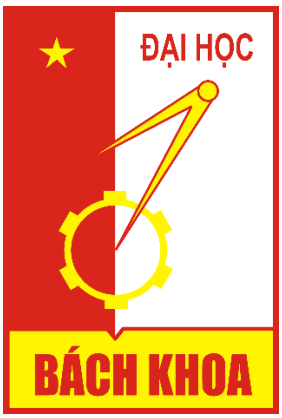
Đại học bách khoa Hà Nội

Trường Công nghệ thông tin và truyền thông

****



PROJECT
Object-Oriented Programming

Traditional game: Ô ăn quan

Giảng viên hướng dẫn : TS. Nguyễn Thu Trang

Nhóm 32: *Vũ Đức Thịnh 20210816*

Nguyễn Ngọc Anh Thư 20215143

Hoàng Văn Thuấn 20215140

Nguyễn Hoàng Ninh Thuận 20215141

Mục lục

I. Nhiệm vụ của các thành viên

1. Vũ Đức Thịnh 20210816

- Package gem: Gem.java, BigGem.java, SmallGem.java
- Package squares: Square.java, HalfCircle.java, NormalSquare.java
- Package boardgame: BoardGame.java
- Cung cấp ý tưởng và tham gia xây dựng usecase diagram, general and detail class diagram
- Demo video
- Tham gia viết Report

2. Nguyễn Ngọc Anh Thư 20215143

- Package screen: MenuScreen.java, HalfCircleScreen.java, NormalSquareScreen.java, SquareInterface.java, menu.fxml, play.fxml, rule.fxml
- Package controller: xử lí event chính trong PlayScreenController.java và New game button
- Vẽ lại usecase diagram, general and detail class diagram bằng astah
- Tham gia viết Report

3. Hoàng Văn Thuấn 20215140

- Package controller: xử lí event khi click vào button trong RuleScreenController, MenuScreenController, PlayScreenController.java
- Tham gia xây dựng usecase diagram, general and detail class diagram
- Tham gia viết Report

4. Nguyễn Hoàng Ninh Thuận 20215141

- Package player: Player.java
- Package match: Match.java
- Cung cấp ý tưởng và tham gia xây dựng usecase diagram, general and detail class diagram
- Presentation

II. Mô tả về mini-project

1. Mô tả chi tiết về yêu cầu của project

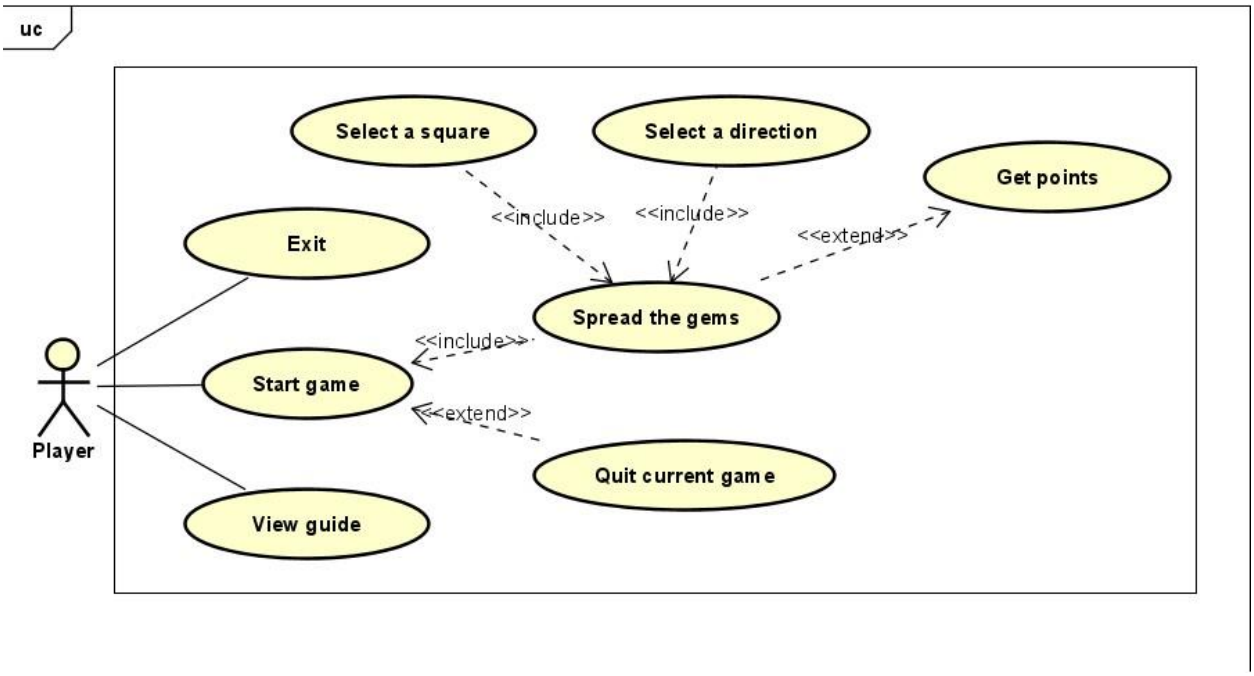
Trên màn hình chính:

- Bắt đầu: bắt đầu trò chơi. Để thuận tiện, không cần phải tạo ra các cấp độ khác nhau.
- Thoát: thoát khỏi chương trình. Hãy chắc chắn hỏi người dùng nếu họ muốn thoát khỏi trò chơi.
- Trợ giúp: Hiển thị hướng dẫn cho việc chơi game.

Trong trò chơi:

- Bảng game: Bảng game bao gồm 10 ô, chia thành 2 hàng và 2 nửa vòng tròn ở hai đầu của bảng. Ban đầu, mỗi ô có 5 viên ngọc nhỏ, và mỗi nửa vòng tròn có 1 viên ngọc lớn. Mỗi viên ngọc nhỏ tương đương 1 điểm, và mỗi viên ngọc lớn tương đương 5 điểm.
- Đối với mỗi lượt, ứng dụng phải rõ ràng hiển thị lượt của ai. Người chơi sẽ chọn một ô và hướng để rải các viên ngọc. Anh ta có điểm khi sau khi hoàn thành rải, có một ô trống liền sau một ô có ngọc. Điểm người chơi được cho lượt chơi đó bằng điểm số ngọc có trong ô đó.
- Trò chơi kết thúc khi không còn ngọc trong cả hai nửa vòng tròn. Ứng dụng phải thông báo người chiến thắng và điểm của mỗi người chơi.
- Để đơn giản, ta không cần phải xây dựng một bot để chơi với người chơi.

2. Use case diagram

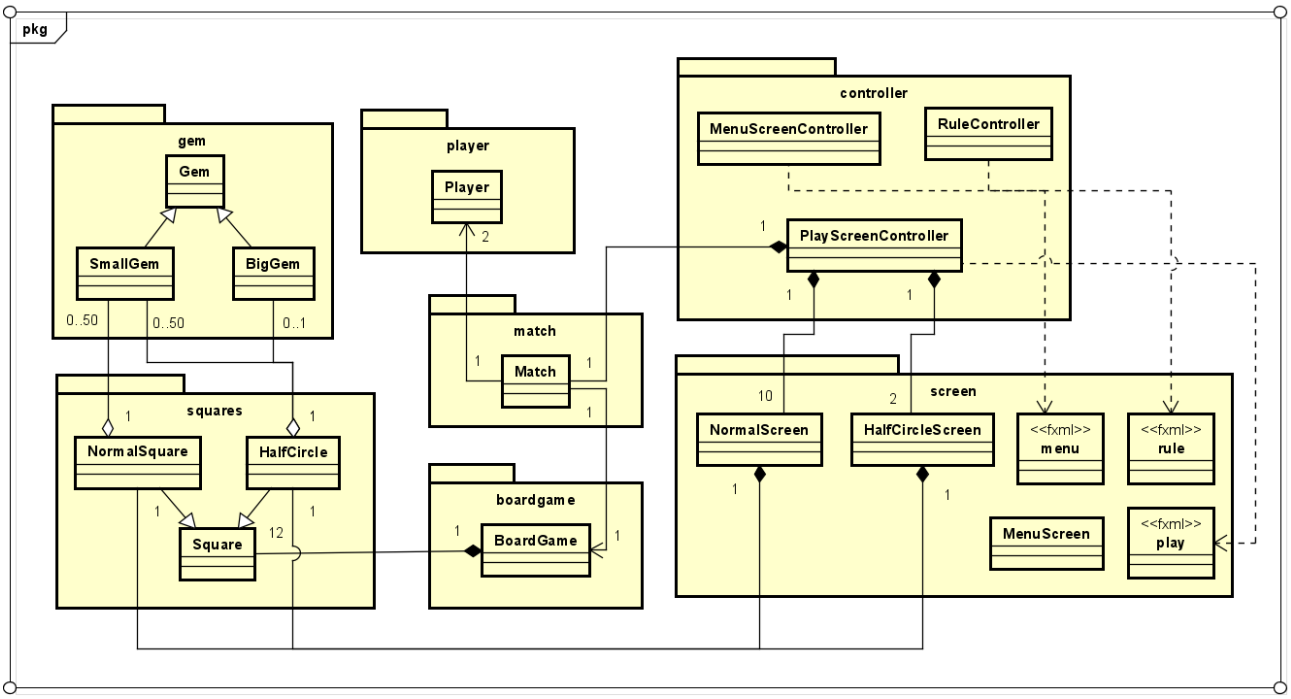


Giải trình:

- Người dùng có thể tương tác với trò chơi thông qua các chức năng : Start game, View guide, Exit.
- Game thực hiện các chức năng cơ bản của trò chơi ô ăn quan :
  + Spread the gems : rải đá :
  Select a direction : chọn hướng để thực hiện rải đá
  Get poin : tính điểm cho người chơi, tính điểm của từng ô trong trò chơi
  Select square : chọn ô để thực hiện rải đá
  + Quit current game : thoát game .

III. Design

1. General class diagram:



Packages :

- Gem : Lưu trữ tất cả đối tượng thuật toán liên quan đến gem như : getPoin, bigGem, smailGem.
- Squares : Gồm các thuật toán , đối tượng liên quan đến square như:

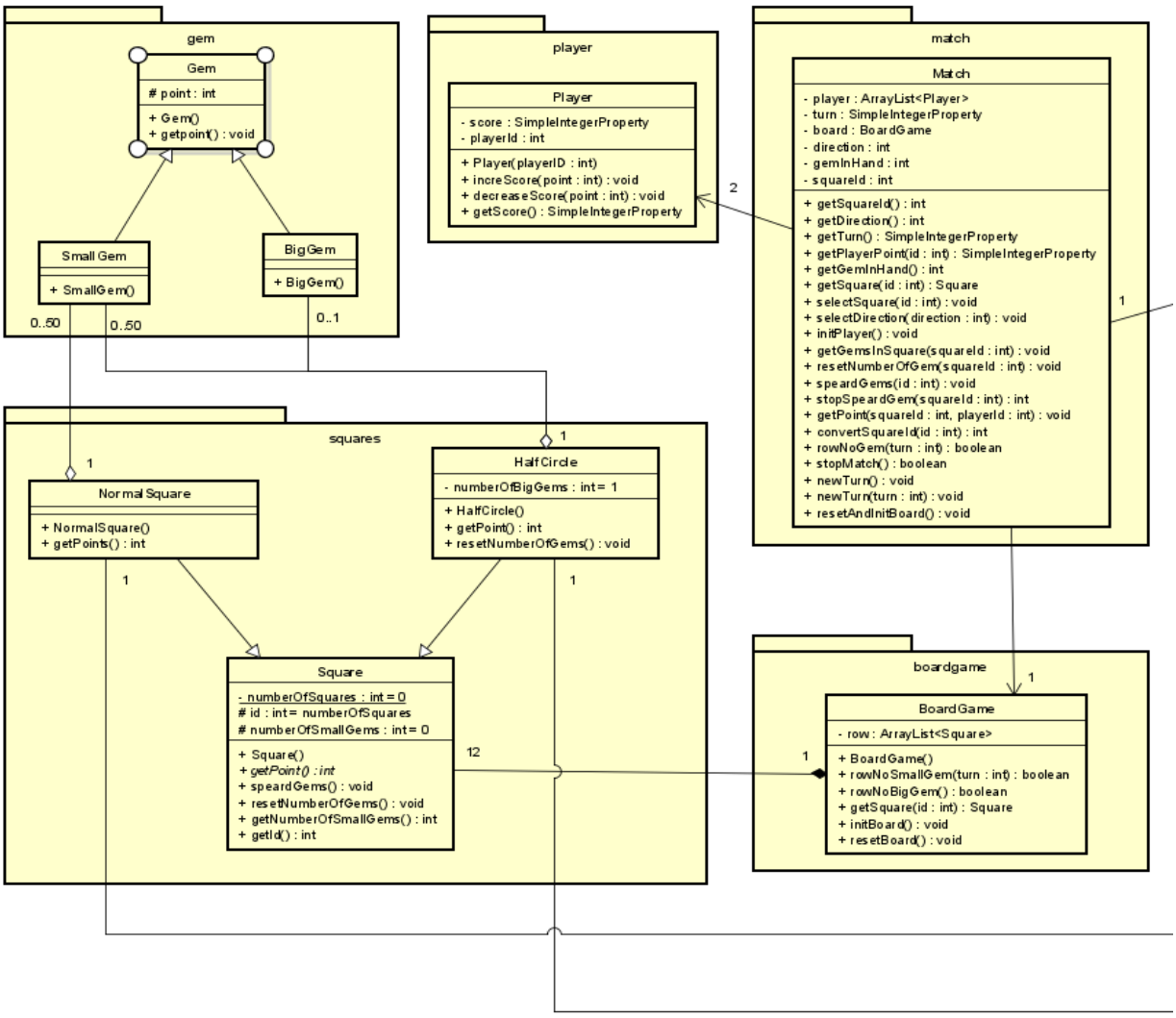
Normal alSquare

HalfCircle

Square

- Player : Gồm class Player
- Match : Gồm class Match
- Boardgame : Gồm class Boardgame
- Controller : Chứa các thuật toán , đối tượng liên quan đến controll game như : MenuScreenControll, PlayScreenControll, RuleControll
- Screen : Gồm các đối tượng liên quan đến screen game như : NormalScreen, HalfCircleScreen, MenuScreeen, thiết kế rule, play, menu.

2. Detail class diagram:



Package gem:

Gem:

+ Thuộc tính:

- int point: điểm của 1 gem

+ Method:

- getPoint(): trả lại điểm của gem

SmallGem:

+ Method:

- constructor: set điểm của gem bằng 1

BigGem:

+ Method:

- constructor: set điểm của gem bằng 5

Package squares:

Square:

+ Thuộc tính:

- static int numberOfSquares: số lượng Square, ban đầu bằng 0
- int id = numberOfSquares: id của square, mỗi square có 1 id khác nhau
- int numberOfSmallGems: số lượng SmallGem trong Square, ban đầu bằng 0

+ Method:

- constructor: tăng numberOfSquares thêm 1. khi tạo ra HalfCircle và NormalSquare (là lớp con của Square) id được gán bằng numberOfSquares và sau đó tăng numberOfSquares
- *getPoint()*: là 1 abstract method yêu cầu các lớp con override, trả lại điểm của Square
- spreadGems(): tăng numberOfSmallGems thêm 1 khi rải đá vào ô
- resetNumberOfSmallGem: set numberOfSmallGems = 0 khi lấy đá ra khỏi ô
- getters methods: trả về giá trị của id và numberOfSmallGems

NormalSquare:

+ Method:

- getPoint(): trả về điểm của NormalSquare bằng tích của điểm của 1 smallgem và numberOfSmallGems

HalfCircle:

+Thuộc tính:

- int numberOfBigGems: số lượng BigGem trong HalfCircle, ban đầu bằng 1

+ Method:

- getPoint(): trả về điểm của HalfCircle bằng tích của điểm của 1 smallgem và numberOfSmallGems cộng với tích của điểm của 1 biggem và numberOfBigGems
- resetNumberOfSmallGem: gọi resetNumberOfSmallGem của lớp cha và set numberOfBigGems = 0 khi lấy đá ra khỏi ô

Package boardgame:

BoardGame:

+ Thuộc tính:

- ArrayList<Square> row: mảng chứa các instance của các lớp con của Square

+ Method:

- initBoard(): khởi tạo row, vòng lặp for từ 0 đến 11, tại 0 và 6 thì tạo 1 HalfCircle và thêm vào row, các trường hợp còn lại tạo NormalSquare.
- resetBoard(): reset lại row
- rowNoSmallGem(int turn): kiểm tra xem hàng ô của người chơi hiện tại có chứa gem hay không, trả về true/false
- rowNoBigGem(): kiểm tra xem 2 ô HalfCircle có chứa gem hay không, trả và true/false
- getSquare(int id): trả về Square có chỉ số mảng bằng id trong row.

Package player:

Player:

+ Thuộc tính:

- SimpleIntegerProperty score: điểm của người chơi
- playerId: id của người chơi

+ Method:

- constructor(int id): set playerId = id
- increScore(int point): tăng điểm của người chơi
- decreScore(int point): giảm điểm của người chơi

- getScore(): lấy ra điểm của người chơi

Package match:

Match:

+ Thuộc tính:

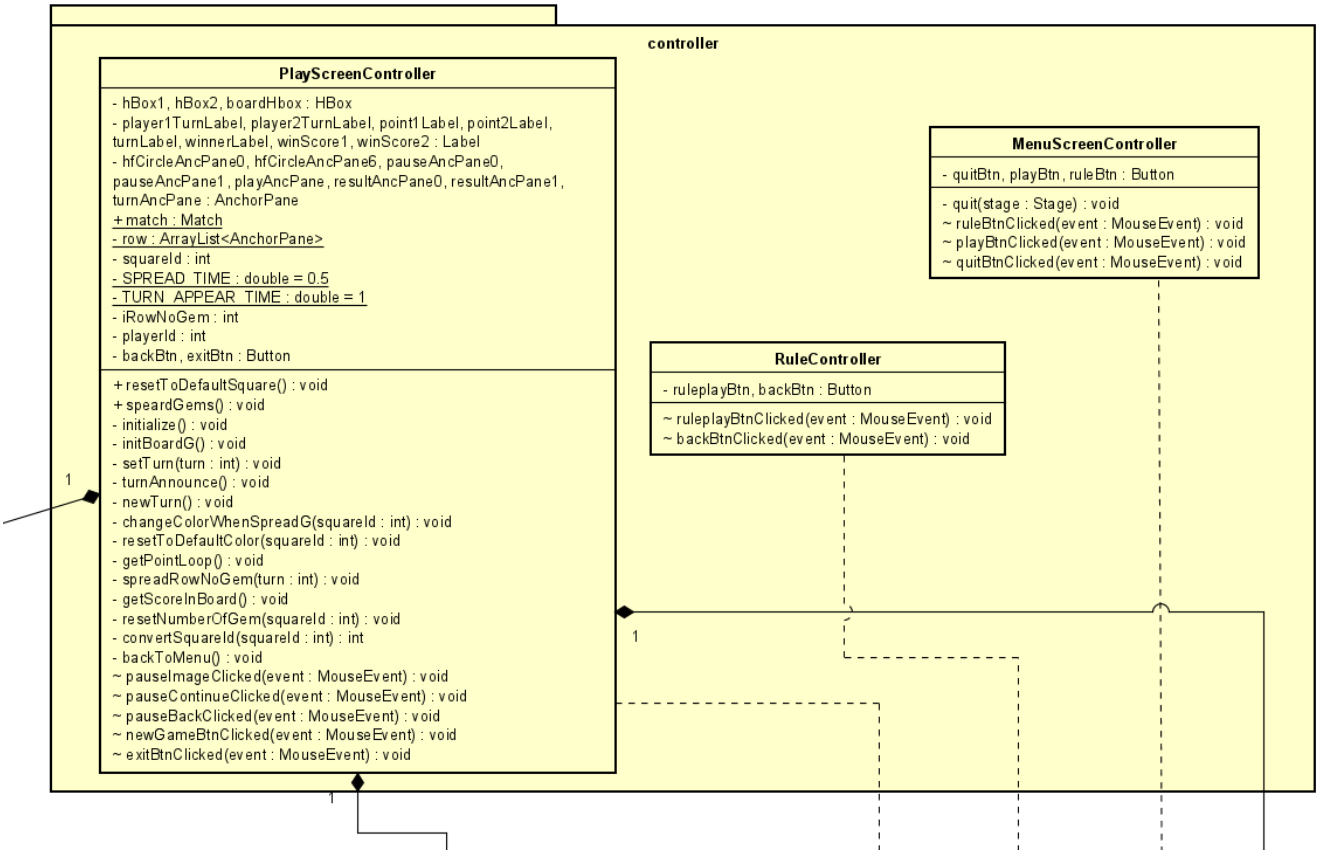
- ArrayList<Player>: mảng chứa các instance của Player
- SimpleIntergerProperty turn: lượt chơi chỉ ra đang là lượt của người chơi nào.
- BoardGame board: tạo bảng chơi
- int direction: hướng rải đá
- int gemInHand: đá lấy ra sau khi chọn ô
- int squareId: id ô mà người chơi chọn

+ Method:

- constructor: gọi initPlayer() khởi tạo mảng player
- getters methods: lấy ra các thuộc tính trong match
- selectSquare(int id): set squareId = id khi người chơi chọn ô
- SelectDirection(int direction): set direction = direction khi người chơi chọn hướng
- initPlayer(): khởi tạo mảng player chứa 2 instance của Player
- GetGemsInSquare(int squareId): set gemInHand = số lượng smallgem trong ô và reset lại số lượng đá khi lấy đá từ ô
- resetNumberOfGem(int squareId): reset lại số lượng gem
- spreadGems(int id): rải đá, gọi spreadGem của square có id bằng id truyền vào
- stopSpreadGem(int squareId): trả về int, -1 nếu dừng rải đá (khi ô liền cuối là 1 HalfCircle), 0 nếu tiếp tục rải đá (khi ô tiếp theo chứa đá), 1 nếu người chơi ăn điểm
- getPoint(int squareId, int playerId): lấy điểm từ ô và cộng cho người chơi
- convertSquareId(int squareId): chuyển đổi squareId, nếu squareId=12 thì set lại =0, còn nếu squareId=-1 thì set = 11 và trả lại squareId.
- rowNoGem(int turn): gọi method trong board để kiểm tra xem ô của ng chơi hiện tại có chứa đá hay không, trả về true/false

- stopMatch(): trả về kết quả kiểm tra xem 2 ô HalfCircle có chứa gem hay không
- newTurn(): chuyển turn sang cho người chơi còn lại
- newTurn(int turn): set turn = giá trị truyền vào
- resetAndInitBoard(): reset và khởi tạo lại bàn chơi

Detail class controller



Package Controller

1. MenuScreenController :

Thuộc tính:

- **@FXML:** Được sử dụng để chú thích trường (field) hoặc phương thức trong controller của FXML. Nó giúp FXMLLoader tìm và liên kết các phần tử trong FXML với trường hoặc phương thức tương ứng trong mã nguồn.

- **private Button playBtn, ruleBtn, quitBtn;**: Khai báo các biến đại diện cho các nút (Button) trong FXML. Bạn cần chú thích **@FXML** trước mỗi biến để kết nối chúng với các nút trong FXML.

Method:

- playBtnClicked(): Xử lý sự kiện khi nút "Play" được nhấn. Trong trường hợp này, nó tải FXML của màn hình chơi (play.fxml) và hiển thị nó.

- ruleBtnClicked(): Xử lý sự kiện khi nút "Rule" được nhấn. Trong trường hợp này, nó tải FXML của màn hình quy tắc (rule.fxml) và hiển thị nó.

- quitBtnClicked(): Xử lý sự kiện khi nút "Quit" được nhấn. Nó hiển thị một hộp thoại xác nhận và đóng ứng dụng nếu người dùng chọn OK.

2. RuleController:

Thuộc tính (@FXML):

- **@FXML private Button ruleplayBtn**: Đối tượng nút được liên kết với "Play" button trong FXML.
- **@FXML private Button backBtn**: Đối tượng nút được liên kết với "Back" button trong FXML.

Method :

- **@FXML private void ruleplayBtnClicked(ActionEvent event) throws IOException**: Phương thức xử lý sự kiện khi nút "Play" ở màn hình quy tắc được nhấn.
- **@FXML private void backBtnClicked(ActionEvent event) throws IOException**: Phương thức xử lý sự kiện khi nút "Back" được nhấn.

3. PlayScreenController

- **Thuộc tính:**

sPRAncPane: Đối tượng **AnchorPane** được chú thích bằng **@FXML**.

imageView1, **imageView2**, **imageView3**: Đối tượng **ImageView** được chú thích bằng **@FXML**.

hfCircleAncPane0, **hfCircleAncPane6**: Đối tượng **AnchorPane** được chú thích bằng **@FXML**.

boardHbox: Đối tượng **HBox** được chú thích bằng **@FXML**.

player1TurnLabel, **player2TurnLabel**: Đối tượng **Label** được chú thích bằng **@FXML**.

point1Label, **point2Label**: Đối tượng **Label** được chú thích bằng **@FXML**.

turnLabel: Đối tượng **Label** được chú thích bằng **@FXML**.

winnerLabel: Đối tượng **Label** được chú thích bằng **@FXML**.

winScore1, **winScore2**: Đối tượng **Label** được chú thích bằng **@FXML**.

pauseAncPane0, **pauseAncPane1**: Đối tượng **AnchorPane** được chú thích bằng **@FXML**.

playAncPane, **resultAncPane0**, **resultAncPane1**, **turnAncPane**: Đối tượng **AnchorPane** được chú thích bằng **@FXML**.

hboxSPR2, **hboxSPR1**: Đối tượng **HBox** được chú thích bằng **@FXML**.

- **Method:**

initialize(): Phương thức khởi tạo, được chú thích bằng **@FXML**.

resetToDefaultSquare(): Phương thức reset các ô vuông về trạng thái mặc định.

getGemsInSquare(int i): Phương thức lấy các viên đá từ ô vuông.

setTurn(int turn): Phương thức đặt lượt cho người chơi.

turnAnnounce(): Phương thức thông báo lượt chơi.

newTurn(): Phương thức bắt đầu lượt mới.

changeColorWhenSpreadG(int squareId): Phương thức thay đổi màu khi viên đá lan truyền.

resetToDefaultColor(int squareId): Phương thức reset màu của ô vuông về màu mặc định.

resetBoardToDefaultColor(): Phương thức reset màu của toàn bộ bảng về màu mặc định.

speardGems(): Phương thức lan truyền viên đá.

getPointLoop(): Phương thức lặp để lấy điểm.

spreadRowNoGem(int turn): Phương thức lan truyền viên đá trên dòng không có viên đá.

getScoreInBoard(): Phương thức lấy điểm khi kết thúc ván.

getGemsToPoint(int squareId): Phương thức lấy các viên đá vào ô điểm.

convertSquareId(int squareId): Phương thức chuyển đổi ID của ô vuông.

- Cửa sổ Pause:

@FXML void pauseBackClicked(MouseEvent event): Phương thức xử lý sự kiện khi người dùng nhấn nút "Back" trong màn hình tạm dừng.

Sử dụng FXMLLoader để tải FXML của màn hình menu (menu.fxml).

Gọi loader.load() để tạo ra một đối tượng Parent từ FXML.

Tạo một Scene mới từ đối tượng Parent.

Lấy ra Stage hiện tại và thiết lập Scene mới với màn hình menu.

Hiển thị màn hình menu.

@FXML void pauseContinueClicked(MouseEvent event): Phương thức xử lý sự kiện khi người dùng nhấn nút "Continue" trong màn hình tạm dừng.

Đặt hiệu ứng của playAncPane thành null để loại bỏ hiệu ứng GaussianBlur (mờ).

Đặt pauseAncPane0 thành không hiển thị (false).

@FXML void pauseImageClicked(MouseEvent event): Phương thức xử lý sự kiện khi người dùng nhấn vào hình ảnh để tạm dừng.

Đặt hiệu ứng của playAncPane thành một hiệu ứng GaussianBlur để làm mờ nền.

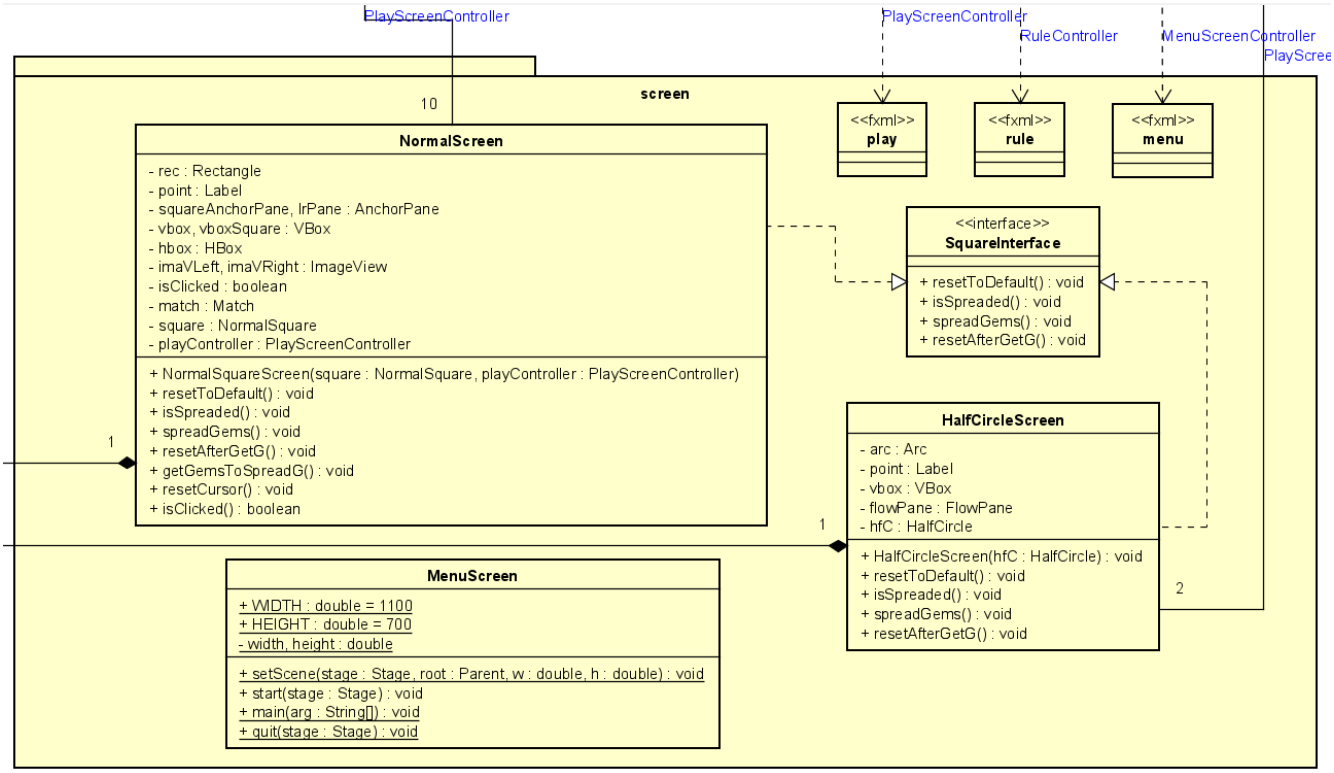
Đặt pauseAncPane0 thành hiển thị (true).

- Cửa sổ kết thúc ván chơi:

@FXML void newGameBtnClicked (MouseEvent event): tạo bàn chơi mới khi người dùng nhấn nút "New game" trong màn hình kết thúc ván chơi.

@FXML void exitBtnClicked (MouseEvent event): thoát ra màn hình menu khi người dùng nhấn nút "Exit" trong màn hình kết thúc ván chơi.

Detail class screen:



Package screen

MenuScreen:

+ Thuộc tính:

- Final static double WIDTH, HEIGHT: chiều rộng và chiều dài ban đầu của scene
- double width, height: tỉ lệ chiều rộng và chiều dài khi resize so với chiều rộng và chiều dài ban đầu

+ Method:

- static quit(Stage stage): set thông báo khi tắt màn hình
- static setScene(Stage stage, Parent root, double w, double h): set kích thước của scene khi xuất hiện và resize.

SquareInterface:

+ Method:

- resetToDefault(): reset lại color của square về mặc định
- isSpreaded(): set màu của square khi rải đá vào ô
- spreadGems(): thêm 1 circle (gem) và set lại point
- resetAfterGetG(): clear flow pane và set lại point của square sau khi get gems

NormalSquareScreen:

+ Thuộc tính:

- boolean isClicked: true nếu người chơi chọn ô, false nếu không chọn

+ Method:

- Override methods trong SquareInterface
- getGemToSpreadG(): khi người chơi chọn hướng thì lấy id của ô và lấy đá trong ô, set lại giao diện ô thành mặc định sau đó gọi spreadGem trong playController để rải đá.
- resetCursor(): chỉ hiện con trỏ chuột hình bàn tay ở những ô thuộc dãy của người chơi và chứa đá
- isClicked(): trả về thuộc tính isClicked

HalfCircleScreen:

+ Method:

- Override methods trong SquareInterface

3. Giải thích về thiết kế:

Vốn là trò chơi dân gian được biết đến qua bao nhiêu thế hệ, khá phổ biến với mọi người từ những thôn xóm đến những thành thị. Thiết kế của chúng tôi bao gồm :

Bàn chơi : Bàn chơi hình chữ nhật chia thành 10 ô vuông, mỗi bên 5 ô đối xứng (được gói là ô dân), ở 2 bên cạnh có 2 ô hình bán nguyệt (hoặc hình vòng cung) gọi là ô quan.

- Ô dân : được thiết kế tại class NormalSquare
- Ô quan : được thiết kế tại HalfCircleSquare

Quân chơi : gồm 2 loại quân quan(BigGem) và dân (SmallGem) với số lượng chia đều cho 2 player

Người chơi (Player) : có 2 người chơi , mỗi người một phía

Gói màn hình được viết bởi JavaFX cho thực hiện các tương tác giữa các thành phần trong game như nút , hướng di chuyển , hiển thị giao diện

Gói màn hình chứa lớp màn hình và bộ điều khiển để lấy đầu vào, điều khiển mô hình và biểu thị nó trên GUI cho người dùng.