Báo cáo Mini Project OOP Lý Thuyết
Thiết kế trò chơi truyền thống: Ô ăn quan

Nhóm 20
Giáo viên hướng dẫn: TS. Nguyễn Thị Thu Trang
Trợ giảng: Phạm Phan Anh

Ngày 25 tháng 12 năm 2024

- Nguyễn Gia Phong - 20205013
- Nguyễn Vũ Linh Phong - 20225902
- Lê Đồng Cảnh Phú - 20225755
- Trần Cao Bảo Phúc - 20225756
- Đỗ Hồng Quân - 20194650

Mục lục

1 Giới thiệu

1.1 Mô tả đề bài

Ở màn hình chính

- **Game mới:** Bắt đầu trò chơi. Không cần tạo các cấp độ khó khác nhau.
- **Thoát:** Thoát chương trình. Đảm bảo hỏi người dùng liệu họ có thực sự muốn thoát trò chơi.
- **Hướng dẫn:** Hiển thị hướng dẫn cách chơi trò chơi.

Trong trò chơi:
Bàn chơi:

- Bàn chơi gồm 10 ô vuông, chia thành 2 hàng, và 2 nửa hình tròn ở hai đầu bàn.
- Ban đầu, mỗi ô vuông có 5 viên sỏi nhỏ, mỗi nửa hình tròn có 1 viên sỏi lớn.
- Mỗi viên sỏi nhỏ tương ứng 1 điểm.
- Mỗi viên sỏi lớn tương ứng 5 điểm.

Lượt chơi:

- Ứng dụng phải hiển thị rõ lượt của người nào (ví dụ: làm nổi bật tên người chơi hoặc sử dụng các ràng buộc).
- Người chơi chọn một ô vuông và hướng để rải sỏi.
- Ghi điểm khi sau khi rải sỏi, một ô vuông trống và theo sau là ô vuông có sỏi.
- Điểm nhận được bằng số sỏi trong ô vuông tiếp theo.

Kết thúc trò chơi:

- Trò chơi kết thúc khi không còn viên sỏi nào trong cả hai nửa hình tròn.
- Ứng dụng phải thông báo người thắng và điểm số của từng người chơi.

Chức năng: Người chơi sẽ chọn một ô và hướng để rải sỏi. *Ghi chú:* Không cần xây dựng bot chơi với người.

1.2 Yêu cầu đề bài

Mô hình hóa lớp (Class modeling)
Player (Người chơi)

- Triển khai lớp `Player` với các thuộc tính như:
  - **Name:** Tên người chơi.
  - **Score:** Điểm số.
  - **Thông tin bổ sung:** Ví dụ: lịch sử lượt chơi.
- Lớp phải có các phương thức sau:
  - **Cập nhật điểm số:** Phương thức để thay đổi điểm số khi người chơi ghi điểm.
  - **Thực hiện lượt chơi:** Phương thức cho phép người chơi chọn ô và hướng rải sỏi.

Gem (Sỏi)

- Triển khai lớp `Gem` để đại diện cho:
  - **Sỏi nhỏ:** Có giá trị 1 điểm.
  - **Sỏi lớn:** Có giá trị 5 điểm.
- Lớp phải có các thuộc tính và phương thức:
  - **Thuộc tính:** Giá trị của sỏi.
  - **Phương thức:** Ví dụ:
    - `move()`: Di chuyển sỏi từ ô này sang ô khác.
    - `capture()`: Bắt sỏi trong lượt chơi.

Gameboard (Bàn chơi)

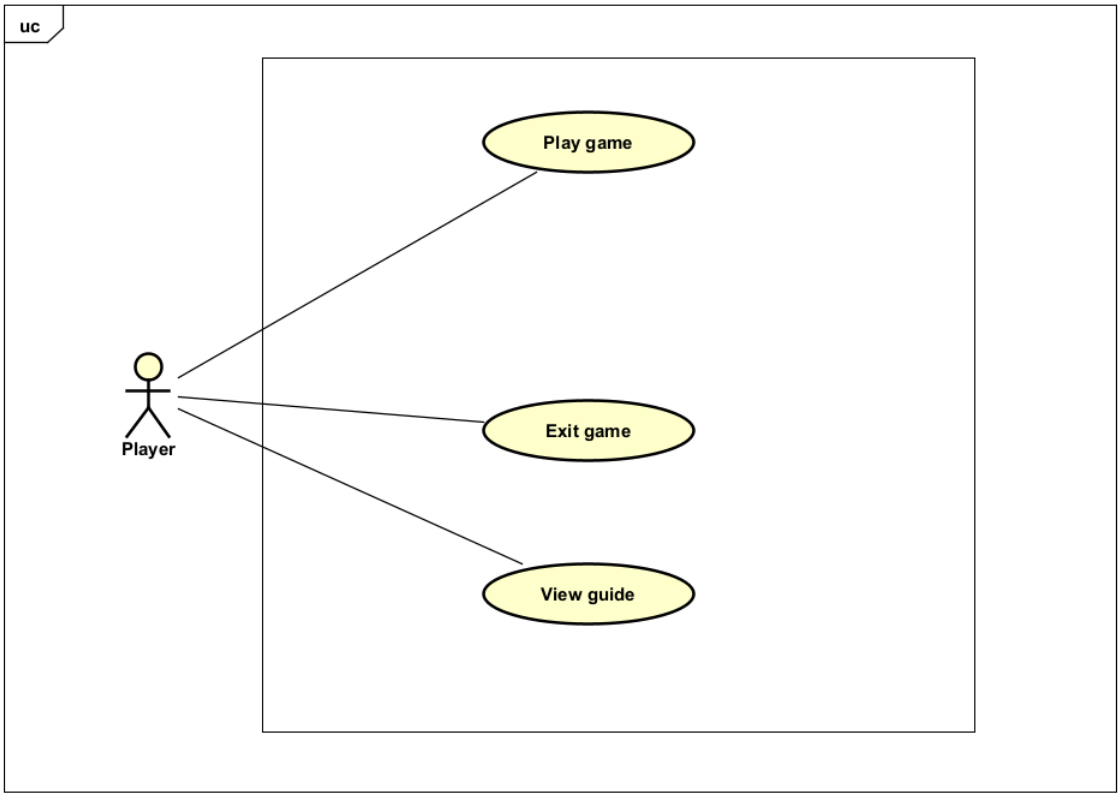
- Triển khai lớp `Gameboard` để mô hình hóa:
  - Các ô vuông: 10 ô vuông trên bàn chơi.
  - Các nửa hình tròn: 2 nửa hình tròn ở hai đầu bàn chơi.
- Lớp phải có các phương thức:
  - **Quản lý sỏi:** Đặt và di chuyển các viên sỏi.
  - **Tính điểm:** Kiểm tra điểm số dựa trên trạng thái bàn chơi.

Tương tác động (Dynamic interactions)

- Đảm bảo cơ chế chơi, chẳng hạn như rải sỏi và tính điểm, được triển khai dưới dạng các phương thức trong các lớp thích hợp.
- Tránh sử dụng:
  - Các hiệu ứng động dựng sẵn.
  - Logic cố định.
- Tất cả phải tương tác động dựa trên các thuộc tính và hành vi của đối tượng.

2 Design

2.1 Use Case Diagram

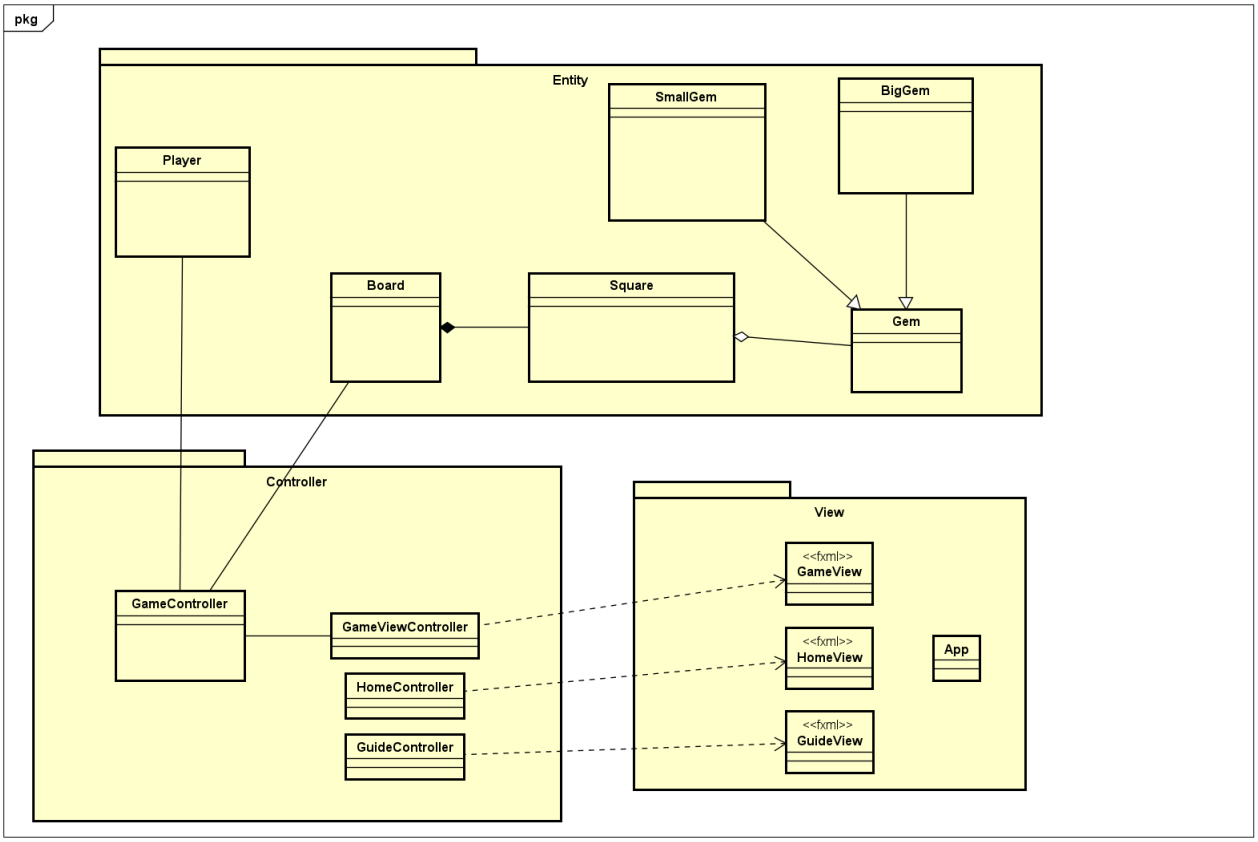


Giải thích:
Ở đây chúng em thiết kế 3 chức năng chính bao gồm:

- **Play game:** Chức năng được sử dụng khi người chơi muốn bắt đầu trò chơi. Người chơi sẽ được yêu cầu nhập tên của mình, sau đó ấn nút *Xác nhận* để bắt đầu trò chơi, nếu như người dùng không nhập tên vào thì tên mặc định sẽ là Player 1 và Player 2.
- **Exit game:** Người chơi sử dụng chức năng này để thoát khỏi chương trình. Trước khi thoát sẽ có một thông báo hiện lên để đảm bảo rằng họ muốn thoát khỏi chương trình.
- **View guide:** Chức năng được sử dụng khi người chơi muốn tìm hiểu về cách chơi cũng như là luật của trò chơi.

2.2 Class Diagram

2.2.1 General Class Diagram

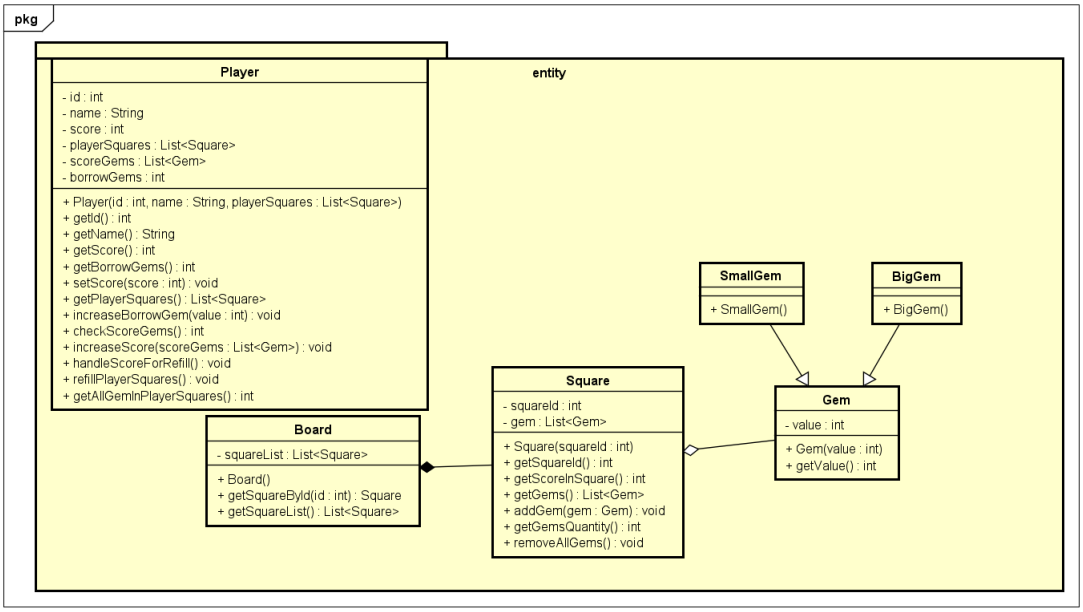


Packages

- Entity: Lưu trữ tất cả các thực thể như Gem, BigGem, Small Gem, Square, Board, Player

- Controller: Chứa các thuật toán, đối tượng liên quan tới controll game như: GameController, GameViewController, HomeController, GuideController
- View: Chứa các đối tượng liên quan tới màn hình game như App, thiết kế GameView, HomeView, GuideView

2.2.2 Detail Class Diagram



Package entity:
Player

- **Thuộc tính**:
  - `int id`: ID của người chơi
  - `String name`: Tên người chơi
  - `int score`: Điểm của người chơi
  - `List<Square> playerSquares`: Các ô của người chơi (Player 1: 0-4, Player 2: 5-9)
  - `List<Gem> scoreGems`: Danh sách gem mà player ăn được
  - `int borrowGems`: Số gem mà player mượn
- **Phương thức**:
  - `Player(int id,String name,List<Square> playerSquares): Constructor`

- `Getters and Setters`
- `increaseBorrowGem(int value)`: Tăng số gem mượn
- `int checkScoreGems()`: Đếm số small Gem trong số gem mà player ăn được
- `increaseScore(List<Gem> scoreGems)`Tăng điểm và thêm gem vào danh sách gem đã ăn
- `handleScoreForRefill()`: Xử lý điểm sau khi người chơi rải sỏi khi 5 ô của người chơi đều trống
- `refillPlayerSquares()`: Rải sỏi cho người chơi khi 5 ô của người chơi trống
- `getAllGemInPlayerSquares()`: Đếm tất cả gem trong ô của người chơi, lấy số lượng gem ở mỗi ô sau đó cộng dồn

Board

- **Thuộc tính**:
  - `List<Square> squareList`: Danh sách chứa các instance của Square
- **Phương thức**:
  - `Board()`: Khởi tạo bàn cờ, tạo các ô từ 0 tới 11, sau đó khởi tạo gem, 5 gem dân cho mỗi ô dân (1-5 và 7-11) còn 1 gem quan cho mỗi ô quan (0 và 6)
  - `getSquareById(int id)` : Trả về Square có chỉ số mảng bằng id
  - `getSquareList()`: Trả về danh sách các Square

Square

- **Thuộc tính**:
  - `int squareId`: id của square, mỗi square có một id khác nhau
  - `List<Gem> gems`: Danh sách gem trong ô
- **Phương thức**:
  - `Square(int squareId)`: Khởi tạo ô với ID và danh sách gem rỗng
  - `getSquareId()`: Lấy Id của ô
  - `getScoreInSquare()`: Tính tổng giá trị các gem trong ô

- `getGems()` : Trả về danh sách gem trong ô
- `addGem(Gem gem)`: Thêm gem vào ô
- `getGemsQuantity()`: Trả về số lượng gem trong ô
- `removeAllGems()` : Xóa tất cả gem trong ô

Gem

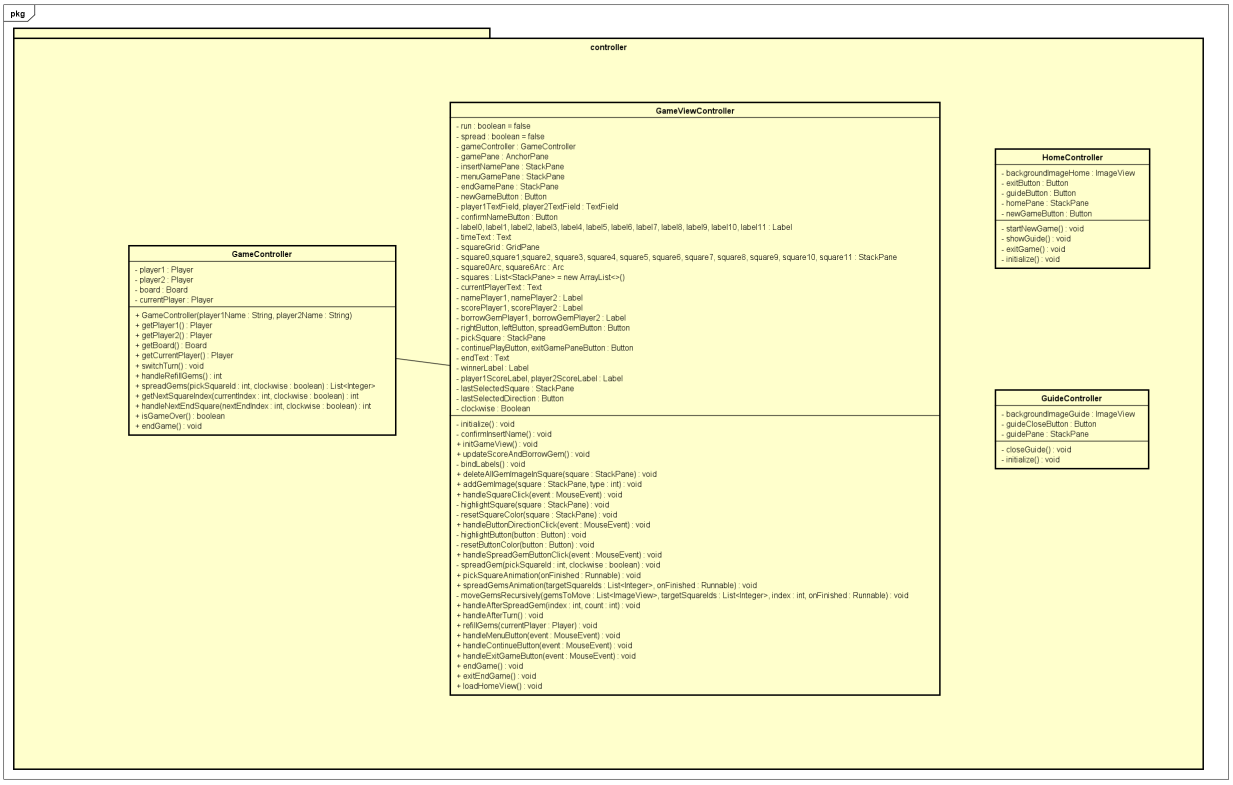
- **Thuộc tính**:
  - `int value`: điểm của 1 gem
- **Phương thức**:
  - `getValue()`: Trả lại điểm của 1 gem

SmallGem (kế thừa từ `Gem`)

- **Phương thức**:
  - `SmallGem()`: constructor set điểm của gem bằng 1

BigGem (kế thừa từ `Gem`)

- **Phương thức**:
  - `BigGem()`: constructor set điểm của gem bằng 5



Package controller:
HomeController

- **Thuộc tính**:
  - `@FXML private Button exitButton, guideButton, newGameButton`: Khai báo các biến đại diện cho các nút trong FXML, bạn cần chú thích @FXML trước mỗi biến để kết nối chúng với các nút trong FXML
- **Phương thức**:
  - `startNewGame()`: Tải FXML của màn hình chơi (GameView.fxml) và hiển thị nó
  - `showGuide()`: Tải FXML của màn hình hướng dẫn (GuideView.fxml) và hiển thị nó
  - `exitGame()`: Hiển thị hộp thoại xác nhận đóng ứng dụng và đóng ứng dụng nếu người dùng bấm OK.
  - `initialize()`: Phương thức khởi tạo

GuideController

- **Thuộc tính**:
  - `@FXML private Button guideCloseButton`: Đối tượng được kết nối với "Đóng" trong FXML
  - `@FXML private StackPane guidePane`: Đối tượng StackPane
- **Phương thức**:
  - `closeGuide()`: Đặt visible của guidePane = false, tải màn hình chính (HomeView.fxml)
  - `initialize()`: Phương thức khởi tạo

GameViewController

- **Thuộc tính**:
  - `boolean spread = false`
  - `GameController gameController`: Đối tượng GameController
  - `@FXML private AnchorPane gamePane`: AnchorPane cho màn chơi
  - `@FXML private StackPane insertNamePane`: màn nhập tên người chơi trước khi chơi
  - `@FXML private StackPane menuGamePane`: màn hình khi ấn nút Menu trong game
  - `@FXML private StackPane endGamePane`: màn end Game
  - `private TextField player1TextField, player2TextField`: Đối tượng TextField được chú thích bằng FXML
  - `private Button confirmNameButton`: Đối tượng Button được chú thích bằng FXML
  - `private Label label0,label1, label2, label3, label4, label5, label6, label7, label8, label9, label10, label11`: Đối tượng Label được chú thích bằng FXML
  - `private Text timeText`: Đối tượng Text được chú thích bằng FXML
  - `private GridPane squareGrid`: Đối tượng GridPane được chú thích bằng FXML
  - `private StackPane square0, square1, square2, square3, square4, square5, square6, square7, square8, square9, square10, square11`: Đối tượng StackPane được chú thích bằng FXML

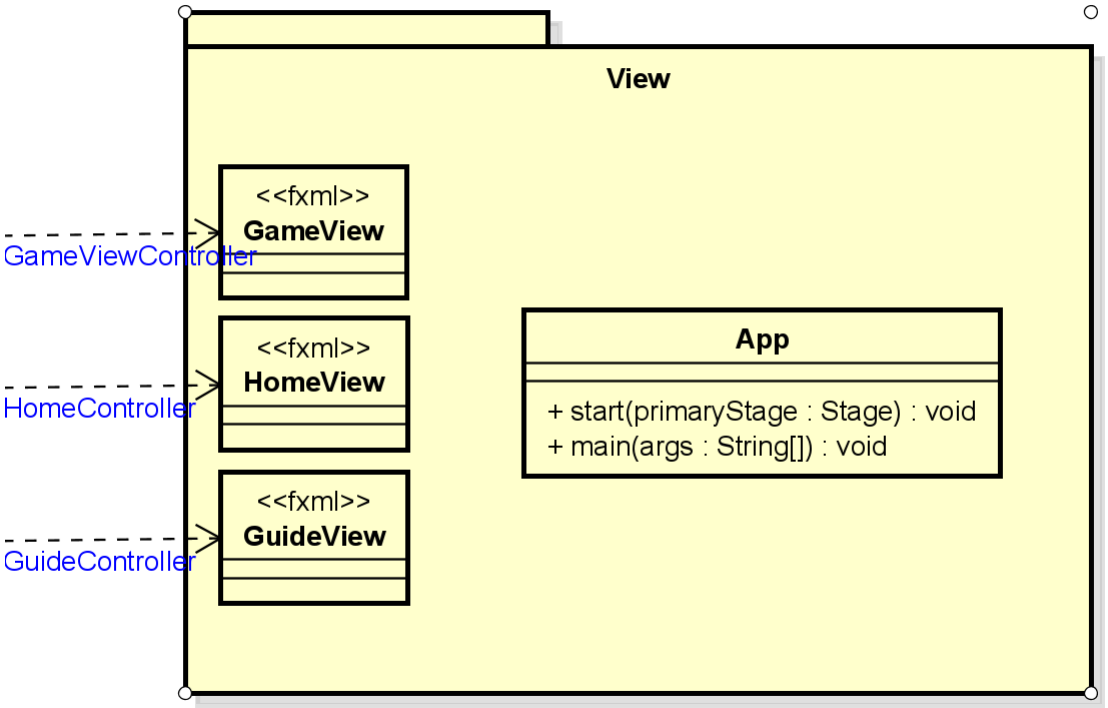
- `private Arc square0Arc, square6Arc`: Đối tượng Arc được chú thích bằng FXML
- `private final List<StackPane> squares = new ArrayList<>()`
- `private Text currentPlayerText`: Đối tượng Text được chú thích bằng FXML
- `private Label namePlayer1, namePlayer2, scorePlayer1, scorePlayer2, borrowGemPlayer1, borrowGemPlayer2`: Đối tượng Label được chú thích bằng FXML
- `private Button rightButton, leftButton, spreadGemButton`: Đối tượng Button được chú thích bằng FXML
- `private StackPane pickSquare`: Đối tượng StackPane được chú thích bằng FXML
- `private Text endText`: Đối tượng Text được chú thích bằng FXML
- `private Label winnerLabel, player1ScoreLabel, player2ScoreLabel`: Đối tượng Label được chú thích bằng FXML
- `private StackPane lastSelectedSquare`: ô được chọn
- `private Button lastSelectedDirection`: hướng được chọn
- `private Boolean clockwise`: chiều được chọn

- **Phương thức**:
  - `initialize()`: Phương thức khởi tạo
  - `initGameView()`: Khởi tạo màn hình GameView, bao gồm các gem, tên người chơi
  - `updateScoreAndBorrowGem()`: Cập nhật điểm và số đá đã mượn
  - `bindLabels()`: Cập nhật số gem ở mỗi ô
  - `deleteAllGemImageInSquare(StackPane square)`: Xóa hình ảnh gem ở ô vuông
  - `addGemImage(StackPane square, int type)`: Thêm hình ảnh Gem
  - `handleSquareClick(MouseEvent event)`: Xử lý sự kiện khi người chơi click vào ô vuông, khi ô tương ứng được chọn thì sẽ ô vuông sẽ chuyển sang màu vàng.
  - `highlightSquare(StackPane square)`: Highlight ô vuông được chọn

- `resetSquareColor(StackPane square)`: Reset màu của ô về màu mặc định
- `handleButtonDirectionClick(MouseEvent event)`: Xử lý sự kiện khi người chơi chọn hướng
- `highlightButton(Button button)`: Highlight nút được chọn
- `resetButtonColor(Button button)`: Reset nút về màu mặc định
- `handleSpreadGemButtonClick(MouseEvent event)`: Xử lý sự kiện khi bấm nút "Spread Gem"
- `spreadGem(int pickSquareId, boolean clockwise)`: Xử lý việc chuyển đá từ ô pickSquareId sang các ô khác
- `pickSquareAnimation(Runnable onFinished)`: Hoạt ảnh di chuyển tới vị trí pickSquare
- `spreadGemsAnimation(List<Integer> targetSquareIds,Runnable onFinished)`: Xử lý hoạt ảnh Spread Gem
- `moveGemsRecursively(List<ImageView> gemsToMove, List<Integer> targetSquareIds, int index, Runnable onFinished)`: Hàm đệ quy cho việc spread gem
- `handleAfterSpreadGem(int index,int count)`: Hàm xử lý sau khi đã Spread Gem
- `handleAfterTurn()`: Xử lý sau mỗi lượt chơi, nếu 2 ô quan trống hoặc đối phương không đủ đá cho mượn thì trò chơi sẽ kết thúc, nếu 5 ô của người chơi trống thì sẽ rải đá, còn không thì tiếp tục trò chơi.
- `refillGems( Player currentPlayer)`: Rải đá cho người chơi
- `handleMenuButton(MouseEvent event)`: Hàm xứ lý sự kiện bấm nút Menu
- `handleContinueButton(MouseEvent event)`: Hàm xử lý sự kiện ấn nút Tiếp tục
- `handleExitGameButton(MouseEvent event)`: Hàm xử lý sự kiện khi ấn nút Thoát
- `endGame()`: Hàm xử lý khi kết thúc trò chơi
- `exitEndGame()`: Thoát khỏi trò chơi, về màn hình chính
- `loadHomeView()`: Về lại màn hình chính khi dừng cuộc chơi

GameController

- **Thuộc tính**:
  - `Player player1, player2`: Người chơi 1, người chơi 2
  - `Board board`: Bàn cờ
  - `Player currentPlayer`: Người chơi hiện tại
- **Phương thức**:
  - `GameController(String player1Name, String player2Name)`: Constructor, đặt người chơi Player1 là người đi trước
  - `Các getters`
  - `switchTurn()`: Đổi lượt chơi
  - `handleRefillGems()`: Hàm xử lý việc rải gem khi 5 ô của người chơi trống
  - `List<Integer> spreadGems(int pickSquareId, boolean clockwise)`: Hàm xử lý việc rải đá, clockwise = true là theo chiều kim đồng hồ
  - `handleNextEndSquare(int nextEndIndex, boolean clockwise)`: Hàm xử lý khi kết thúc một lượt rải đá
  - `boolean isGameOver()`: Kiểm tra xem game có kết thúc không
  - `endGame()`: Kết thúc game



Package view:
App

- **Phương thức**:
  - `public void start(Stage primaryStage) throws Exception`: Bắt đầu chương trình

2.2.3 Giải thích thiết kế

Ô ăn quan là một trò chơi gắn liền với tuổi thơ của nhiều người, là một món ăn tinh thần không thể thiếu của người dân Việt Nam. Thiết kế của chúng tôi bao gồm:
Bàn chơi: Bàn chơi hình chữ nhật gồm 10 ô vuông, mỗi bên 5 ô đối xứng (gọi là ô dân), ở 2 bên cạnh có 2 ô hình bán nguyệt gọi là ô quan. Cả ô dân và ô quan được thiết kế ở class Square. Ô quan được đánh số squareId là 0 và 6.
Quân chơi: gồm 2 loại quân là dân (SmallGem) và quan (BigGem) với số lượng chia đều cho 2 player.
Người chơi (Player) gồm 2 người chơi, mỗi người một phía.
Gói view được viết bởi JavaFX cho thực hiện các tương tác giữa các thành

phần trong game như nút, di chuyển, hiển thị giao diện.
Gói controller chứa các bộ điều khiển, lấy đầu vào, điều khiển mô hình và hiển thị nó trên GUI của người dùng.

3 Phân công công việc

3.1 Nguyễn Gia Phong 20205013

- Package controller: GameController.java, chỉnh sửa GameViewController.java
- Package entity: Player.java, chỉnh sửa Board.java, Square.java
- Package view: GameView.fxml
- Design: Thiết kế Detail Class Diagram, General Class Diagram
- Image

3.2 Nguyễn Vũ Linh Phong 20225902

- Package controller: GameController.java
- Package entity: Player.java, chỉnh sửa Board.java, Square.java
- Design: Thiết kế Use-case diagram, chỉnh sửa Detail Class Diagram
- Presentation

3.3 Lê Đồng Cảnh Phú 20225755

- Package view: HomeView.fxml, GuideView.fxml, chỉnh sửa GameView.fxml
- Package controller: HomeController.java, GuideController.java
- Package app: App.java
- Demo Video, viết report, reorganize project

3.4 Trần Cao Bảo Phúc 20225756

- Package view: GameView.fxml
- Package controller: GameViewController.java, thiết kế các chức năng cơ bản GameController.java
- Tham gia viết report

3.5 Đỗ Hồng Quân 20194650

- Package entity: Gem.java, BigGem.java, SmallGem.java, Square.java, Board.java
- Tham gia viết report