nguyen Quang Minh - 20194444

1. Team members and assignments

* The Board class :
  + moveclockwise/counterclockwise : Ha Viet Hung
  + takeStone, checkWinning : Nguyen Quang Minh 50%, Ha Viet Hung 50%
  + returnStone, printBoard(): Ha Viet Hung
* The Scene package : Nguyen Quang Minh
* The cell package : Ha Viet Hung 50%, Nguyen Quang Minh 50%
* Player class and stone class: Nguyen Quang Minh

1. Mini project description:

* Make an application for 2 users to play the traditional game O an quan.
* On the main screen:
* Start: start the game.
* Exit: exit the program, ask the user if they really want to exit
* Help: Show guide for playing the game
* In the game:
* Gameboard: The gameboard consists of 12 cells with 10 squares, divided into 2 rows, and 2 half-circle on the 2 ends of the board. Initially, each square has 5 small gems, and each half-circle has 1 big gem. Each small gem equals 1 point, and each big gem equals 5 points.
* For each turn, the application must show clearly whose turn it is. That player will select a square on their side of the field and a direction (clockwise or counterclockwise) to spread the gems.
  + The software will then take every gems in the selected square, then put one of them in each square/half-circles in the chosen direction
  + After spreading the last gem, depending on the following cell there can be various events :
    - If the next cell contains at least one gem, then we use the gems in that square to continue spreading in the same direction
    - If the next cell is empty followed by a non-empty one, then the player gets every gems in the non-empty cells as points. This process is indefinitely repeated if the following two cells are still an empty cell and then a non-empty one
    - If the next cell is a half-circle or two consecutive empty cells, then the player’s turn ends
* If during their turn, a player doesn’t have any gem in all of their 5 squares then 5 small gems are taken from their scores and put one in each
* The game ends when there is no gem in both half-circles, both players gets every gems on their respective sides of the fields. The application must notify who is the winner and the score of each player.

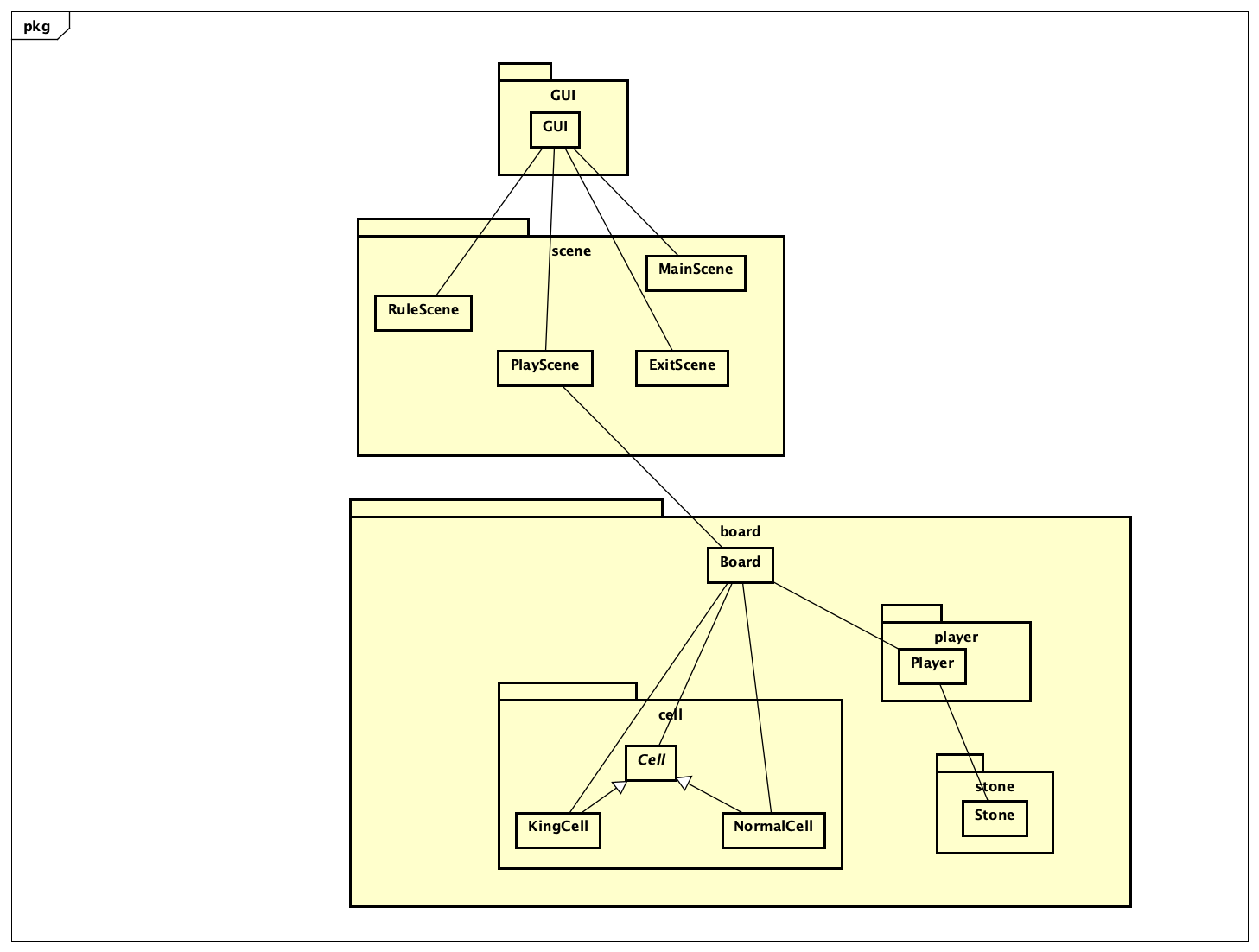
1. User case diagram and application:

* The application’s main purpose is for two people playing against each other, so our primary actor is Player
* Start the game : in the main screen , click the play button on the screen to play the game
* View guide: in the in the main screen, click the rule button on the screen to view the rule
* Exit the game: in the main screen, click the exit button on the screen to exit the application
* Spread the gems: in the play screen, right click the cell you want to spread the gems, a context menu will show up, choose the direction

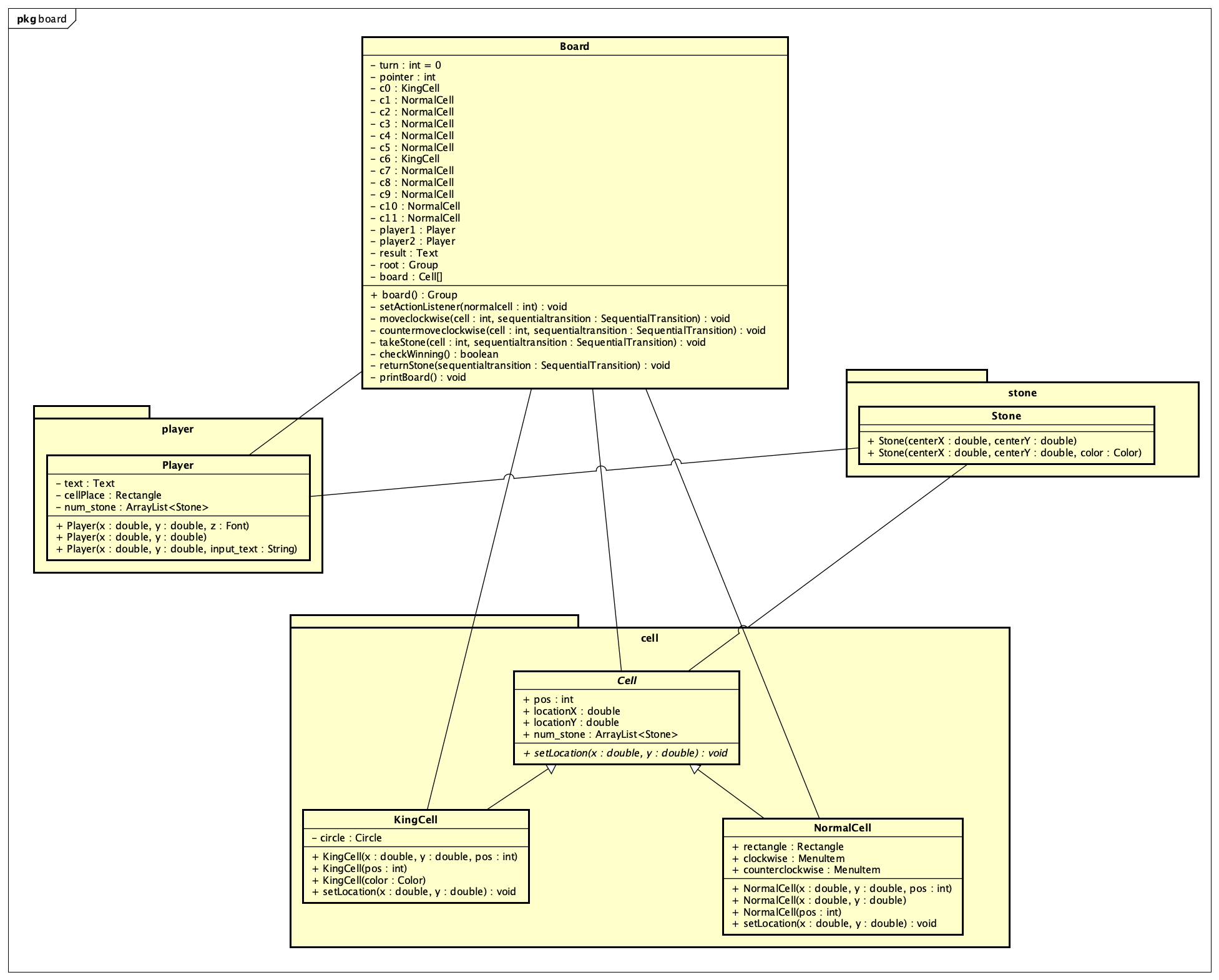
Diagram

Description automatically generated

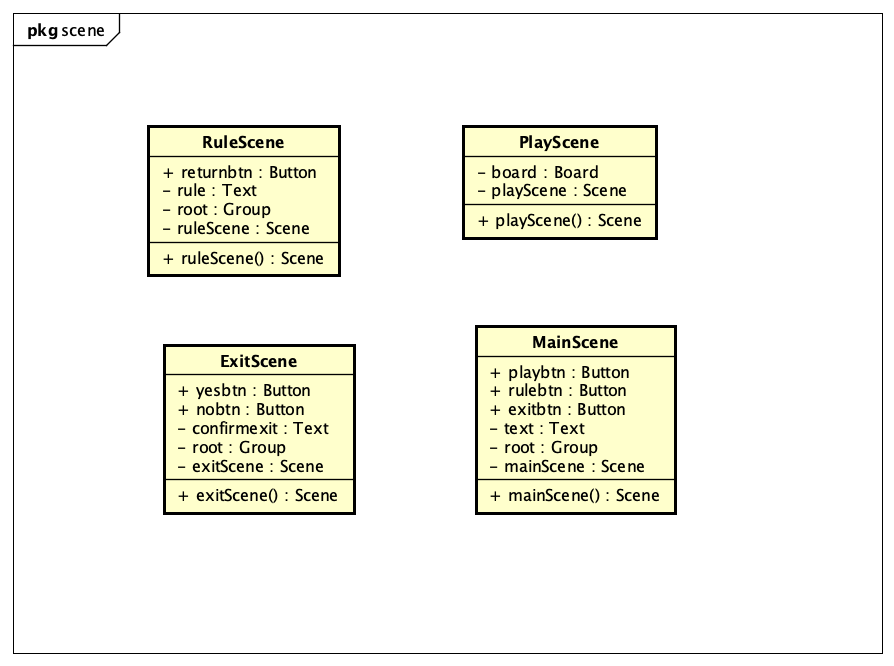
1. Class Diagrams:



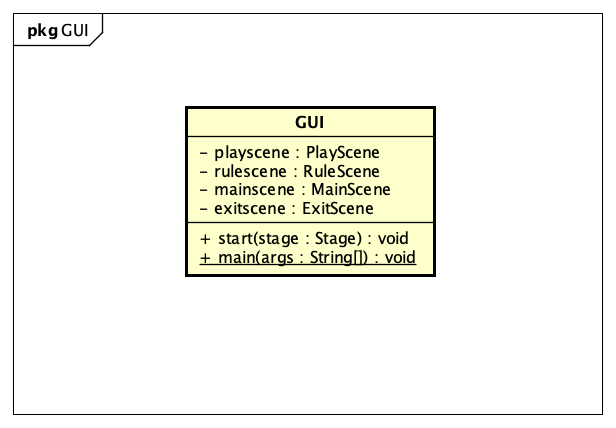
General Class Diagram



* Board package class diagram
* :



Scene package diagram



* GUI class diagram

1. Detailed explanation of classes and methods :

* The board package :
  + The cell package : Keeps tracks of the number of gems in every cell and their GUI components
    - KingCell : every object is a half-circle
    - NormalCell : every object is a single squares with ActionListener responding to player’s input
  + Stone class : GUI component for the gems
  + Player class : Keep tracks of each player’s scores and contains it’s GUI component
  + Board class : implementation of the rules of the game, and a javafx Group container with UI components in playing screen
    - We are storing the board in a 12-length array containing a cell in each from c0 to c11, with c0 and c6 being half-circles and the rest being squares
    - A turn variable and a pointer variable storing current turn and spot on the board
    - moveclockwise()/countermoveclockwise() : spread the gems in clockwise or counterclockwise direction
    - takeStone() : takes every gems that a player has captured during their turn
    - checkWinning() : return true if two half-cirles are empty and ends the game
    - returnStone() : check if during their turn, if a player has no gems on their side of the field.
      * If not, this method reduce 5 gems from their scores and spread those on their board, one in each square
      * If that player has less than 5 points, they automatically lose the game
    - printBoard() : reprint the number of gems in each cell after every move
* The scene package : Contains classes with methods that return their respective scene for the GUI class
  + ExitScene : exitScene() returns the confirmation scene to ask users if they really want to quit the game
  + MainScene : mainScene() where the user can choose to play, view rule or exit the software
  + PlayScene : playScene() contains the board class where users can play the game
  + RuleScene : ruleScene() display the rules of the game
* The GUI class : Javafx application with container stage for aforementioned scenes and main() method to run the software