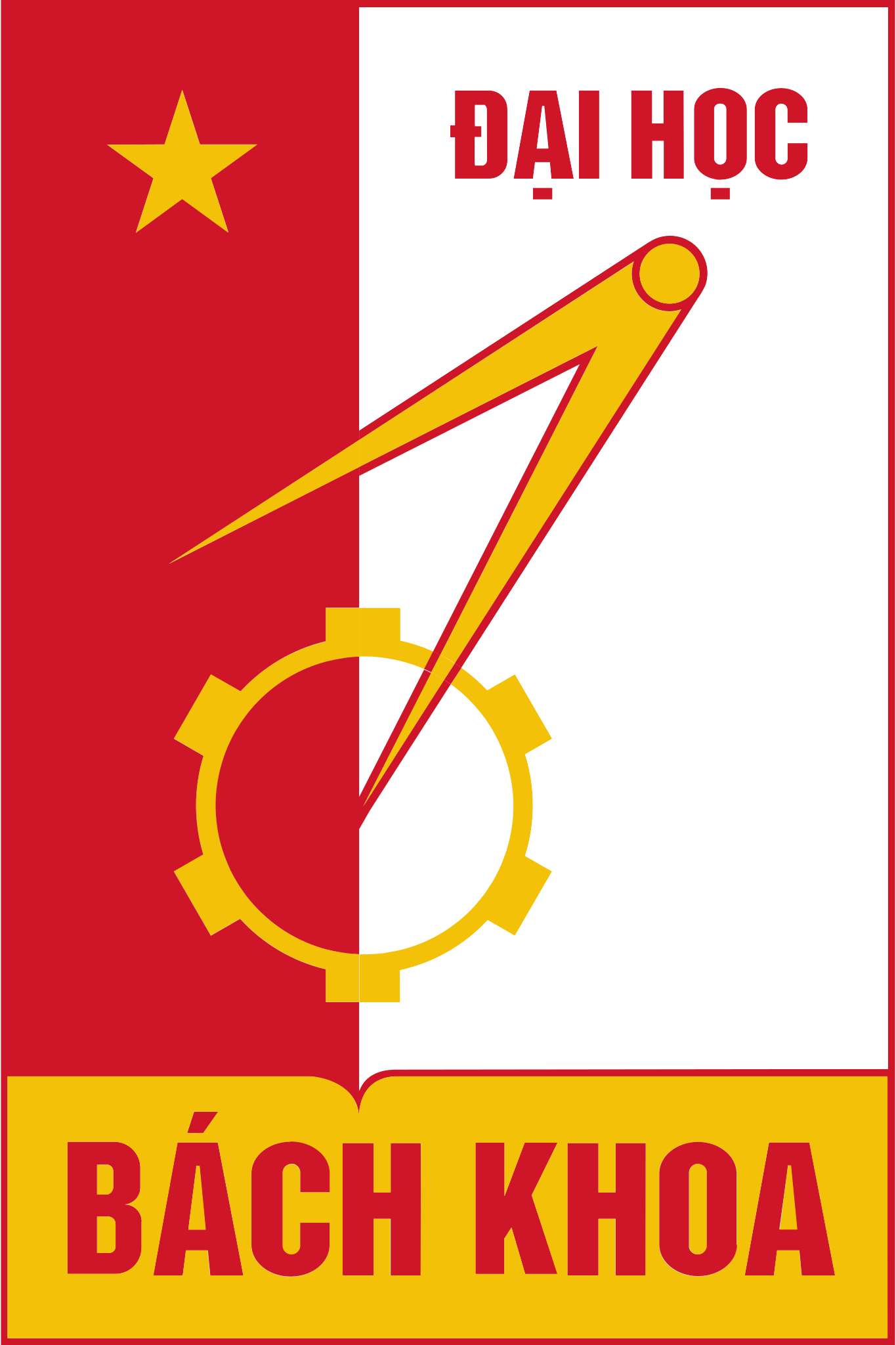
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



ĐỀ TÀI:

**XÂY DỰNG TRÒ CHƠI DÂN GIAN**

**Ô ĂN QUAN**

Bài tập môn: Lập trình hướng đối tượng

Lớp: 151963

Mã học phần: IT3103

Nhóm: 4

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Danh sách thành viên nhóm:

| **Họ và tên** | **Mã số sinh viên** | **Tỷ lệ đóng góp** |
| --- | --- | --- |
| Tô Minh Ánh | 20225787 | 19% |
| Trần Thị Vân Anh | 20225786 | 19% |
| Trần Huy Hoàng Anh | 20226076 | 19% |
| Đàm Thanh Bách (nhóm trưởng) | 20225600 | 24% |
| Mạc Thanh Bình | 20225790 | 19% |

# **MỤC LỤC**

[**MỤC LỤC 2**](#_kx2tfj9zwqc0)

[**1. Phân công nhiệm vụ 3**](#_9t7ho4ur99xj)

[**2. Mô tả dự án 4**](#_hyvjhwdoxcrk)

[2.1. Luật chơi 4](#_xol31so2rpb8)

[2.2. Yêu cầu của dự án 5](#_3mk9m2npwplq)

[**3. Thiết kế 6**](#_g3o5jycdztkh)

[2.1. Usecase Diagram 6](#_q532xn48scqr)

[2.2. Class Diagram 7](#_rgz4gnc18k14)

[2.2.1. Các lớp và vai trò : 7](#_vvsw56qlhl17)

[2.2.2. Xác định các mối quan hệ giữa các lớp: 8](#_5h4v0n9xeytb)

[2.3. Chi tiết 9](#_bp2k2cv5oat5)

[2.3.1. Cell class: Lớp trừu tượng Cell đại diện cho các ô trên bảng chơi, nơi người dùng thực hiện các tương tác trong trò chơi. Lớp này có các thuộc tính chính: 10](#_qrppdo6x3zj8)

[2.3.2. Gem Class: là lớp một lớp trừu tượng (abstract class). 11](#_kekmh567lqm7)

[2.3.3. Player Class: đại diện cho người chơi trong trò chơi, với các thuộc tính và phương thức như sau: 13](#_y2c2xwz8mcnz)

[**Tài liệu tham khảo 15**](#_wjktatowjzvg)

# **Phân công nhiệm vụ**

| **Thành viên** | **Problem**  **Design** | **Class**  **Modeling** | **GUI & khác** |
| --- | --- | --- | --- |
| Tô Minh Ánh | Nêu ý tưởng | **Square(70%)**  **Constructor**  **Pickable()** và các method khác  (trừ addGem() và removeGem()) | * Viết slide báo cáo * **GUI**   (Design hình ảnh, ShareData, EndScreen, EndScreenController (80%), EndScreenController (ngoại trừ Initialize) |
| Trần Thị Vân Anh | Nêu ý tưởng, xây dựng usecase diagram | **BigGem** (10%)  **Cell** (60%): Xử lý các method (trừ getter và setter, emptyCell(), calculateScore ())  **Player** (90%): Xử lý các method (trừ borrowGemsIfNeed())  **Game** (20%) | * Format lại project |
| Trần Huy Hoàng Anh | Nêu ý tưởng, xây dựng class diagram (70%) | **SmallGem**  **Cell** (40%): Thêm getter, setter cho Cell class, và các method như emptyCell(), calculateScore (), addGem(), removeGem(),  chỉnh sửa Constructor | * Viết slide báo cáo |
| Đàm Thanh Bách | Nêu ý tưởng, xây dựng class diagram (30%) | **Game**(80%): trừ Constructor  **BigGem**(90%)  **GameBoard**  **Player**(10%): phần borrowGemsIfNeed() | * Viết báo cáo * **GUI** (HelpScreen, HelpScreenController, IntroScreen, IntroScreenController, PlayScreen, PlayScreenController) |
| Mạc Thanh Bình | Nêu ý tưởng | **Square**(30%): addGem() & removeGem()  **HalfCircle**  **Gem**  **Pickable** | * Viết báo cáo, tìm nhạc nền cho game |

# **2. Mô tả dự án**

## **2.1. Luật chơi**

Trò chơi hai người này bao gồm một bàn cờ có 10 ô vuông, chia thành 2 hàng, và 2 nửa hình tròn ở hai đầu của bàn cờ. Ban đầu, mỗi ô vuông có 5 viên sỏi nhỏ gọi là quân dân, và mỗi nửa hình tròn có một viên sỏi to gọi là quân quan. Mỗi viên sỏi nhỏ trị giá 1 điểm, và mỗi viên sỏi to trị giá 5 điểm. Người chơi bắt đầu trò chơi sẽ được chọn ngẫu nhiên.

Mỗi người chơi sở hữu 5 ô vuông ở phía của mình và có thể bắt đầu lượt chơi từ bất kỳ ô nào trong số đó, nhặt tất cả các viên sỏi từ ô này và rải chúng theo một trong hai hướng: theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ. Người chơi phải thả một viên sỏi vào mỗi ô trên con đường của mình và tiếp tục cho đến khi không còn viên sỏi nào trong tay. Việc bắt đầu lượt chơi từ bất kỳ nửa hình tròn nào là bị cấm.

Sau khi rải hết các viên sỏi trong tay, ô vuông kế tiếp sau ô vuông cuối cùng trên con đường của người chơi được gọi là “terminal”. Nếu ô “terminal” có sỏi , người chơi có thể tiếp tục sử dụng các viên sỏi trong ô này để phân tán theo cùng một hướng. Việc rải kết thúc khi ô “terminal” không còn đá.

Sau khi hoàn thành việc phân tán, nếu ô “terminal” (trống) được theo sau bởi một ô có sỏi, người chơi có thể lấy tất cả các viên sỏi trong ô này. Nếu sau ô đã lấy đá có một ô trống nữa, tiếp theo là một ô có sỏi , người chơi có thể tiếp tục chuỗi hành động theo cách tương tự.

Người chơi bị mất lượt chơi nếu nếu ô “terminal” (trống) được theo sau bởi một ô trống nữa hoặc ô terminal là ô quan.

Trường hợp đến lượt đi nhưng cả năm ô vuông thuộc quyền kiểm soát của người chơi đều không có *sỏi* thì người đó sẽ phải dùng 5 *điểm* đã *ăn* được của mình để đặt vào mỗi ô 1 *sỏi* để có thể thực hiện việc di chuyển quân.

Trong một lượt chơi, người chơi không thể thay đổi hướng đi vào bất kỳ thời điểm nào.

Điểm số được đánh giá dựa trên số viên sỏi mà mỗi người chơi kiếm được (một viên sỏi nhỏ bằng 1 điểm, và một viên sỏi to bằng 5 điểm.

Trò chơi sẽ kết thúc khi 2 nửa hình tròn đều trống.

## **2.2. Yêu cầu của dự án**

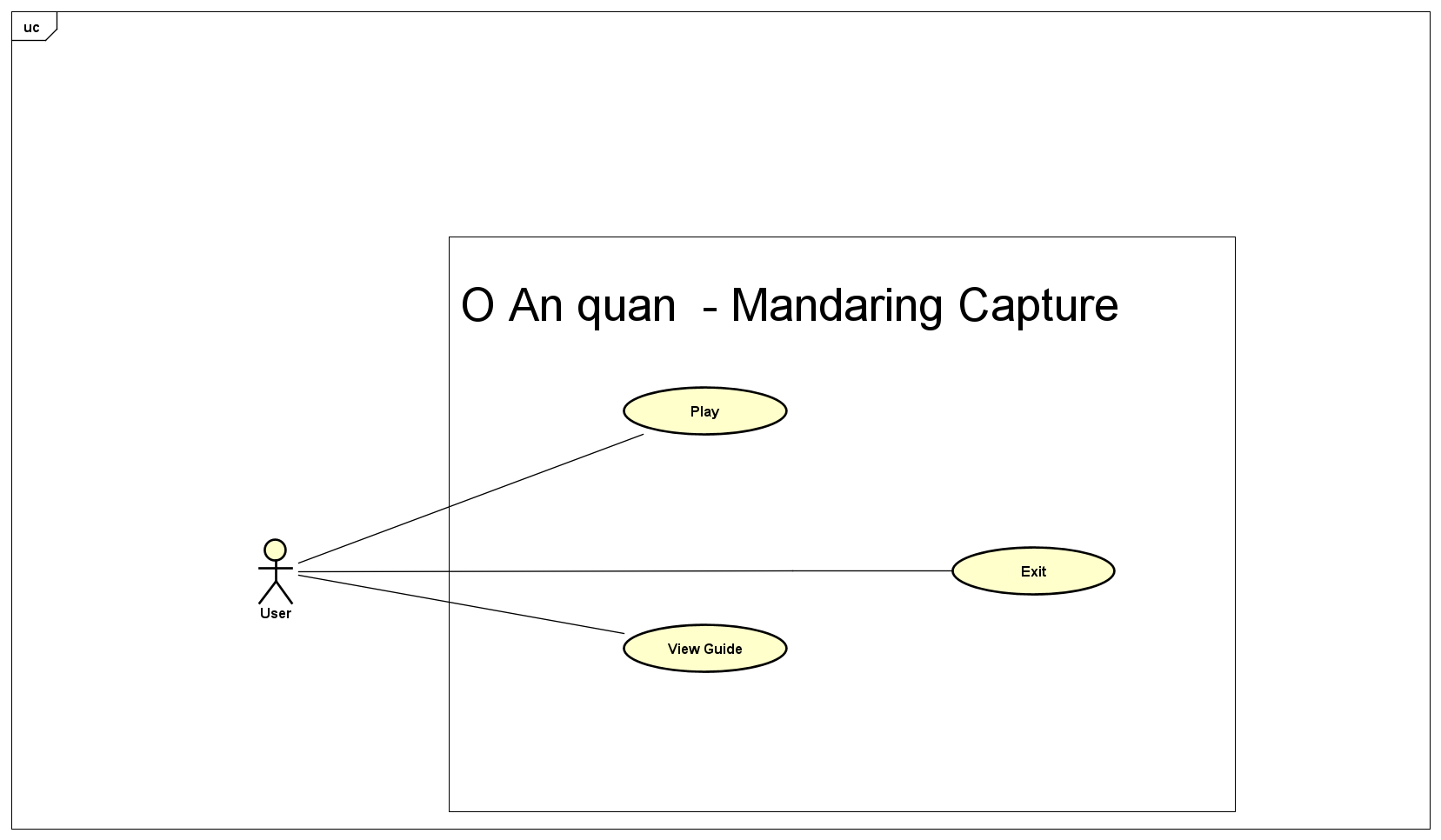
Trên màn hình chính, trò chơi sẽ có các nút sau:

* Bắt đầu: Bắt đầu trò chơi, không cần tạo các độ khó khác nhau.
* Thoát: Thoát khỏi chương trình, hãy nhớ hỏi người dùng xem họ có muốn thoát khỏi trò chơi không.
* Trợ giúp: Hiển thị hướng dẫn chơi trò chơi.

# **3. Thiết kế**

## **2.1. Usecase Diagram**

### 

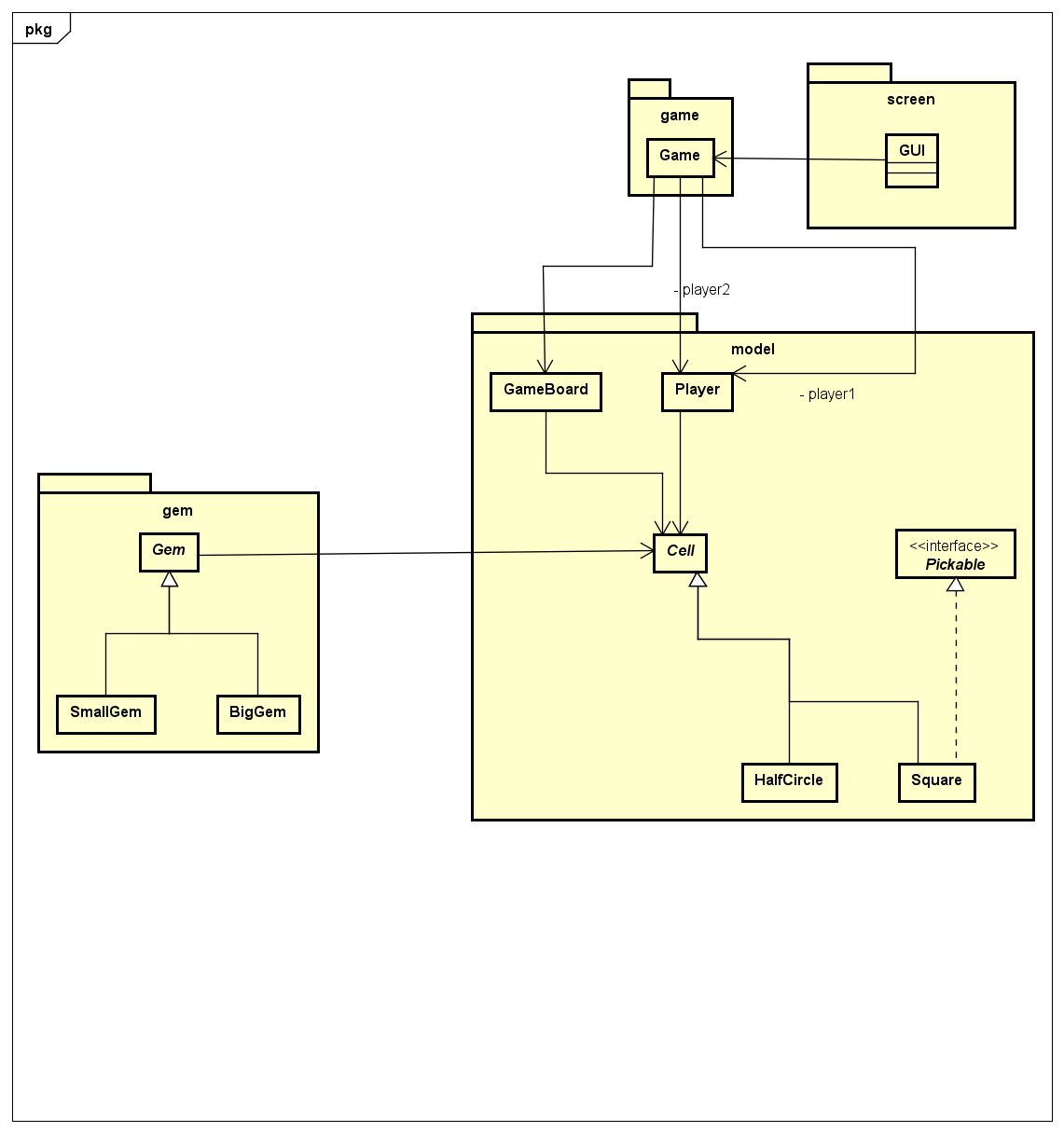


*Usecase diagram*

Người chơi có thể sử dụng qua các trường hợp sau:

* **Chơi trò chơi (Play):** Người dùng bắt đầu một ván chơi thông qua chức năng chính "Play".
* **Xem hướng dẫn (View Guide):** Người dùng có thể tìm hiểu luật chơi và cách vận hành trò chơi thông qua chức năng này.
* **Thoát khỏi hệ thống (Exit):** Người dùng có thể dừng chơi và thoát khỏi hệ thống thông qua chức năng "Exit". Hệ thống sẽ yêu cầu người dùng xác nhận có thoát chương trình không.

## **2.2. Class Diagram**

*  
Class Diagram*

### **2.2.1. Các lớp và vai trò :**

Chương trình được chia thành hai phần chính: phần "chủ động" (các thành phần khởi tạo hành động, như lớp Player) và phần "bị động" (các thành phần được tương tác, như GameBoard, Cell, và Gem).

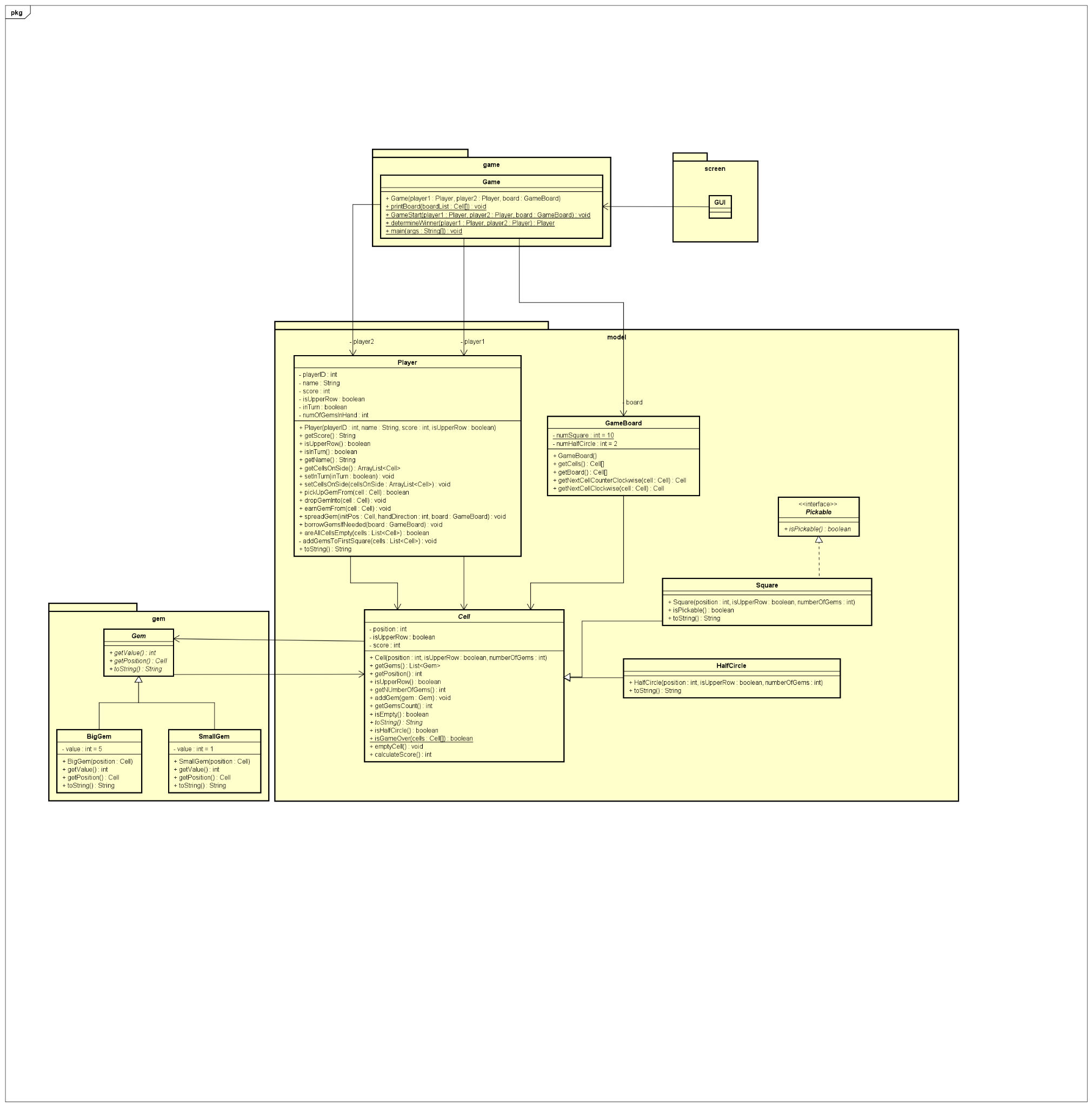
* Lớp **GameBoard** xây dựng bảng chơi với các ô **Cell**, được chia thành hai loại là **HalfCircle** (nửa vòng tròn ) và **Square** (ô vuông thông thường).
* Lớp **Gem**, đại diện cho quân cờ, được chia thành **SmallGem** là dân (1 điểm) và **BigGem** là quan (5 điểm).
* Người chơi (**Player**) thực hiện các hành động như lấy quân, rải quân, và mượn quân thông qua tương tác với các ô **Cell**.
* Lớp **Game** quản lý toàn bộ luồng trò chơi, từ khởi tạo bảng chơi, điều phối lượt chơi của người chơi, đến kiểm tra điều kiện thắng.

### **2.2.2. Xác định các mối quan hệ giữa các lớp:**

* **Quan hệ kế thừa (Inheritance):**
  + **Gem** là lớp cha của **SmallGem** và **BigGem**. Điều này thể hiện rằng các quân chơi chia sẻ chung một bộ hành vi cơ bản, nhưng có thể có những đặc điểm cụ thể riêng.
  + **Cell** là lớp cha của **HalfCircle** và **Square**.
* **Quan hệ kết hợp (Association):**
  + **Game** liên kết với **GameBoard** để quản lý bàn chơi và hai đối tượng **Player** (player1 và player2) để thực hiện logic người chơi.
  + **GameBoard** chứa danh sách các **Cell**, đại diện cho trạng thái của bàn chơi.
  + **Cell** có thể liên kết với các đối tượng **Gem**, đại diện cho quân được đặt trong ô.
* **Quan hệ phụ thuộc (Dependency):**
  + **Game** phụ thuộc vào **GUI** để hiển thị và nhận tương tác từ người chơi.
  + **Player** phụ thuộc vào **GameBoard** để thực hiện các hành động trên bàn chơi.
* **Giao diện Pickable:**
  + Được triển khai bởi **Square**, đảm bảo chỉ các ô vuông có khả năng "nhặt quân".

### 

## **2.3. Chi tiết**



*Chi tiết Class Diagram*

### **2.3.1. Cell class:** Lớp trừu tượng Cell đại diện cho các ô trên bảng chơi, nơi người dùng thực hiện các tương tác trong trò chơi. Lớp này có các thuộc tính chính:

* **position** (kiểu số nguyên): Xác định vị trí của ô trên bảng.
* **score** (kiểu số nguyên): Biểu thị số điểm hoặc số viên sỏi hiện có trong ô.
* **isUpperRow** (kiểu boolean): Xác định ô nằm ở hàng trên hay hàng dưới của bảng.
* **gems** (kiểu ArrayList): Chứa danh sách các đối tượng **Gem** hiện có trong ô.
* **Square class** và **halfCircle class**: Là hai lớp kế thừa từ lớp Cell.
* Lớp **Square** còn có các phương thức sau:
* **addGem(gem: Gem)**: Thêm một viên đá vào ô vuông.(Kế thừa từ lớp cha)
* **removeGem(gem: Gem)**: Loại bỏ một viên đá khỏi ô vuông. (Kế thừa từ lớp cha)
* **getGemsCount():** Trả về số lượng viên đá trong ô vuông. (Kế thừa từ lớp cha)
* **isEmpty()**: Kiểm tra xem ô vuông có rỗng không.(Kế thừa từ lớp cha)
* **Square class** triển khai interface **Pickable** để cho phép người dùng chọn đá từ các đối tượng ô vuông (cấm chọn đá quý từ bất kỳ bán nguyệt nào).
* **calculateScore()**: Tính toán điểm số của ô vuông dựa trên các viên đá trong nó. (Kế thừa từ lớp cha)
* **toString()**: Hiển thị ra thông tin chi tiết của Square.
* **haflCircle** cũng có các phương thức khác sau:
* **addGem(gem: Gem)**: Thêm một viên đá vào nửa vòng tròn. (Kế thừa từ lớp cha)
* **removeGem(gem: Gem)**: Loại bỏ một viên đá khỏi nửa vòng tròn.(Kế thừa từ lớp cha)
* **getGemCount()**: Trả về số lượng viên đá trong nửa vòng tròn.(Kế thừa từ lớp cha)
* **isEmpty()**: Kiểm tra xem nửa vòng tròn có rỗng không.(Kế thừa từ lớp cha)
* **calculateScore()**: Tính toán điểm số của nửa vòng tròn dựa trên các viên đá trong nó.(Kế thừa từ lớp cha)
* **toString()**: Hiển thị ra thông tin chi tiết của halfCircle.

Các tính chất của OOP được thể hiện ở lớp **Cell** là **Tính kết thừa (Inheritance), Tính trừu tượng (Abstract), Tính đa hình (Polymorphism), Tính đóng gói (Encapsulation).** Cụ thể:

* Tính kế thừa (Inheritance) được biểu hiện như sau:
* Thông qua từ khóa extends, các lớp con **halfCirlce** và **Square** được kế thừa hầu như toàn bộ các method đã được khai báo ở lớp cha. Điểm khác biệt ở đây là Square có được implement Interface **Pickable** bởi theo như luật chơi chỉ có thể nhặt sỏi lên từ các ô vuông **(Square)** chứ không phải là hình bán nguyệt **(halfCircle)**.
* Tính trừu tượng (Abstract) được thể hiện như sau:
* Ở **Cell** phương thức **toString()** là bắt buộc đối với cả **Square** và **halfCircle**, nhưng tại thời điểm khởi tạo **Cell**, ta chưa biết được rõ ràng kết quả của các phương thức này là gì vì vậy chúng ta sẽ khai báo phương thức này là phương thức trừu tượng (abstract method) và bắt buộc các lớp con **Square** và **halfCircle** phải ghi đè (Override).
* Tính đa hình (Polymorphism) được biểu hiện như sau:
* Khi khởi tạo **Cell**, chúng ta chưa biết được loại ô của chúng ta sẽ là ô vuông (**Square**) hay là hình bán nguyệt **(halfCircle)**, nên chúng ta chỉ khởi tạo phương thức **toString()** chứ chưa thực hiện chi tiết phương thức này. Và khi các lớp con **Square** hay **halfCircle** ghi đè phương thức này, ta sẽ in ra chi tiết nội dung của lớp đó.
* Tính đóng gói (Encapsulation) được biểu hiện như sau:
* Trong lớp **Cell** có những thuộc tính như **position**, **isUpperRow**, **score** có phạm vi truy cập là riêng tư (private), nghĩa là chỉ lớp **Cell** mới có thể truy cập được đến những thuộc tính này. Vì vậy, để có thể truy cập được những thuộc tính này từ bên ngoài lớp Cell, chúng ta phải có những getter, setter tương ứng.

### **2.3.2. Gem Class:** là lớp một lớp trừu tượng (abstract class).

Lớp này đại diện cho các viên đá (hoặc quân dân) trong trò chơi, được đặt tại các ô trên bảng. có mối quan hệ rất chặt chẽ với **Cell class** (Composition). Khi đối tượng **Cell** mất đi thì toàn bộ đối tượng **Gem** cũng mất đi theo, nhưng nếu gem mất đi thì đối tượng **Cell** vẫn tồn tại.

Thuộc tính

* **position** (kiểu Cell, có phạm vi truy cập protected chỉ có thể áp dụng bên trong class như thuộc tính, phương thức hay lớp con. Không thể áp dụng cho lớp ngoài hay interface.):
  + Biểu diễn vị trí hiện tại của viên đá trên bảng.
  + Viên đá được liên kết với một ô (**Cell**) thông qua thuộc tính này.

Phương thức trừu tượng

* **int getValue():**
  + Phương thức trừu tượng trả về giá trị của viên đá.
  + Các lớp con sẽ triển khai phương thức này để xác định giá trị cụ thể (ví dụ: **SmallGem** trả về 1, **BigGem** trả về 5).
* **Cell getPosition():**
  + Phương thức trừu tượng trả về vị trí của viên đá trên bảng.
* **String toString():**
  + Phương thức trừu tượng trả về chuỗi mô tả thông tin của viên đá.
  + Các lớp con sẽ triển khai để cung cấp thông tin chi tiết hơn về viên đá.

Từ đây ta có thể thấy các tính chất quan trọng của OOP đã được áp dụng trong Gem đó là **Tính kế thừa (Inheritance), Tính trừu tượng(Abstraction), Tính đa hình (Polymorphism).**

* Tính kế thừa (Inheritance) được thể hiện như sau:
* Thông qua từ khóa **extends** ở các lớp con **BigGem** và **SmallGem**, thuộc tính position có thể được truy cập bởi các lớp con vì phạm vi truy cập là protected, vì vậy các lớp con không cần phải khai báo lại thuộc tính này nữa vì đã được kế thừa từ lớp cha (**lớp Gem**).
* Tính đa hình (Polymorphis) được thể hiện như sau:
* Ở lớp **Gem**, có các phương thức **getValue()** và **toString()** được khởi tạo nhưng chưa được viết chi tiết cụ thể chức năng, mà những phương thức này sẽ được các lớp con là **SmallGem** và **BigGem** ghi đè (Override) lại.
  + Đối với **getValue()** ở **class Gem** chúng ta chưa có giá trị cụ thể cho **Gem** nhưng khi các lớp con như **SmallGem** và **BigGem** ghi đè lại phương thức này, **getValue()** sẽ trả về cho ta giá trị 1 (đối với **SmallGem**) hoặc giá trị 5 (đối với **BigGem**).
  + Đối với **toString()** ở class **Gem** chúng ta chưa có thông tin cụ thể về loại **Gem** của chúng ta, nhưng khi phương thức này được ghi đè (Override) lại ở các lớp con, **toString()** sẽ trả về giá trị khác nhau tương ứng với lớp con đó là **BigGem** hay **SmallGem**.
* Tính trừu tượng (Abstraction) được thể hiện như sau:

Ở lớp **Gem**, có các phương thức trừu tượng như **getValue()**, **getPosition()**, **toString()**. Các phương thức này là bắt buộc phải có đối với các viên sỏi ở trên bàn cờ, không phân biệt sỏi to hay sỏi nhỏ. Nhưng khi khởi tạo lớp **Gem**, chúng ta chưa biết được chính xác những giá trị mà các phương thức này sẽ trả lại, thêm vào đó, đối với loại sỏi to và sỏi nhỏ sẽ trả về các giá trị khác nhau. Vì vậy, chúng ta khai báo các phương thức này là phương thức trừu tượng (abstract method) để bắt buộc lớp con phải ghi đè (Override) lại cái phương thức này.

**SmallGem class** kế thừa từ lớp trừu tượng Gem và biểu diễn viên đá nhỏ (hoặc quân nhỏ) trong trò chơi. Mỗi viên đá nhỏ có giá trị cố định là 1 điểm.

Thuộc tính

* **value** (kiểu int, phạm vi truy cập private):
  + Giá trị của viên đá nhỏ, cố định là 1.
  + Biểu thị số điểm mà viên đá này mang lại khi được ăn hoặc sử dụng trong trò chơi.
* **position** (kế thừa từ lớp **Gem**, kiểu **Cell**):
  + Vị trí hiện tại của viên đá nhỏ trên bảng chơi.

Phương thức

* **SmallGem**(Cell position) (constructor):
  + Hàm khởi tạo nhận tham số là một đối tượng **Cell**, thiết lập vị trí (position) của viên đá nhỏ.
* **int getValue():**
  + Ghi đè (override) phương thức từ lớp **Gem**.
  + Trả về giá trị của viên đá nhỏ (luôn là 1).
* **Cell getPosition():**
  + Ghi đè phương thức từ lớp **Gem**.
  + Trả về vị trí hiện tại (position) của viên đá nhỏ.
* **String toString():**
  + Ghi đè phương thức từ lớp **Gem**.
  + Trả về chuỗi mô tả chi tiết về viên đá nhỏ, bao gồm vị trí (position) và giá trị (value).

**BigGem** kế thừa từ lớp **Gem**, tương tự như lớp **SmallGem**, đại diện cho viên ngọc lớn với giá trị cố định là 5 và vị trí được xác định bởi đối tượng **Cell**.

Lớp **BigGem** có một thuộc tính **value** có giá trị cố định là 5. Đặc biệt, trong lớp này không cần truyền giá trị **value** qua **constructor**, vì giá trị của ngọc lớn luôn là 5.

Phương thức:

* **getValue:** Phương thức này trả về giá trị của viên ngọc lớn, luôn trả về 5 (không thay đổi).
* **getPosition**: Phương thức này trả về vị trí của viên ngọc lớn trên bàn cờ. Vị trí này được thiết lập từ constructor.
* **toString**: Phương thức này trả về một chuỗi mô tả viên ngọc lớn, bao gồm thông tin về vị trí và giá trị của viên ngọc.

### **2.3.3. Player Class:** đại diện cho người chơi trong trò chơi, với các thuộc tính và phương thức như sau:

Thuộc tính:

* **playerID**: Mã số của người chơi (kiểu int).
* **name**: Tên của người chơi (kiểu String).
* **score**: Điểm của người chơi (kiểu int).
* **isUpperRow**: Xác định xem người chơi có thuộc hàng trên (kiểu boolean).
* **handPosition**: Vị trí ô hiện tại mà người chơi đang cầm (kiểu Cell).
* **inTurn**: Xác định lượt chơi của người chơi (kiểu boolean).
* **numOfGemsInHand**: Số lượng sỏi mà người chơi đang cầm (kiểu int).
* **cellsOnSide**: Danh sách các ô bên người chơi (kiểu ArrayList<Cell>).

Phương thức:

* **getScore()**: Trả về điểm của người chơi dưới dạng chuỗi.
* **isUpperRow()**: Kiểm tra xem người chơi có thuộc hàng trên không.
* **isInTurn()**: Kiểm tra xem người chơi có đang trong lượt chơi không.
* **getName()**: Trả về tên của người chơi.
* **getCellsOnSide()**: Trả về danh sách các ô bên cạnh người chơi.
* **setInTurn(boolean inTurn)**: Thiết lập giá trị cho thuộc tính inTurn.
* **setCellsOnSide(ArrayList<Cell> cellsOnSide)**: Thiết lập lại danh sách các ô bên cạnh người chơi.
* **pickUpGemFrom(Cell cell)**: Trả về boolean xem việc người chơi nhặt sỏi từ ô đã chỉ định có thoả mãn không. Quy tắc nhặt sỏi: chỉ có thể nhặt ô vuông và ô đó phải có sỏi. Nếu việc nhặt thành công cập nhập số lượng sỏi trong tay người chơi.
* **dropGemInto(Cell cell)**: Rải sỏi vào ô được chỉ định.
* **earnGemFrom(Cell cell)**: Ăn điểm từ ô đã chỉ định và trừ đi số sỏi trong ô, nếu viên sỏi đó là quan thực hiện cộng vào điểm người chơi 5 điểm, là dân thì cộng 1 điểm.
* **spreadGem(Cell initPos, int handDirection, GameBoard board)**: Rải sỏi theo lựa chọn hướng handDirection của người chơi (theo chiều kim đồng hồ hoặc ngược chiều) cho phép người chơi ăn điểm, đi tiếp , mất lượt theo luật chơi đã nêu ở trên.
* **borrowGemsIfNeeded(GameBoard board)**: Mượn sỏi nếu khi tất cả các ô vuông bên phía người chơi đều trống và điểm số của người chơi phải >= 5. Sau khi mượn thì trừ đi số điểm người chơi 5 điểm.
* **areAllCellsEmpty(List<Cell> cells)**: Kiểm tra xem tất cả các ô trong phía người chơi có trống không.
* **addGemsToFirstSquare(List<Cell> cells):** Thêm sỏi vào các ô vuông trong danh sách các ô bên phía người chơi.

Phương thức phụ:

* **toString()**: Chuyển đổi thông tin người chơi thành chuỗi để hiển thị.

Từ đây ta có thể thấy tính chất quan trọng OOP đã được áp dụng trong lớp Player là **Tính đóng gói (Encapsulation)**

* Các thuộc tính như **playerID, name, score, isUpperRow, handPosition,** và **cellsOnSide** được khai báo là private, đảm bảo chúng không thể bị truy cập trực tiếp từ bên ngoài lớp.
* Các **getter (getScore, getName, getCellsOnSide)** và **setter (setInTurn, setCellsOnSide)** cung cấp một cách an toàn để đọc và sửa đổi các thuộc tính. Điều này giúp bảo vệ dữ liệu và duy trì tính toàn vẹn của chúng.

# **Tài liệu tham khảo**

1. Ý tưởng cho khung của dự án: Design a Chess Game - GeeksforGeeks (https://www.geeksforgeeks.org/design-a-chess-game/)
2. Ý tưởng cho thiết kế giao diện: Game Ô ăn quan - Dân gian - Game Vui

(https://gamevui.vn/o-an-quan/game)

1. Luật chơi: Ô ăn quan – Wikipedia tiếng Việt

(https://vi.wikipedia.org/wiki/%C3%94\_%C4%83n\_quan)

1. Ý tưởng cho use case diagram: java - Need help for creating use case diagram of Mandarin Square Capturing - Stack Overflow

(https://stackoverflow.com/questions/67753591/need-help-for-creating-use-case-diagram-of-mandarin-square-capturing)

1. Ý tưởng xây dựng GUI cho dự án:

Querz/chess: Chess in Java using JavaFX

(https://github.com/Querz/chess)

Tất tần tật về mô hình MVC (https://viblo.asia/p/tat-tan-tat-ve-mo-hinh-mvc-Rk74avjAJeO)