

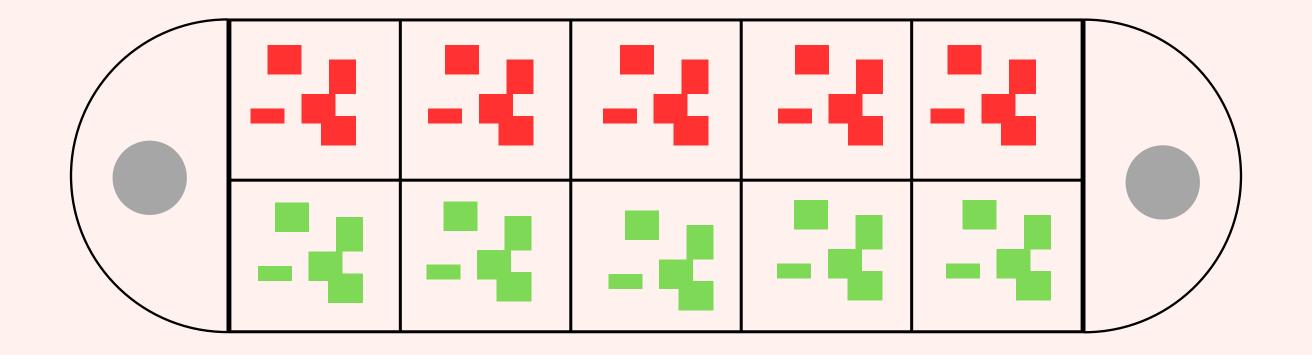
TABLE OF CONTENT

HOW TO PLAY

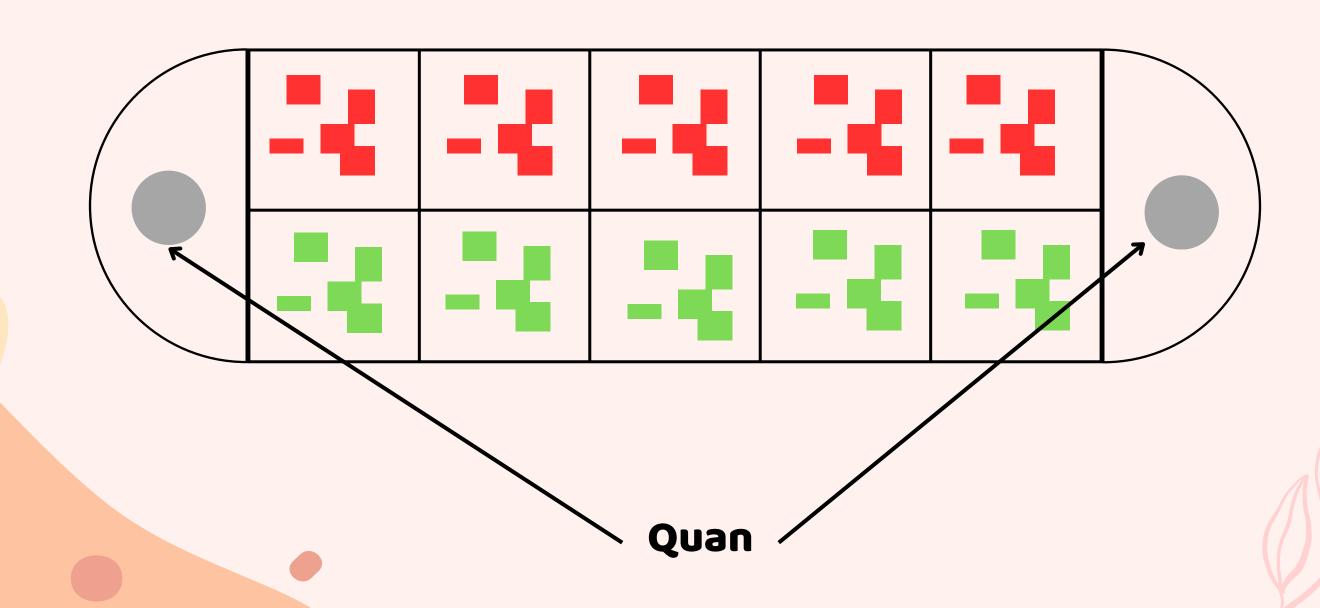
UML

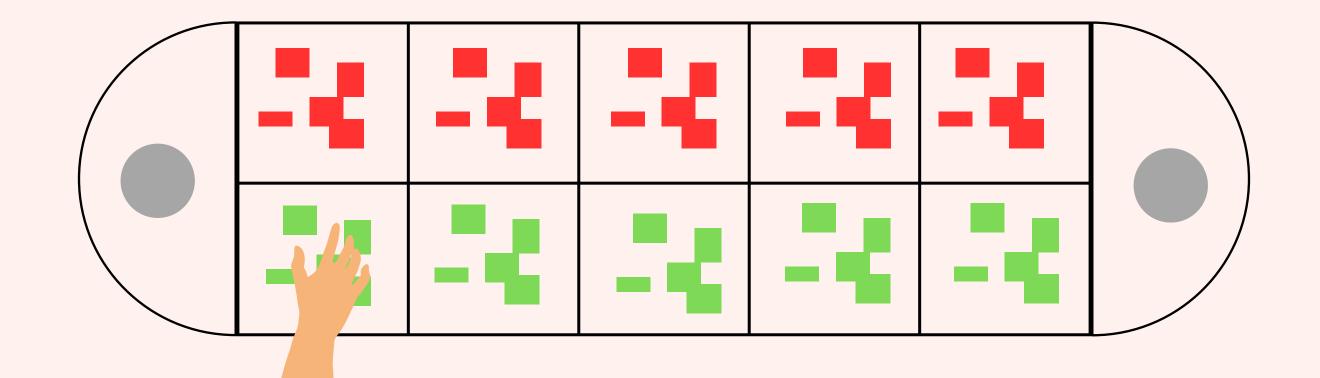
OOP TECHNIQUE

DEMO

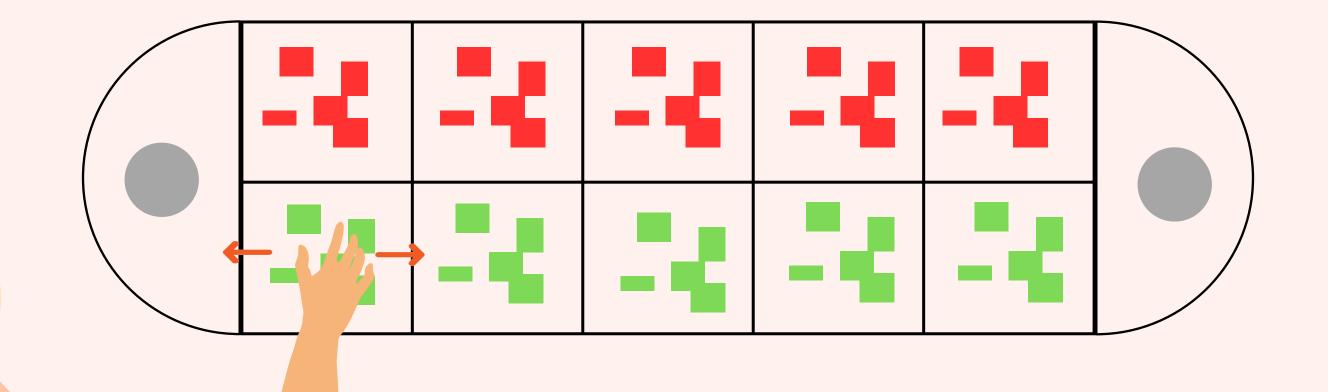


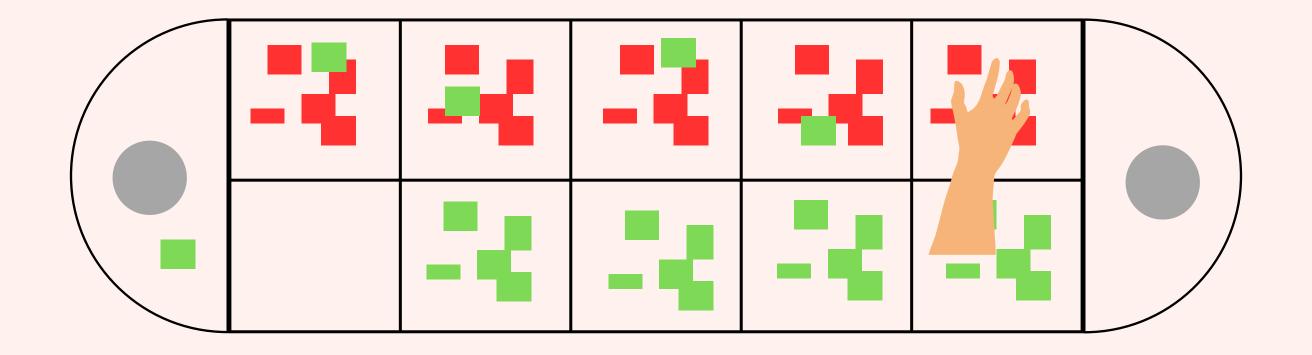




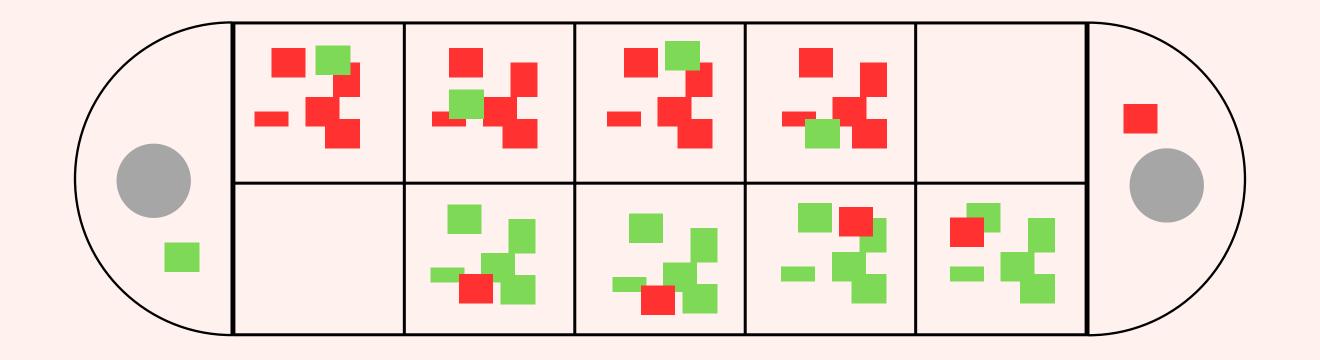




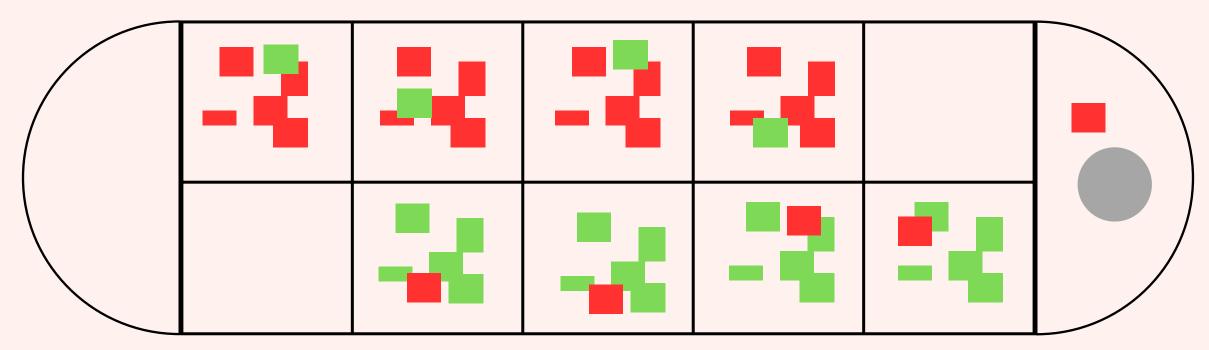


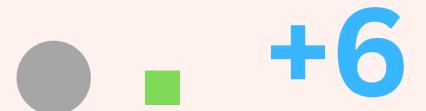






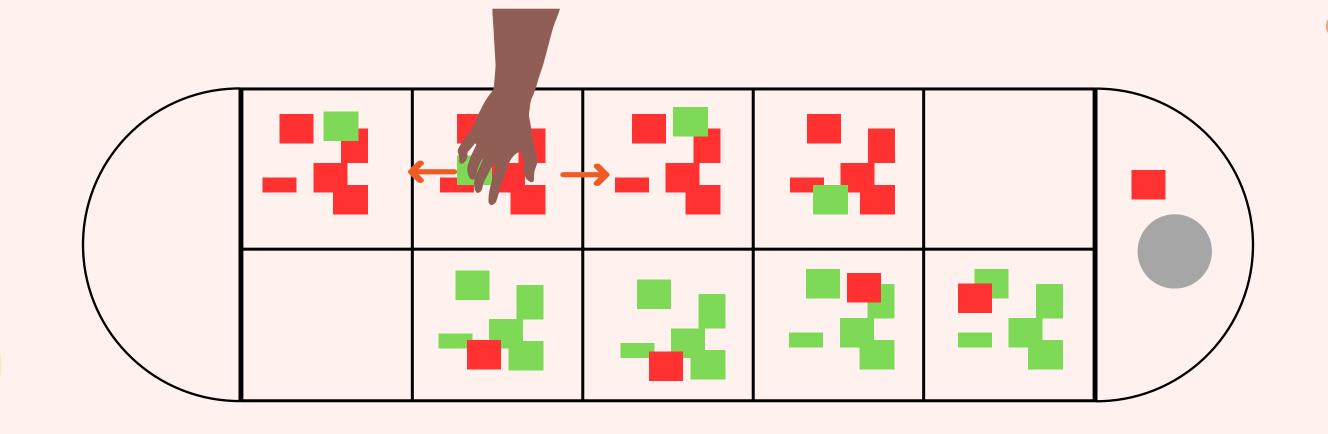










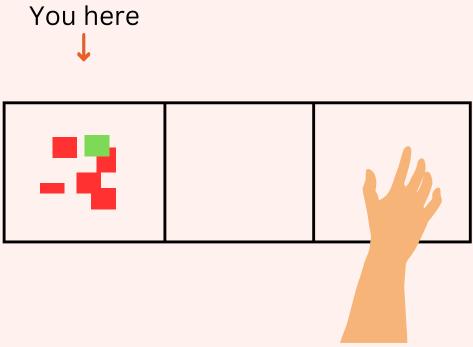


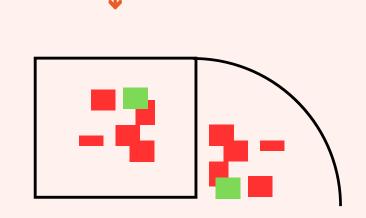


You here

STOP THE TURN AND EARN POINT



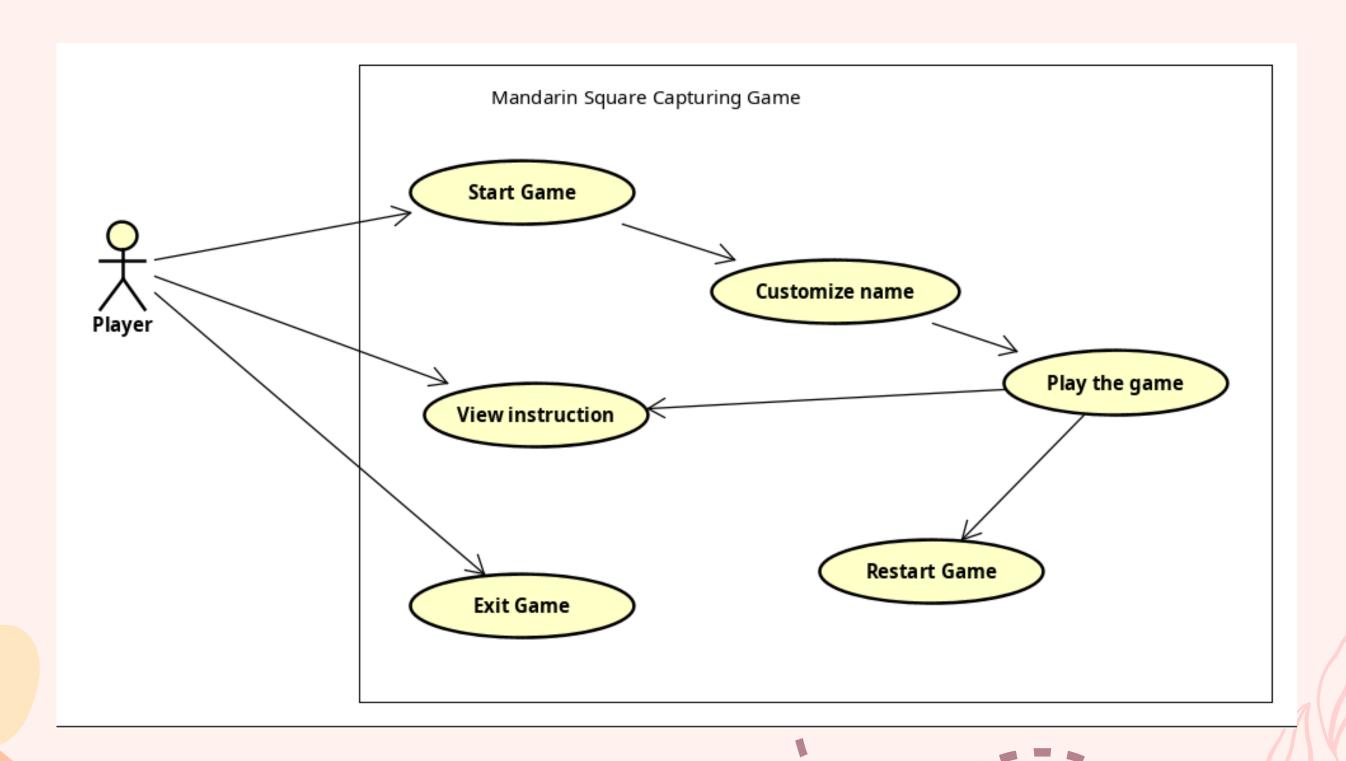




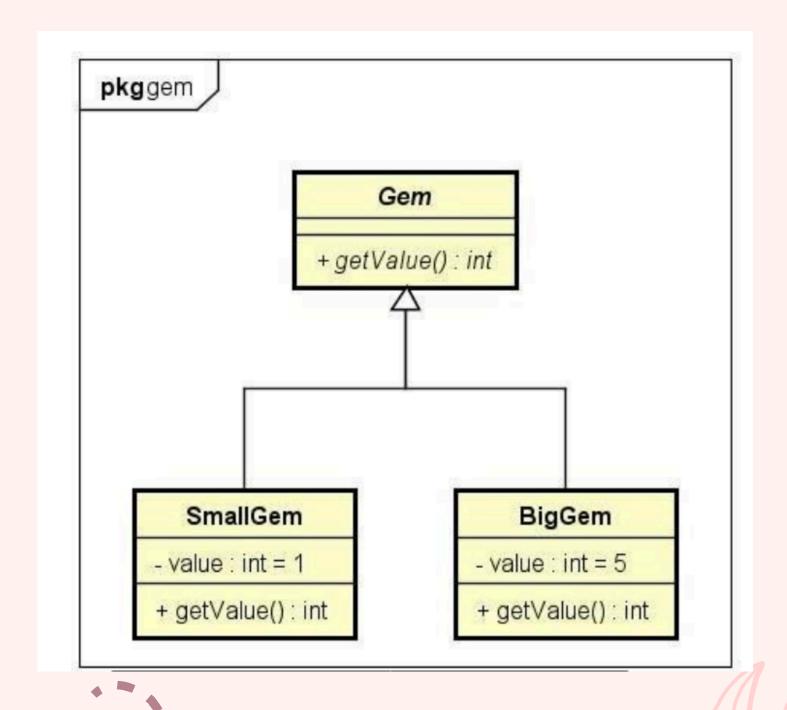
You here

STOP THE TURN WITHOUT EARNING POINT

USE CASE DIAGRAM



CLASS '- DIAGRAM



CLASS '- CLASS '- CLASS

pkg

Board

- numSquares : int = 10numHalfCircles : int = 2
- numBigGems : int = 2
- numSmallGems : int = 50
- + getNumSquares() : int
- + getNumHalfCircles(): int
- + getNumSmallGems(): int
- + getNumBigGems(): int
- + Board()
- initializeCells(): void
- addGemsToCells(): void
- + getCells() : Cell[]
- + getNextCellCounterClockwise(cell : Cell) : Cell
- + getNextCellClockwise(cell : Cell) : Cell
- + endGame() : boolean



CLASS '- DIAGRAM

pkg

Cell

- location : int
- + Cell(location : int)
- + getLocation(): int
- + getGemList() : List<Gem>
- + addGem(gem : Gem) : void
- + isEmpty() : boolean
- + setEmpty() : void
- + seeDetails(): String
- + getNumberOfBigGems(): int
- + getNumberOfSmallGems(): int
- + copyCell(): Cell



CLASS '- DIAGRAM

pkg

Competitors - turn : int direction : int - creditHistory : LinkedHashMap<Integer,Integer> = new LinkedHashMap<Integer,Integer>() + Competitors(player1 : Player, player2 : Player, board : Board) + getPlayer1(): Player + getPlayer2(): Player + getDirection(): int + setDirection(direction : int) : void + getTurn(): int + setTurn(turn : int) : void + getBoard(): Board + earnScore(earnedCell : Cell) : int + reduceScore(player : Player) : void + reduceScore(player : Player, score : int) : void + getWinner() : String + borrow(playerOne : Player, playerTwo : Player) : void + setItinerary(itinerary : List<Cell>) : void + getItinerary(): List<Cell> + assembleSmallGems(itinerary : List<Cell>) : void + spreadGems(player : Player, locationChosen : int, direction : int) : void + resetCreditHistory() : void + checkNoGemsOnSide(player : Player) : boolean + main(args : String[]) : void player2 player1 Player - score : int - name : String + Player(name : String) + getScore(): int + setScore(score : int) : void

+ getName() : String

+ computeScore(earnedScore : int) : void

+ equals(obj : Object) : boolean

CLASS DIAGRAM

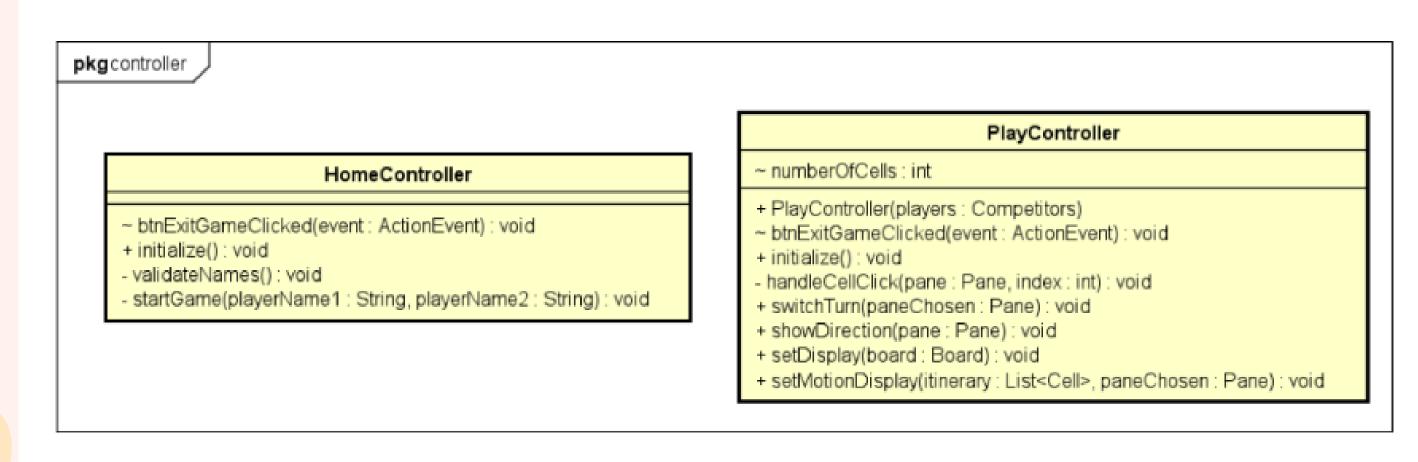
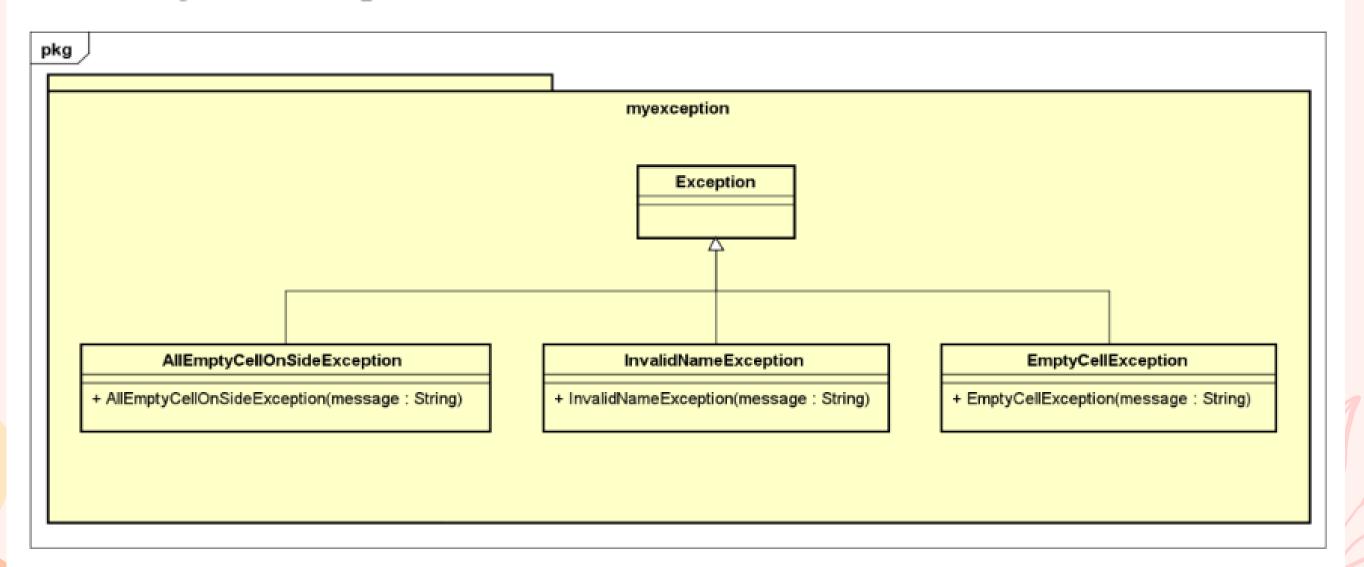


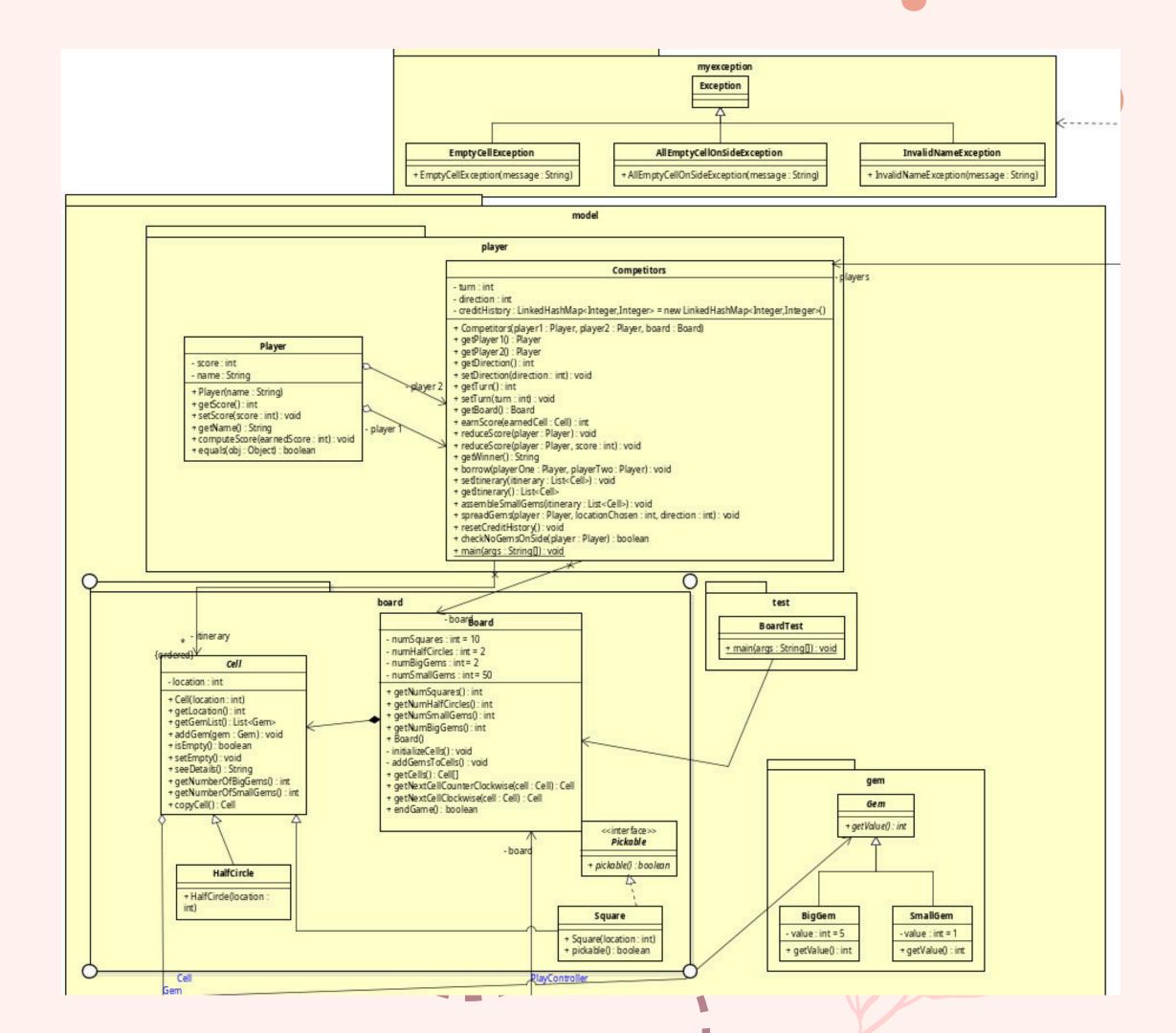
Figure 8. 'PlayController' and 'HomeController' Classes

CLASS DIAGRAM

Class Diagram: Exception



CLASS '- DIAGRAM



EXPLANATION OF OOP TECHNIQUE INHERITANCE

The class SmallGen and BigGem inherit from the abstract class Gem

2. The class HalfCircle and Square inherit from the class Cell

EXPLANATION OF OOP TECHNIQUE ENCAPSULATION

Classes in packkage models (like player, board and gem) use encapsulation to protect their private attributes Each class has its fields (attributes) and methods (functions) bundled together

EXPLANATION OF OOP TECHNIQUE ABSTRACTION

Pickable interface to distinguish the pickable cell and non-pickable one (for example, square and the half circle).

EXPLANATION OF OOP TECHNIQUE ASSOCIATION

The class Competitors is associated with the Players in a one-to-many relationship

Board are also associated with Cell in one-to-many relationship

EXPLANATION OF OOP TECHNIQUE AGGREGATION

Cell aggregates Gem

Competitors class is aggregation of player 1 and player 2

COMPOSITION

The board is composed of Cell objects

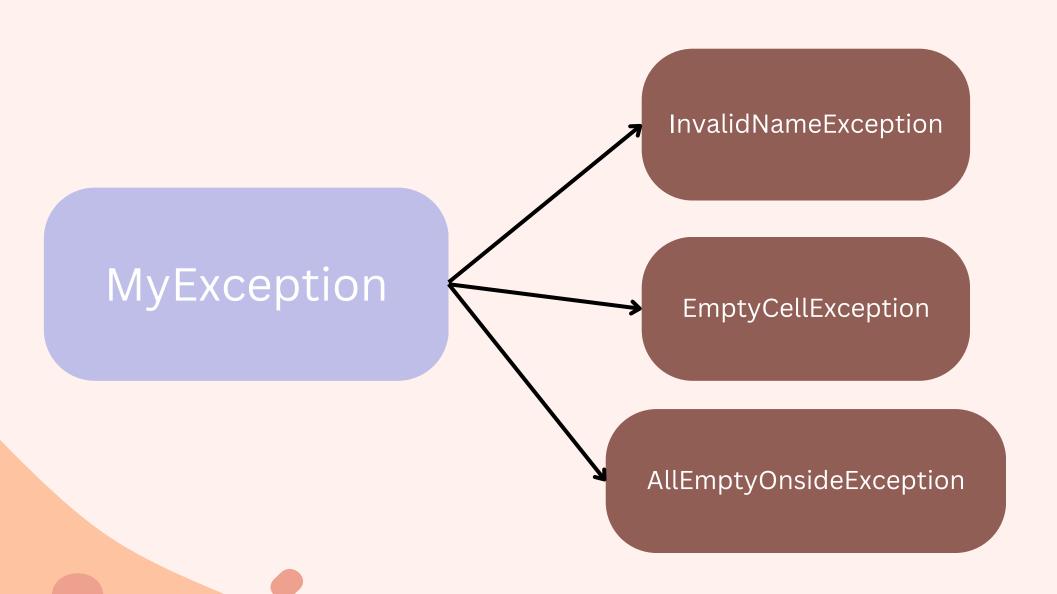
EXPLANATION OF OOP TECHNIQUE REALISATION

All squares implement Pickable Interface

POLYMORPHISM

It can be utilized in both BigGem and SmallGem as object of Gem class, but with distinct attributes and method

EXPLANATION OF OOP TECHNIQUE EXCEPTION



DEMO



