ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
----------*----------

BÁO CÁO MINI-PROJECT

HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
MÃ HỌC PHẦN: IT3103
MÃ LỚP: 143577

NHÓM 27 - TOPIC 8: Ô ĂN QUAN

Họ Và Tên	*MSSV*
Dương Minh Phúc(TL)	20194648
Phạm Hữu Phúc	20215119
Trần Đăng Phúc	20215120
Phạm Hiếu Phương	20215121

Mục Lục

1.Phân công công việc

Họ Và Tên	*Công việc*
Dương Minh Phúc	Quản lý git, General Class Diagram, Thiết kế Details Class Diagram, Console, Chỉnh sửa GUI Player With Bot.
Phạm Hữu Phúc	Use-case diagram, Shape, Board, Slide, Report.
Trần Đăng Phúc	Chỉnh sửa Details Class Diagram, Gem, Player, Thêm tính năng Play with Bot (Board, Player, Thêm class Move).
Phạm Hiếu Phương	Thiết kế hình ảnh mà nhóm sử dụng, Xây dựng GUI.

2.Mô tả project

2.1.Yêu cầu của ứng dụng trò chơi Ô Ăn Quan

- On the main screen:

+ Start: bắt đầu trò chơi

+ Play with bot: chơi với máy

+ Quit: thoát trò chơi. Xác nhận lại với người dùng trước khi thoát game

+ Rule: đưa ra hướng dẫn cách chơi trò chơi

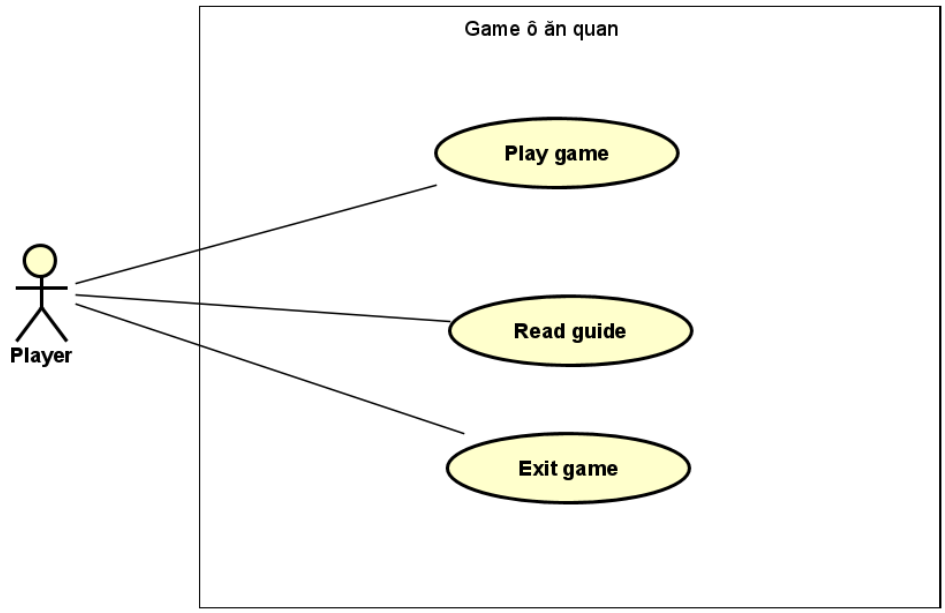
- Gameplay

+ Gameboard: bàn chơi gồm có 10 hình vuông – ô dân (chia thành 2 hàng) và 2 hình bán nguyệt – ô quan ở 2 đầu của bàn. Ban đầu, mỗi hình vuông gồm có 5 viên sỏi nhỏ– quân dân, và mỗi hình bán nguyệt có 1 viên sỏi lớn – quân quan. Mỗi viên sỏi nhỏ có giá trị 1 điểm, mỗi viên sỏi lớn có giá trị 5 điểm.

+ Trò chơi chỉ ra hiện tại là lượt chơi của người chơi nào. Mỗi lượt, người chơi sẽ chọn 1 hình vuông và 1 hướng (phải hoặc trái) để rải sỏi. Người chơi sẽ ăn điểm nếu như sau khi rải, liền sau đó là một ô trống rồi mới đến một ô có chứa quân.Hoặc nếu liền sau ô có quân đã ăn có thêm một ô trống nữa rồi đến một ô có quân thì người chơi có quyền ăn tiếp cả quân ở ô này. Điểm số được quy đổi bằng tùy vào số sỏi nhỏ và sỏi lớn ăn được.

+ Trò chơi kết thúc khi không còn sỏi ở cả 2 hình bán nguyệt. Trò chơi sẽ thông báo người chơi nào chiến thắng và số điểm của mỗi người.

2.2.Use case diagram

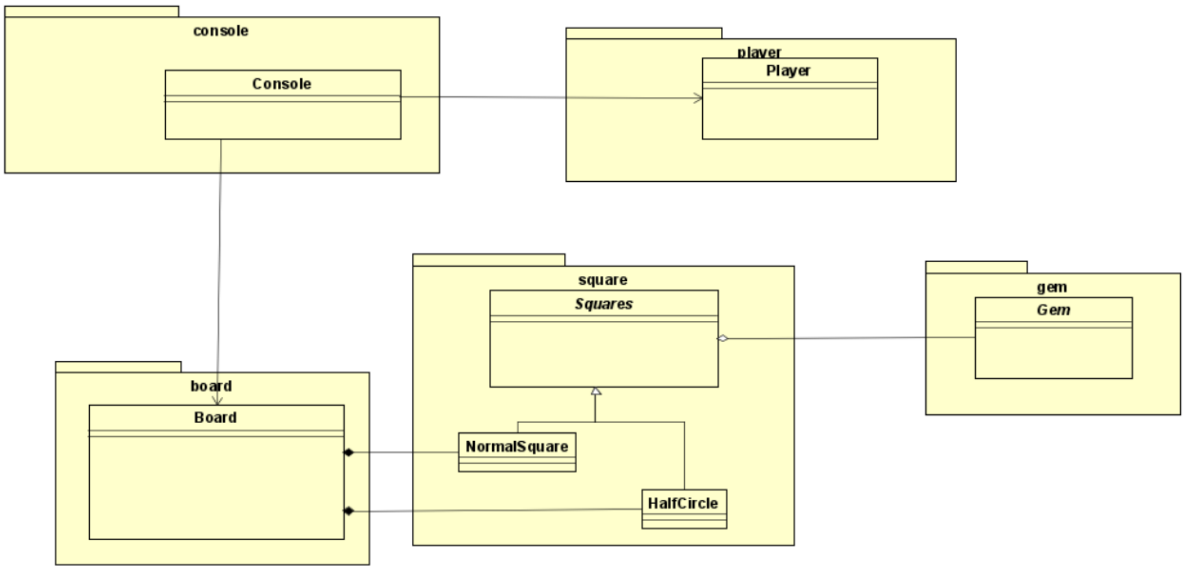


Tương tác giữa người chơi và ứng dụng:

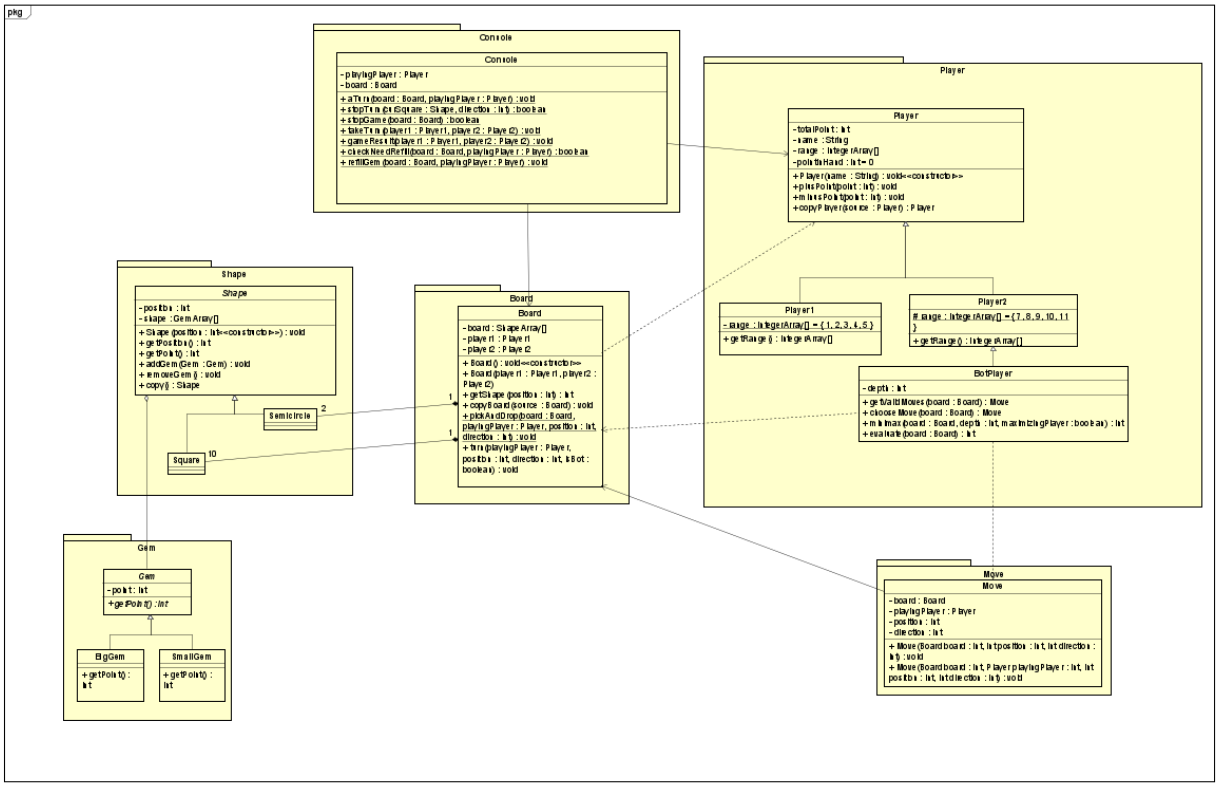
- Player chơi trò chơi
- Player đọc hướng dẫn cách chơi
- Player thoát trò chơi

3.Thiết kế

3.1.General class diagram



3.2.Detail class diagram



3.3.Phân tích thiết kế

3.3.1.Mối quan hệ giữa các class:

- Class Gem, BigGem, SmallGem: Class Gem là 1 abstract class được sử dụng để tạo ra 2 class BigGem và SmallGem kế thừa phương thức getPoint() của lớp Gem.
- Class Shape, Square, SemiCircle: Class Shape là 1 class được sử dụng để tạo ra 2 class Square và SemiCircle kế thừa thuộc tính position và shape của class Shape.
- Class Player, Player1, Player2, BotPlayer: class Player1, Player2, BotPlayer là các class con kế thừa từ class Player
- Class Square , Board: mối quan hệ Composition, Square là 1 phần của Board, khi Board mất đi thì Square cũng mất và khi Board sinh ra thì Square cũng được sinh ra. 1 Board có 10 Square
- Class SemiCircle, Board: mối quan hệ Composition, SemiCircle là 1 phần của Board, khi Board mất đi thì SemiCircle cũng mất và khi Board sinh ra thì SemiCircle cũng được sinh ra. 1 Board có 2 SemiCircle.
- Console và Move có quan hệ Association với Board.
- Console Association với Player.
- Board Dependency với Player.
- BotPlayer Dependency với Board và Move..

3.3.2.Chi tiết hoạt động của các phương thức

Package	Class	Method
Board	Board - board: ShapeArray[] - player1 : Player1 - player2 : Player2	+Board(): void<<constructor>> + Board(player1 : Player1, player2 : Player2) +getShape(position: int) : int ->trả về vị trí của hình + copyBoard(source : Board) : void -> copy để tạo bảng tạm cho các bước duyệt của Bot + pickAndDrop(board : Board, playingPlayer : Player, position : int, direction : int) : void -> thực hiện cho các bước bốc sỏi lên và rải sỏi. + turn(playingPlayer : Player, position : int, direction : int, isBot : boolean) : void -> thực hiện 1 lượt chơi
Console	Console -playingPlayer: Player -board: Board	+ aTurn(board: Board, playingPlayer: Player): void ->Yêu cầu người chơi chọn 1 ô trong bảng và thực hiện rải đá.Trước khi rải, kiểm tra tính hợp lệ của ô đã chọn. Sau đó, tiến hành turn để thực hiện lượt chơi. +stopTurn(curSquare: Shape, direction: int): Boolean ->Kiểm tra kết thúc lượt, nếu số sỏi trong tay người chơi là 0 và ô tiếp theo rỗng thì sẽ kết thúc + stopGame(board: Board): Boolean ->Kiểm tra kết thúc trò chơi, nếu cả hai ô SemiCircle không có viên sỏi lớn thì sẽ kết thúc + takeTurn(player1: Player, player2: Player2): void ->Chuyển lượt chơi + gameResult(player1: Player1, player2: Player2): void ->trả về kết quả trò chơi dựa trên số điểm của 2 người chơi

		+ checkNeedRefill (board: Board, playingPlayer: Player): Boolean ->Kiểm tra xem người chơi có cần nạp sỏi vào bảng, nếu tất cả ô trong phạm vi chọn rỗng thì trả về "true". + refillGem (board: Board, playingPlayer: Player): void ->nạp thêm sỏi vào tất cả các ô mà người chơi có thể chọn, sau đó giảm số điểm của người chươi đi 5 điểm
Gem	Class Gem	+getPoint(): int ->trả về số điểm của sỏi
	Class BigGem -point: int = 5	+getPoint(): int ->trả về số điểm của sỏi lớn
	Class SmallGem -point: int = 1	+getPoint(): int ->trả về số điểm của sỏi bé
Player	Class Player -totalPoint: int ->tổng điểm của người chơi -name: String ->tên người chơi - range: Integer Array[ ] ->Danh sách ô mà người chơi có thể chọn - pointinHand: int = 0 -> số sỏi trên tay	+ Player(name: String): void<<constructor>> + getTotalPoint(): int ->trả về điểm của Player + getName(): String ->trả về tên của Player + getRange(): Integer Array[] ->trả về danh sách của vị trí mà Player có thể rải đá đến + getDirection(): int ->trả về 1 nếu ngược chiều kim đồng hồ và -1 nếu cùng chiều kim đồng hồ + plusPoint(point: int): void ->cộng điểm cho player + minusPoint(point: int): void ->trừ điểm player + setDirection (direction: int): void -> + getPointinHand(): int ->trả về số điểm hiện tại của Player + setPointinHand(point: int): int ->thiết lập điểm cho Player + copyPlayer(source: Player) : Player
	Class Player1 -range:IntegerArray[] = {1,2,3,4,5} ->danh sách các vị trí người chơi2 có thể chọn	+getRange(): IntegerArray[] ->trả về danh sách người chơi 1 có thể chọn +Player1(): void<<constructor>> Phương thức khởi tạo mặc định với tên là Player1.
	Class Player2 -range:IntegerArray[] = {7,8,9,10,11} ->danh sách các vị trí người chơi2 có thể chọn	+getRange(): IntegerArray[] ->trả về danh sách người chơi 2 có thể chọn +Player2(): void<<constructor>> Phương thức khởi tạo mặc định với tên là Player2.
	Class BotPlayer -depth: int -> biểu thị độ sâu của thuật toán minimax	+ getVaildMoves(board: Board): Move -> Liệt kê tập hợp các Move có thể thực hiện. + choosenShape(board: Board): Move -> đưa ra Move tất nhất

		+ minimax(board : Board, depth : int, maximizingPlayer : boolean) : int -> thực hiện thuật toán minimax + evaluate(board : Board) : int -> phương thức đánh giá của thuật toán .
Shape	Class Shape -position: int ->biểu diễn vị trí của hình trong bảng -shape:GemArray[] ->mảng chứa các đối tượng Gem	+ Shape(position: int<<constructor>>): void ->khởi tạo một hình với vị trí được chỉ định + getPosition(): int ->trả về vị trí chính xác của hình + getPoint(): int ->tính tổng số điểm của tất cả các viên sỏi trong hình + addGem (Gem: Gem): void ->Thêm 1 viên sỏi vào danh sách + removeGem(): void ->loại bỏ tất cả các viên sỏi khỏi danh sách của hình +toString(): ->chuyển đối tượng “Shape” thành một chuỗi. +copy(): Shape tạo shape trung gian để lưu trữ lại shape
	Class Square	+toString(): ->chuyển đối tượng “Square” thành một chuỗi.
	Class SemiCircle	+toString(): ->chuyển đối tượng “SemiCircle” thành một chuỗi.
Move	- board : Board - playingPlayer : Player - position : int -> vị trí bắt đầu của bước đi - direction : int -> hướng đi	+ Move(Board board : int, int position : int, int direction : int) : void + Move(Board board : int, Player playingPlayer : int, int position : int, int direction : int) : void