

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN
MÔN PHƯƠNG PHÁP LẬP TRÌNH
HƯỚNG ĐỐI TƯỢNG



TRÒ CHƠI CỜ VUA

NHÓM: 3

LỚP: 21_2

Học kỳ 2 – Năm học 2022-2023

MỤC LỤC

I. THÔNG TIN THÀNH VIÊN	4
1. Thông tin cá nhân	4
2. Phân chia công việc	4
2.1. Trần Tiến – 21120148	4
2.2. Nguyễn Thiên An – 21120192	4
2.3. Bùi Đình Bảo – 21120201	5
2.4. Trần Gia Thịnh – 21120140	5
2.5. Mai Xuân Thắng – 21120130	5
II. XÁC ĐỊNH YÊU CẦU	6
III. MÔI TRƯỜNG LÀM VIỆC VÀ CÁC FRAMEWORK	7
IV. THIẾT KẾ	8
1. Sơ đồ UML	8
2. Cơ sở dữ liệu cho chương trình	14
V. LẬP TRÌNH	15
1. Phương pháp	15
2. Các lớp logic của chương trình	15
2.1. Lớp trừu tượng Piece	15
2.2. Lớp kế thừa King	16
2.3. Lớp kế thừa Queen	17
2.4. Lớp kế thừa Bishop	18
2.5. Lớp kế thừa Knight	18
2.6. Lớp kế thừa Rook	19
2.7. Lớp kế thừa Pawn	20
2.8. Lớp Spot	20
2.9. Lớp Board	21
2.10. Lớp Move	22
2.11. Lớp Player	24
2.12. Các lớp liên quan đến thiết kế giao diện người dùng của chương trình	24
VI. ĐÁNH GIÁ VÀ HOÀN THIỆN	31
1. Các chức năng đã làm được	31
2. Các chức năng chưa làm được	31
TÀI LIỆU THAM KHẢO	32

MỤC LỤC HÌNH ẢNH

Hình 1: Lớp thành phần chính của trò chơi	8
Hình 2: Các lớp và thuộc tính của lớp triển khai từ lớp đối tượng bàn cờ Vua.....	8
Hình 3: Thuộc tính và áp dụng Prototype Design Pattern vào lớp đối tượng Piece	9
Hình 4: Các lớp đối tượng liên quan đến logic của trò chơi	10
Hình 5: Lớp đối tượng Move dùng để cài đặt các nước đi cho trò chơi	11
Hình 6: Các lớp đối tượng phục vụ cho giao diện trò chơi.....	12
Hình 7: Sơ đồ UML của trò chơi	13

I. THÔNG TIN THÀNH VIÊN

1. Thông tin cá nhân

MSSV	Họ và tên	Email	Vai trò
21120148	Trần Tiến	21120148@student.hcmus.edu.vn	Nhóm trưởng
21120192	Nguyễn Thiên An	21120192@student.hcmus.edu.vn	Thành viên
21120201	Bùi Đình Bảo	21120201@student.hcmus.edu.vn	Thành viên
21120140	Trần Gia Thịnh	21120140@student.hcmus.edu.vn	Thành viên
21120130	Mai Xuân Thắng	21120130@student.hcmus.edu.vn	Thành viên

2. Phân chia công việc

2.1. Trần Tiến – 21120148

 **Vai trò:** Nhóm trưởng.

 **Nhiệm vụ:**

- Phân chia công việc cho nhóm, đốc thúc tiến độ làm đồ án.
- Thiết kế UML cho chương trình.
- Tìm kiếm tài nguyên và chỉnh sửa hình ảnh, âm thanh,... phục vụ cho việc lập trình giao diện.
- Lập trình giao diện cho trò chơi: lập trình lớp đối tượng Game, Lobby, PromotionDialog (cài đặt các thuộc tính, phương thức liên kết giữa logic và giao diện của trò chơi).
- Viết nội dung báo cáo cho các lớp đối tượng Game, Lobby, PromotionDialog.
- Tìm kiếm và sửa lỗi chương trình.
- Chỉnh sửa báo cáo.

2.2. Nguyễn Thiên An – 21120192


 **Vai trò:** Thành viên.

 **Nhiệm vụ:**

- Thiết kế UML cho chương trình.

- Lập trình các lớp đối tượng Piece (lớp quân cờ), Pawn (lớp quân tốt), Move (lớp quản lý di chuyển về mặt logic của trò chơi).
- Hỗ trợ tìm kiếm và sửa lỗi chương trình.
- Viết nội dung báo cáo cho các lớp đối tượng Piece, Pawn, Move.
- Chỉnh sửa báo cáo.

2.3. Bùi Đình Bảo – 21120201

 **Vai trò:** Thành viên.

 **Nhiệm vụ:**

- Lập trình các lớp đối tượng Rook (lớp quân xe), Queen (lớp quân hậu), Bishop (lớp quân tượng).
- Hỗ trợ tìm kiếm và sửa lỗi chương trình.
- Viết nội dung báo cáo cho các lớp đối tượng Rook, Queen, Bishop.
- Thiết kế UML cho chương trình.
- Vẽ lại UML và viết nội dung báo cáo cho UML.

2.4. Trần Gia Thịnh – 21120140

 **Vai trò:** Thành viên.

 **Nhiệm vụ:**

- Lập trình các lớp đối tượng Board (lớp bàn cờ về mặt logic của trò chơi), King (lớp quân vua).
- Hỗ trợ tìm kiếm và sửa lỗi chương trình.
- Viết nội dung báo cáo cho các lớp đối tượng Board, King.
- Quay và chỉnh sửa video demo.

2.5. Mai Xuân Thắng – 21120130

 **Vai trò:** Thành viên.

 **Nhiệm vụ:**

- Lập trình các lớp đối tượng Knight (quân mã), GameSaver (thực hiện việc tải và lưu trò chơi).
- Hỗ trợ tìm kiếm và sửa lỗi chương trình.
- Viết nội dung báo cáo cho các lớp đối tượng Knight, GameSaver.
- Quay và chỉnh sửa video demo.

II. XÁC ĐỊNH YÊU CẦU

Xây dựng game Cờ vua với 2 người chơi thông qua các thao tác và hiển thị đơn giản trên màn hình console. Sử dụng ngôn ngữ lập trình C++.

Tính năng cơ bản:

- Cài đặt logic trò chơi.
- Lưu và chơi lại ván cờ trước đó.
- Hiển thị bàn cờ lên màn hình.

Tính năng nâng cao:

- Thao tác chuột trên console.
- Chế độ chơi với máy (Random nước đi + Chế độ máy khó).
- Thao tác Undo/Redo đối với nước đi.
- Thao tác Replay để xem lại ván vừa đấu.
- Thiết kế giao diện người dùng.
- Âm thanh cho trò chơi.

III. MÔI TRƯỜNG LÀM VIỆC VÀ CÁC FRAMEWORK

Môi trường làm việc:

- IDE QT Creator.

Framework:

- Ứng dụng Chess sử dụng Qt (một framework hỗ trợ tạo giao diện người dùng – phiên bản được viết bằng C++).

Thư viện sử dụng:

- Các thư viện chuẩn của C++.
- QtCore: Chứa các thành phần cơ sở, bao gồm các containers, các thư viện nhập xuất, lập trình đa luồng (threading) và xử lý song song (concurrency),....
- QtWidgets: Các widget. Button, hộp thoại và những thứ tương tự trong giao diện đều gọi là các widget.
- QtMultimedia: Thư viện về âm thanh, hình ảnh, camera,....
- QtGUI: Thành phần chính để lập trình giao diện.

Công cụ hỗ trợ:

- QT Design: hỗ trợ trong việc tạo giao diện người dùng.
- Deeaker: hỗ trợ xử lý memory leak.

IV. THIẾT KẾ

1. Sơ đồ UML

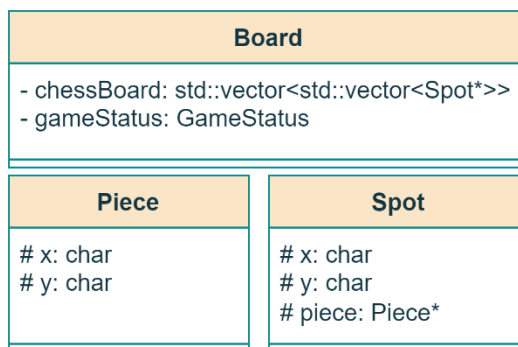
Từ yêu cầu của đồ án ta thấy có thể xây dựng ý tưởng để thiết kế các lớp đối tượng trong trò chơi cờ vua như sau:

Đầu tiên, ta xác định 2 thành phần chính có trong trò chơi là người chơi và bàn cờ vua, vì thế 2 class lớn của project lúc này là Player và Board.



Hình 1: Lớp thành phần chính của trò chơi

Đối với class Player, ta có thể định nghĩa đơn giản một thuộc tính cho lớp này chính là một biến luận lý kiểm tra xem người chơi này thuộc phe trắng hay phe đen. Đối với Board, ta thấy rằng Board chính là tập hợp của những ô cờ và quân cờ. Mỗi bàn cờ là một bảng gồm 8*8 ô cờ và dựa vào bàn cờ đầy ta hoàn toàn có thể xác định được trạng thái của bàn cờ là bị chiếu, bị chiếu bí hay là hòa cờ. Mỗi ô cờ có thể có hoặc không chứa quân cờ, mỗi quân cờ cũng sẽ chứa vị trí của ô cờ mà quân cờ đó đang đứng (khi quân cờ này chưa bị giết). Vậy có thêm 2 lớp đối tượng nữa được thêm vào là Spot và Piece, đồng thời ta có thể cập nhật các thuộc tính mới cho lớp Board bao gồm một mảng 2 chiều các spot và trạng thái của bàn cờ, lớp Spot bao gồm tọa độ x, y và quân cờ đang nắm giữ, lớp Piece bao gồm tọa độ x, y của quân cờ trên bàn cờ.



Hình 2: Các lớp và thuộc tính của lớp triển khai từ lớp đối tượng bàn cờ Vua

Xét đến quân cờ, vì một quân cờ có thể là một trong các quân (Vua, Hậu, Xe, Tượng, Mã, Tốt) nên ta sẽ xem class Piece như một lớp trừu tượng và mỗi loại quân cờ là một lớp kế thừa của lớp trừu tượng này. Lớp trừu tượng Piece cũng sẽ có thêm các thuộc tính chung cho các lớp con, bao gồm các biến kiểm tra xem quân cờ đó đã bị ăn hay chưa, đã di chuyển hay chưa, quân cờ đó là màu đen hay màu trắng.

Ngoài ra, trong quá trình triển khai lớp Piece, ta có thể tối ưu bằng cách sử dụng Prototype Design Pattern. Mẫu thiết kế này có tác dụng sao chép một đối tượng quân cờ dựa trên đối tượng có sẵn mà không cần phải copy từng giá trị của thuộc tính này sang thuộc tính khác.

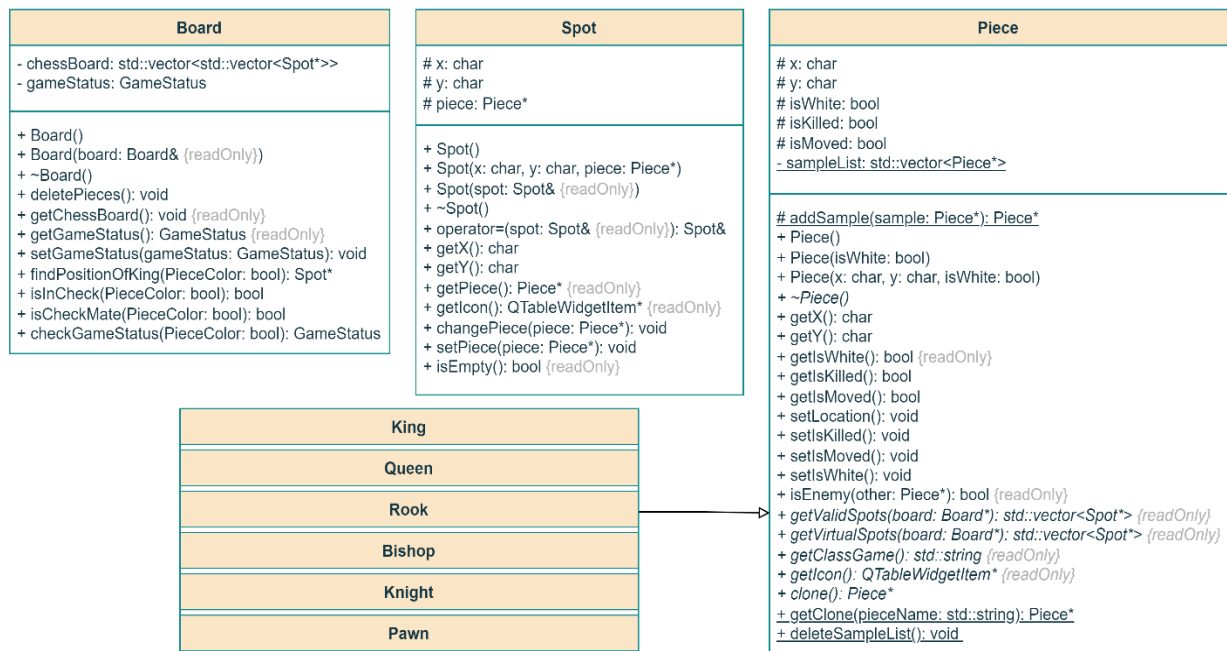
Piece
<pre># x: char # y: char # isWhite: bool # isKilled: bool # isMoved: bool - <u>sampleList: std::vector<Piece*></u></pre>
<pre># <u>addSample(sample: Piece*): Piece*</u> + <u>deleteSampleList(): void</u></pre>

Hình 3: Thuộc tính và áp dụng Prototype Design Pattern vào lớp đối tượng Piece

Sau khi xác định được các lớp và thuộc tính của lớp cần triển khai như trên, ta tiến hành cài đặt các phương thức để phục vụ cho việc thực thi chương trình. Các phương thức cơ bản bao gồm phương thức khởi tạo, phương thức hủy, phương thức getters và setters. Đặc biệt đối với mỗi class Board, Spot, Piece ta cần phải có thêm các phương thức hỗ trợ logic cho trò chơi, cụ thể:

- Lớp đối tượng Board: cần một phương thức để xác định trạng thái hiện tại của trò chơi. Phương thức này sẽ cần thêm các hàm kiểm tra khác bao gồm kiểm tra có đang bị chiếu hay không, kiểm tra có đang bị chiếu bí hay chưa để có thể kết thúc trò chơi.
- Lớp đối tượng Spot: cần một phương thức để có thể xác định và thiết đặt lại quân cờ mà ô cờ đó đang chứa. Lưu ý khi triển khai phương thức này là ta cần phải kiểm tra việc cấp phát và thu hồi vùng nhớ của quân cờ (đối với ngôn ngữ lập trình C++).

- Lớp đối tượng Piece: cần một phương thức lọc ra toàn bộ các ô cờ mà quân cờ này có thể đến được một cách hợp lệ trong trò chơi dựa trên luật chơi cờ vua quốc tế. Đồng thời cần thêm một phương thức dùng để xác định quân cờ này là quân cờ gì bằng cách xây dựng một interface lấy tên của các lớp con kế thừa lớp đối tượng Piece.



Hình 4: Các lớp đối tượng liên quan đến logic của trò chơi

Tiếp theo, ta cần phải kết nối những lớp đối tượng mà ta đã khởi tạo để tạo thành một logic hoàn chỉnh cho trò chơi. Vấn đề đó là làm sao để ta có thể tạo một giao tiếp kết nối giữa người chơi và bàn cờ lại với nhau. Giải pháp nằm ở việc thêm một lớp đối tượng mới là **Move**. Class **Move** đóng vai trò thực hiện mỗi nước đi trong trò chơi, bao gồm các thuộc tính cơ bản sau:

- (Player) **player**: nắm giữ thông tin người chơi thực hiện nước đi này.
- (Spot) **spotStart**: nắm giữ vị trí xuất phát của nước đi này.
- (Spot) **spotEnd**: nắm giữ vị trí đích của nước đi này.
- (Bool) **isCastlingMove**: kiểm tra xem nước đi này có phải là nước đi nhập thành hay không.
- (Bool) **isAbleToPromote**: kiểm tra xem nước đi này có thể là nước đi phong cấp hay không.

- (Bool) isEnPassantMove: kiểm tra xem nước đi này có phải là nước đi bắt Tốt qua đường hay không.
- (String) namePieceMove: nắm giữ tên của quân cờ thực hiện nước đi.
- (String) namePieceKilled: nắm giữ tên của quân cờ bị ăn.

Move
<ul style="list-style-type: none"> - player: Player* - spotStart: Spot* - spotEnd: Spot* - isCastlingMove: bool - isAbleToPromote: bool - isEnPassantMove: bool - namePieceMove: std::string - namePieceKilled: std::string + lastMove: Move*
<ul style="list-style-type: none"> + Move(playerTurn: Player*, spotStart: Spot*) + ~Move() + setSpotEnd(spotEnd: Spot*): void + setSpotStart(spotStart: Spot*): void + setIsCastlingMove(): void + setIsAbleToPromote(): void + setIsEnPassantMove(): void + setPieceMove(pieceMove: std::string): void + setPieceKilled(): void + getPlayer(): Player* {readOnly} + getSpotStart(): Spot* {readOnly} + getSpotEnd(): Spot* {readOnly} + getIsCastlingMove(): bool {readOnly} + getIsAbleToPromote(): bool {readOnly} + getIsEnPassantMove(): bool {readOnly} + getNamePieceMove(): std::string {readOnly} + getNamePieceKilled(): std::string {readOnly} + move(board: Board*): void + castlingMove(board: Board*): void + promote(namePiece: std::string): void + enPassant(board: Board*): void

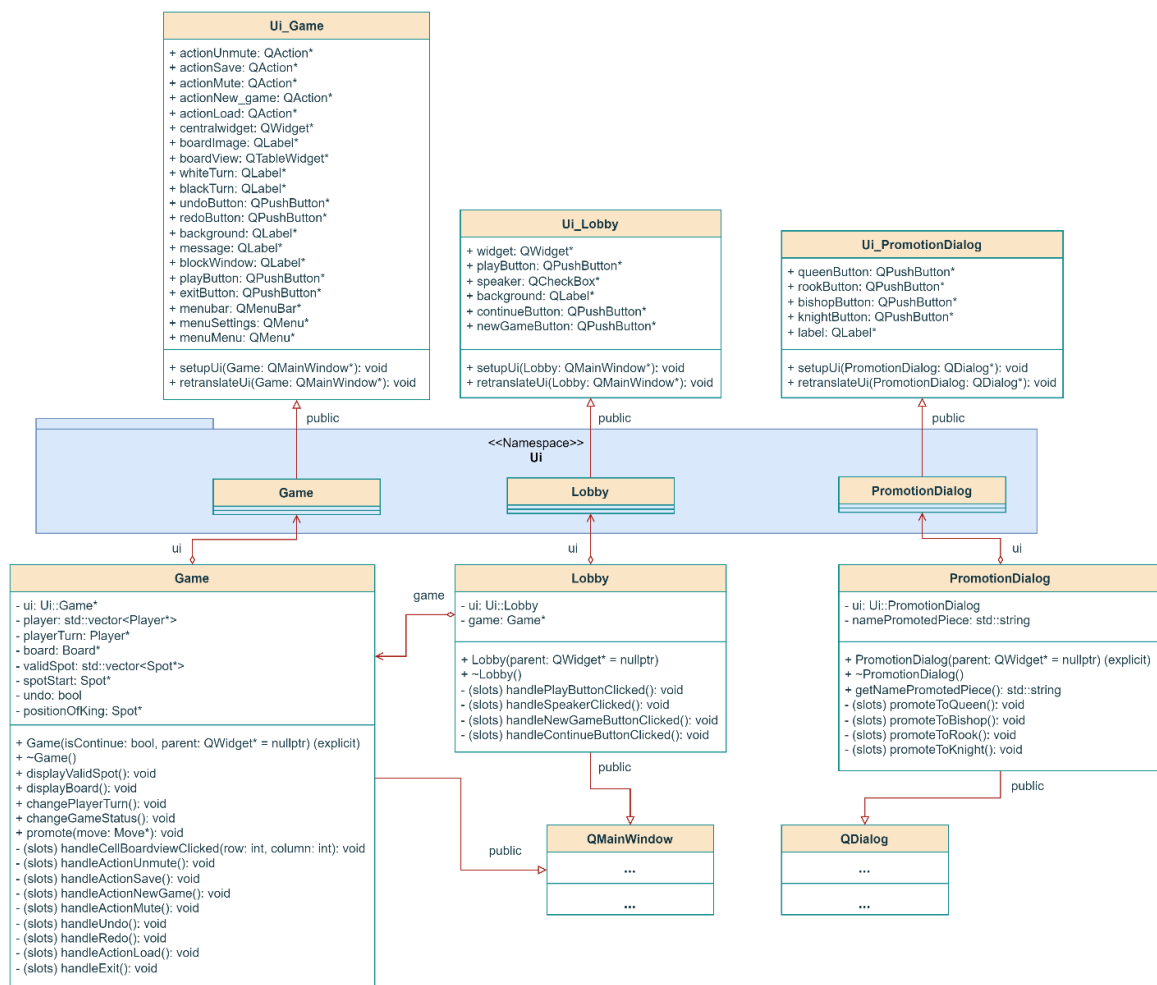
Hình 5: Lớp đối tượng Move dùng để cài đặt các nước đi cho trò chơi

Sau khi xong các bước thiết kế các lớp logic của trò chơi ta bắt đầu xây dựng các lớp quản lý giao diện gồm các lớp như sau:

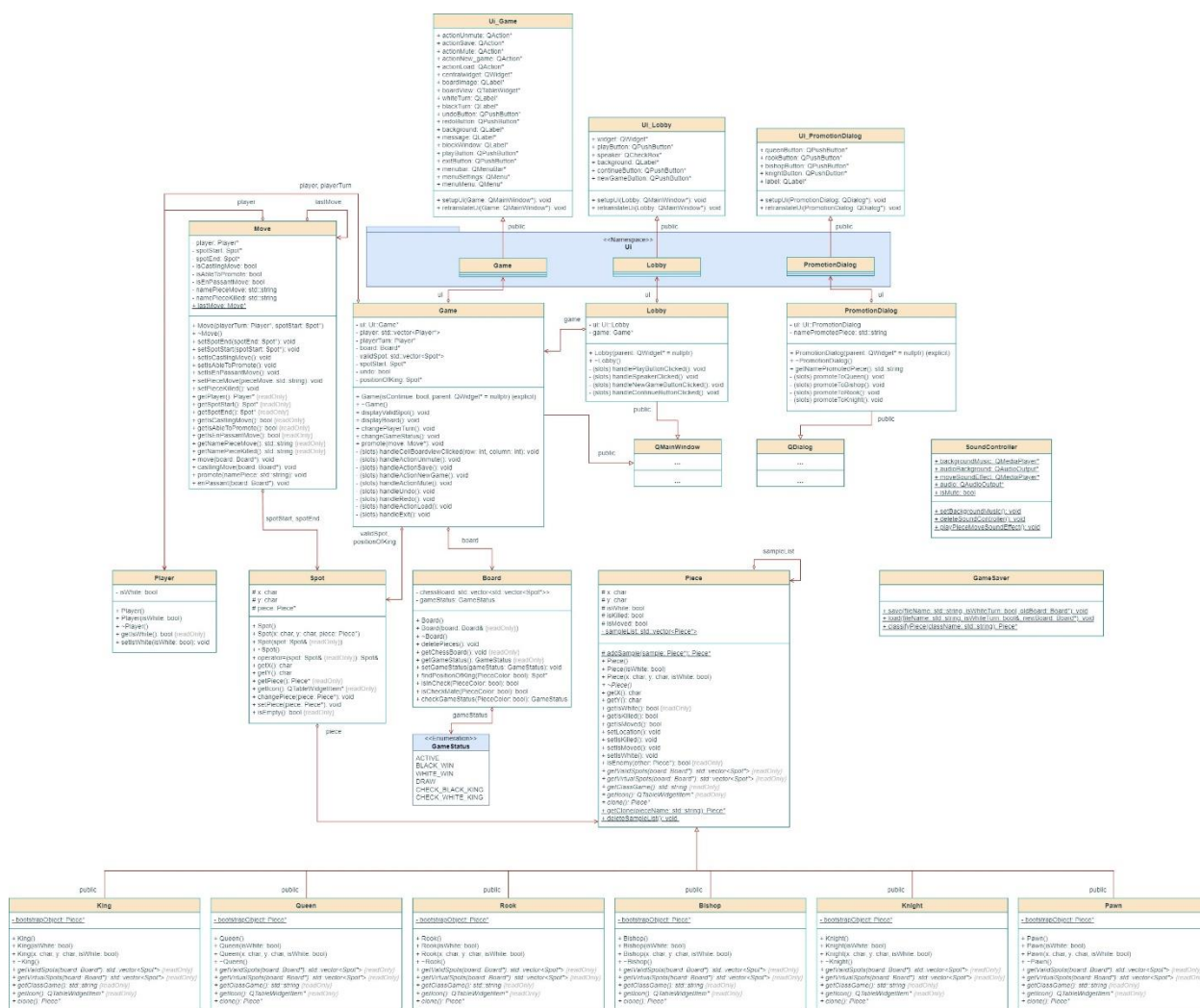
- Class UI_Game quản lý các đối tượng của trò chơi hiển thị trên màn hình. Class UI::Game kế thừa class UI_Game đặt trong namespace UI để dễ dàng cài đặt Class Game theo quy tắc lập trình của QT.
- Class Game dùng để cung cấp các xử lý cần thiết khi chơi trò chơi (thao tác với giao diện bàn cờ, quản lý hình ảnh quân cờ, quản lý âm thanh của chương trình,...).

- Class `UI_Lobby` quản lý các đối tượng hiển thị trên sảnh chờ vào trò chơi. Class `UI::Lobby` kế thừa class `UI_Lobby` đặt trong namespace `UI` để dễ dàng cài đặt Class `Lobby` theo quy tắc lập trình của QT.
- Class `Lobby` dùng để cung cấp các xử lý cần thiết ở giao diện sảnh chờ.
- Class `UI_PromotionDialog` quản lý các đối tượng của trò chơi hiển thị trong cửa sổ phong cấp tốt. Class `UI::PromotionDialog` kế thừa class `UI_PromotionDialog` đặt trong namespace `UI` để dễ dàng cài đặt Class `PromotionDialog` theo quy tắc lập trình của QT.
- Class `PromotionDialog` dùng để cung cấp các xử lý cần thiết khi phong cấp quân Tốt (thao tác với giao diện chọn loại quân cờ mà người chơi muốn phong cấp, gồm 4 quân: Hậu, Xe, Mã, Tượng).

Sơ đồ UML cho các lớp quản lý giao diện như sau:



Hình 6: Các lớp đối tượng phục vụ cho giao diện trò chơi



Hình 7: Sơ đồ UML của trò chơi

2. Cơ sở dữ liệu cho chương trình

Chương trình sử dụng ba tập tin “RecordedGame.txt”, “Undo.txt”, “Redo.txt” để lưu trữ dữ liệu.

Trạng thái của bàn cờ vua(vị trí các quân cờ, lượt đi hiện tại, nước đi cuối cùng,...) khi người dùng muốn lưu và tiếp tục trò chơi được lưu trong tập tin “RecordedGame.txt”. Dữ liệu được lưu trữ như sau:

- Dòng đầu tiên là lượt đi của người chơi nào hiện tại.
- Các dòng tiếp theo là thông tin tên quân cờ, màu, vị trí, trạng thái đã di chuyển hay chưa.
- Kết thúc tập tin là nước đi gần nhất của trò chơi để thuận tiện trong việc kiểm tra điều kiện nước đi “bắt tốt qua đường”.
- Tập tin “Undo.txt” dùng trong chức năng Undo để lưu trữ trạng thái bàn cờ trước khi đi. Cách lưu trữ tương tự như tập tin “RecordedGame.txt”.

Tập tin “Redo.txt” dùng trong chức năng Redo để lưu trữ trạng thái bàn cờ trước khi đi. Cách lưu trữ tương tự như tập tin “RecordedGame.txt”.

V. LẬP TRÌNH

1. Phương pháp

Sử dụng phương pháp hướng đối tượng và sơ đồ UML đã trình bày để tiến hành lập trình theo yêu cầu của chương trình.

2. Các lớp logic của chương trình

2.1. Lớp trừu tượng Piece

Thuộc tính riêng:

- sampleList: Thuộc tính static kiểu vector<Piece*> dùng để lưu trữ các đối tượng quân cờ. Phục vụ cho việc tạo lập một quân cờ mới dựa trên tên quân cờ có sẵn trong sampleList.

Thuộc tính chung:

- char x, char y: tọa độ của quân cờ.
- isWhite: kiểu bool để kiểm tra màu của quân cờ.
- isKilled: kiểu bool để kiểm tra quân cờ có bị ăn hay chưa.
- isMoved: kiểu bool để kiểm tra quân cờ đã di chuyển trong trận đấu hay chưa.

Phương thức chung:

- addSample(Piece* sample): kiểu Piece* để thêm đối tượng quân cờ vào thuộc tính riêng sampleList ở trên.
- getX(), getY(): kiểu char để lấy tọa độ của quân cờ.
- getIsWhite(): kiểu bool để lấy thuộc tính riêng isWhite(để biết quân cờ này là trắng hay đen).
- getIsKilled(): kiểu bool để lấy thuộc tính riêng isKilled(để biết piece này bị giết hay chưa).
- getIsMoved(): kiểu bool để lấy thuộc tính riêng isMoved ở trên (để biết quân cờ này đã di chuyển hay chưa).
- setLocation(char X, char Y): kiểu void để thiết lập tọa độ cho quân cờ này.
- setIsKilled(): kiểu void để gán biến isKilled.
- setIsMoved(): kiểu void để gán biến isMoved.
- setIsWhite(): kiểu void để gán biến isWhite.

- `ableToPromote()`: kiểu `void` để biết rằng quân cờ này (cụ thể là con tốt) có phong được lên các quân cờ khác hay không (trừ con vua).
- `isEnemy(Piece* other)`: kiểu `bool` để kiểm tra quân cờ truyền vào có phải là quân cờ địch đối với piece này không.
- `getValidSpots(Board* board) const = 0`: phương thức thuần ảo trả về kiểu `vector<Spot*>` giúp lấy vị trí các ô mà quân cờ có thể đi được.
- `getVirtualSpots(Board* board) const = 0`: phương thức thuần ảo trả về kiểu `vector<Spot*>` giúp lấy vị trí các ô mà quân cờ có thể đi được nhưng chưa kiểm tra đủ điều kiện đi của trò chơi.
- `getClassName() const = 0`: phương thức thuần ảo trả về kiểu `string` để lấy tên của các quân cờ.
- `clone() const = 0`: phương thức thuần ảo trả về kiểu `Piece*` để sao chép quân cờ có sẵn.
- `deleteSampleList()`: kiểu `static void` để xóa vùng nhớ của vector `sampleList` ở trên.

2.2. Lớp kế thừa King

Thuộc tính (Các thuộc tính riêng không kế thừa từ lớp Piece):

- Thuộc tính static `bootstrapObject`: Kiểu con trỏ `Piece` => Phục vụ cho việc phong cấp quân tốt thành quân nào khác bằng input của người dùng, sử dụng Prototype Design Pattern.

Phương thức (Các phương thức riêng và phương thức kế thừa phương thức thuần ảo của lớp Piece):

- Phương thức `getVirtualSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí hiện tại có thể đi được của quân Vua mà chưa xét đến nước đi đó có khiến quân vua bị chiếu hay không.
- Phương thức `getValidSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí

VirtualSpot mà những vị trí đó khi thực hiện nước đi với quân Vua sẽ không khiến quân vua bị chiếu.

- Phương thức `getIcon()` trả về con trỏ `QTableWidgetItem`: dùng để trả về hình ảnh hiển thị của quân Vua trên màn hình.
- Phương thức `clone()` trả về con trỏ quân cờ (Kiểu `Piece*`): dùng để tạo ra một đối tượng quân Vua mới (sử dụng Prototype Design Pattern để tối ưu việc copy thuộc tính của quân cờ bằng cách sử dụng lại đối tượng hiện có).

2.3. Lớp kế thừa Queen

Thuộc tính (Các thuộc tính riêng không kế thừa từ lớp Piece):

- Thuộc tính static `bootstrapObject`: Kiểu con trỏ `Piece` => Phục vụ cho việc phong cấp quân tốt thành quân nào khác bằng input của người dùng, sử dụng Prototype Design Pattern.

Phương thức (Các phương thức riêng và phương thức kế thừa phương thức thuần ảo của lớp Piece):

- Phương thức `getVirtualSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí hiện tại có thể đi được của quân Hậu mà chưa xét đến nước đi đó có khiến quân vua bị chiếu hay không.
- Phương thức `getValidSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí VirtualSpot mà những vị trí đó khi thực hiện nước đi với quân Hậu sẽ không khiến quân vua bị chiếu.
- Phương thức `getIcon()` trả về con trỏ `QTableWidgetItem`: dùng để trả về hình ảnh hiển thị của quân Hậu trên màn hình.
- Phương thức `clone()` trả về con trỏ quân cờ (Kiểu `Piece*`): dùng để tạo ra một đối tượng quân Hậu mới (sử dụng Prototype Design Pattern để tối ưu việc copy thuộc tính của quân cờ bằng cách sử dụng lại đối tượng hiện có).

2.4. Lớp kế thừa Bishop

Thuộc tính (Các thuộc tính riêng không kế thừa từ lớp Piece):

- Thuộc tính static bootstrapObject: Kiểu con trỏ Piece => Phục vụ cho việc phong cấp quân tốt thành quân nào khác bằng input của người dùng, sử dụng Prototype Design Pattern.

Phương thức (Các phương thức riêng và phương thức kế thừa phương thức thuần ảo của lớp Piece):

- Phương thức getVirtualSpot(Board*) tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu vector<Spot*>): dùng để lấy những vị trí hiện tại có thể đi được của quân Tượng mà chưa xét đến nước đi đó có khiến quân vua bị chiếu hay không.
- Phương thức getValidSpot(Board*) tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu vector<Spot*>): dùng để lấy những vị trí VirtualSpot mà những vị trí đó khi thực hiện nước đi với quân Tượng sẽ không khiến quân vua bị chiếu.
- Phương thức getIcon() trả về con trỏ QTableWidgetItem: dùng để trả về hình ảnh hiển thị của quân Tượng trên màn hình.
- Phương thức clone() trả về con trỏ quân cờ (Kiểu Piece*): dùng để tạo ra một đối tượng quân Tượng mới (sử dụng Prototype Design Pattern để tối ưu việc copy thuộc tính của quân cờ bằng cách sử dụng lại đối tượng hiện có).

2.5. Lớp kế thừa Knight

Thuộc tính (Các thuộc tính riêng không kế thừa từ lớp Piece):

- Thuộc tính static bootstrapObject: Kiểu con trỏ Piece => Phục vụ cho việc phong cấp quân tốt thành quân nào khác bằng input của người dùng, sử dụng Prototype Design Pattern.

Phương thức (Các phương thức riêng và phương thức kế thừa phương thức thuần ảo của lớp Piece):

- Phương thức getVirtualSpot(Board*) tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu vector<Spot*>): dùng để lấy những vị trí hiện

tại có thể đi được của quân Mã mà chưa xét đến nước đi đó có khiến quân vua bị chiếu hay không.

- Phương thức `getValidSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí `VirtualSpot` mà những vị trí đó khi thực hiện nước đi với quân Mã sẽ không khiến quân vua bị chiếu.
- Phương thức `getIcon()` trả về con trỏ `QTableWidgetItem`: dùng để trả về hình ảnh hiển thị của quân Mã trên màn hình.
- Phương thức `clone()` trả về con trỏ quân cờ (Kiểu `Piece*`): dùng để tạo ra một đối tượng quân Mã mới (sử dụng Prototype Design Pattern để tối ưu việc copy thuộc tính của quân cờ bằng cách sử dụng lại đối tượng hiện có).

2.6. Lớp kế thừa Rook

Thuộc tính (Các thuộc tính riêng không kế thừa từ lớp Piece):

- Thuộc tính static `bootstrapObject`: Kiểu con trỏ `Piece` => Phục vụ cho việc phong cấp quân tốt thành quân nào khác bằng input của người dùng, sử dụng Prototype Design Pattern.

Phương thức (Các phương thức riêng và phương thức kế thừa phương thức thuần ảo của lớp Piece):

- Phương thức `getVirtualSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí hiện tại có thể đi được của quân Xe mà chưa xét đến nước đi đó có khiến quân vua bị chiếu hay không.
- Phương thức `getValidSpot(Board*)` tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu `vector<Spot*>`): dùng để lấy những vị trí `VirtualSpot` mà những vị trí đó khi thực hiện nước đi với quân Xe sẽ không khiến quân vua bị chiếu.
- Phương thức `getIcon()` trả về con trỏ `QTableWidgetItem`: dùng để trả về hình ảnh hiển thị của quân Xe trên màn hình.

- Phương thức clone() trả về con trỏ quân cờ (Kiểu Piece*): dùng để tạo ra một đối tượng quân Xe mới (sử dụng Prototype Design Pattern để tối ưu việc copy thuộc tính của quân cờ bằng cách sử dụng lại đối tượng hiện có).

2.7. Lớp kế thừa Pawn

Thuộc tính riêng:

- bootStrapObject: Thuộc tính static, kiểu Piece*. Phục vụ cho việc phong cấp quân tốt thành quân nào khác bằng input của người dùng.

Phương thức riêng:

- Phương thức getVirtualSpot(Board*) tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu vector<Spot*>): dùng để lấy những vị trí hiện tại có thể đi được của quân Tốt mà chưa xét đến nước đi đó có khiến quân vua bị chiếu hay không.
- Phương thức getValidSpot(Board*) tham số đầu vào là một con trỏ bàn cờ, trả về các con trỏ ô cờ (Kiểu vector<Spot*>): dùng để lấy những vị trí VirtualSpot mà những vị trí đó khi thực hiện nước đi với quân Tốt sẽ không khiến quân vua bị chiếu.
- Phương thức getIcon() trả về con trỏ QTableWidgetItem: dùng để trả về hình ảnh hiển thị của quân Tốt trên màn hình.
- Phương thức clone() trả về con trỏ quân cờ (Kiểu Piece*): dùng để tạo ra một đối tượng quân Tốt mới (sử dụng Prototype Design Pattern để tối ưu việc copy thuộc tính của quân cờ bằng cách sử dụng lại đối tượng hiện có).

2.8. Lớp Spot

Thuộc tính:

- char x, char y: tọa độ của ô trên bàn cờ.
- piece: kiểu Piece* để biết được ô có tọa độ x, y trên bàn cờ là quân cờ nào.

Phương thức:

- getX(), getY(): kiểu char để lấy tọa độ x, y của spot.
- getPiece(): kiểu Piece* để biết ô cờ này đang giữ quân cờ nào.

- `changePiece(Piece* piece)`: kiểu `void` để thay đổi ô cờ này từ quân cờ này sang quân cờ khác.
- `setPiece(Piece* piece)`: kiểu `void` để gán cho ô cờ này chiếm giữ quân cờ `piece`.
- `isEmpty()`: kiểu `bool` để biết rằng ô cờ này có đang trống không (có đang chiếm giữ quân cờ nào không).

2.9. Lớp Board

Thuộc tính riêng:

- `chessBoard`: kiểu `vector<vector<Spot*>>` lưu trữ thông tin của 64 ô trên bàn cờ.
- `gameStatus`: kiểu `GameStatus`, dạng `enum` để biểu thị trạng thái của bàn cờ:

ACTIVE: Trạng thái bình thường.

BLACK_WIN: Quân đen thắng.

WHITE_WIN: Quân trắng thắng.

DRAW: Hòa.

CHECK_BLACK_KING: Vua đen bị chiếu.

CHECK_WHITE_KING: Vua trắng bị chiếu.

Thuộc tính chung:

- `getChessBoard()`: kiểu `vector<vector<Spot*>>` để lấy bàn cờ `chessBoard` ở trên.
- `getGameStatus()`: kiểu `GameStatus` để lấy thuộc tính `gameStatus` ở trên.
- `SetGameStatus(GameStatus gamestatus)`: kiểu `void` để thiết lập trạng thái của bàn cờ.
- `IsInCheck(bool PieceColor)`: kiểu `bool` để kiểm tra xem vua có màu `pieceColor` có đang bị chiếu hay không.

2.10. Lớp Move

Thuộc tính:

- player: kiểu Player* dùng để chứa con trỏ trỏ đến đối tượng của người chơi đang thực hiện nước đi này.
- spotStart: kiểu Spot* dùng để chứa con trỏ trỏ đến ô ban đầu của quân cờ sắp đi.
- spotEnd: kiểu Spot* dùng để chứa con trỏ trỏ đến ô quân cờ đi đến.
- isCastlingMove: kiểu bool dùng để xác định xem nước đi có phải là nước nhập thành không.
- isAbleToPromote: kiểu bool dùng để xác định xem sau khi thực hiện nước đi này người chơi đã đủ điều kiện nhập thành chưa.
- isEnPassantMove: kiểu bool dùng để xác định xem nước đi đó có phải là nước bắt tốt qua đường không.
- namePieceMove: kiểu string dùng để lưu lại tên quân cờ vừa được đi.
- namePieceKilled: kiểu string dùng để lưu lại tên quân cờ của đối phương mà người chơi vừa ăn.

Phương thức:

- lastMove: kiểu static Move* dùng để lưu lại thông tin nước đi liền trước, phục vụ cho việc lưu và tải lại game cũ.
- Move(Player* playerTurn, Spot* spotStart): phương thức khởi tạo với hai tham số là con trỏ trỏ đến người dùng đang thực hiện nước đi này, và con trỏ trỏ đến ô bắt đầu trên bàn cờ. Từ 2 tham số này sẽ gán giá trị cho các thuộc tính player, sportStart, namePieceMove.
- ~Move(): Phương thức hủy, giải phóng vùng nhớ được cấp phát.
- setSpotEnd(Spot* spotEnd): kiểu void, dùng để gán giá trị cho thuộc tính spotEnd thông qua tham số truyền vào. Tham số này đã được kiểm tra đầy đủ các điều kiện hợp lệ.

- `setSpotStart(Spot* spotStart)`: kiểu `void`, dùng để gán giá trị cho thuộc tính `spotStart` thông qua tham số truyền.
- `setIsCastlingMove()`: kiểu `void`, dùng để kiểm tra điều kiện và gán giá trị cho thuộc tính `isCastlingMove`.
- `setIsAbleToPromote()`: kiểu `void`, dùng để kiểm tra điều kiện và gán giá trị cho thuộc tính `isAbleToPromote`.
- `setIsEnPassantMove()`: kiểu `void`, dùng để kiểm tra điều kiện và gán giá trị cho thuộc tính `isEnPassantMove`.
- `setPieceMove(string pieceMove)`: kiểu `void`, dùng để gán giá trị cho thuộc tính `namePieceMove`.
- `setPieceKilled()`: kiểu `void`, dùng để gán giá trị cho thuộc tính `namePieceKilled` được lấy từ thuộc tính `spotEnd`.
- `getPlayer() const`: trả về kiểu `Player*` dùng để lấy ra thuộc tính `player`.
- `getSpotStart() const`: trả về kiểu `Spot*` dùng để lấy ra thuộc tính `spotStart`.
- `getSpotEnd() const`: trả về kiểu `Spot*` dùng để lấy ra thuộc tính `spotEnd`.
- `getIsCastlingMove() const`: trả về kiểu `bool` dùng để lấy ra thuộc tính `isCastlingMove`.
- `getIsAbleToPromote() const`: trả về kiểu `bool` dùng để lấy ra thuộc tính `isAbleToPromote`.
- `getIsEnPassantMove() const`: trả về kiểu `bool` dùng để lấy ra thuộc tính `isEnPassantMove`.
- `getNamePieceMove() const`: trả về kiểu `string` dùng để lấy ra thuộc tính `namePieceMove`.
- `getNamePieceKilled() const`: trả về kiểu `string` dùng để lấy ra thuộc tính `namePieceKilled`.
- `move(Board* board)`: kiểu `void`, dùng để gọi các hàm kiểm tra các nước đi đặc biệt, di chuyển piece từ `spotStart` đến `spotEnd`, xóa vùng nhớ `spotEnd` nếu có, và thực hiện các yêu cầu của các nước đi đặc biệt nếu có.
- `castlingMove(Board* board)`: kiểu `void`, dùng để di chuyển quân Xe sang vị trí thích hợp trong nước đi nhập thành.

- `promote(std::string namePiece)`: kiểu void, dùng để tạo quân cờ mới có tên trùng với tham số truyền vào và thay thế quân Tốt được phong bằng quân cờ vừa tạo.
- `enPassant(Board* board)`: kiểu void, dùng để xóa con Tốt của đối phương bị bắt khi đang qua đường.

2.11. Lớp Player


Thuộc tính:

- `isWhite`: kiểu bool, dùng để xác định màu của người chơi.

Phương thức:

- `Player()`: phương thức tạo lập mặc định, người chơi luôn có màu trắng.
- `Player(bool isWhite)`: phương thức tạo lập, gán giá trị cho biến `isWhite` thông qua giá trị của tham số truyền vào.
- `~Player()`: Phương thức hủy, giải phóng vùng nhớ được cấp phát.
- `getIsWhite()`: là hàm const, kiểu bool dùng để lấy thuộc tính `isWhite`.
- `setIsWhite(bool isWhite)`: không có kiểu trả về, dùng để gán giá trị thuộc tính `isWhite` bằng giá trị của tham số truyền vào.

2.12. Các lớp liên quan đến thiết kế giao diện người dùng của chương trình

 **Lớp UI_Game**: sử dụng các đối tượng trong thư viện thiết kế giao diện của QT để quản lý các đối tượng trong cửa sổ chính của trò chơi.

• *Thuộc tính:*

- `QAction *actionUnmute`: nút mở âm thanh trong thanh menu.
- `QAction *actionSave`: nút lưu game trong thanh menu.
- `QAction *actionMute`: nút tắt âm trong thanh menu.
- `QAction *actionNew_game`: nút tạo trò chơi mới trong thanh menu.
- `QAction *actionLoad`: nút tải trò chơi đã lưu trong thanh menu.
- `QWidget *centralwidget`: khung giao diện chính.
- `QLabel *boardImage`: hình ảnh bàn cờ.
- `QTableWidget *boardView`: bảng nắm giữ các hình ảnh quân cờ.

- QLabel *whiteTurn: hình ảnh tới lượt quân trắng.
- QLabel *blackTurn: hình ảnh tới lượt quân đen.
- QPushButton *undoButton: nút undo.
- QPushButton *redoButton: nút redo.
- QLabel *background: hình nền của trò chơi.
- QLabel *message: thông báo khi trò chơi kết thúc.
- QLabel *blockWindow: hình ảnh hiệu ứng làm mờ trò chơi khi trò chơi kết thúc.
- QPushButton *playButton: nút play.
- QPushButton *exitButton: nút thoát thông báo kết thúc trò chơi.
- QMenuBar *menubar: thanh menu.
- QMenu *menuSettings: mục cài đặt trong thanh menu.
- QMenu *menuMenu: mục menu trong thanh menu.

- **Phương thức:**

- void setupUi(QMainWindow *Game): phương thức trả về kiểu void, hiển thị các thuộc tính của lớp UI_Game lên màn hình.
- void retranslateUi(QMainWindow *Game): phương thức trả về kiểu void, hỗ trợ phương thức setupUI hiển thị các đối tượng lên màn hình.

🚦 **Lớp Game thuộc namespace UI:** kế thừa lớp UI_Game và đặt trong namespace UI.

🚦 **Lớp Game:** cài đặt các thuộc tính, phương thức liên kết giữa logic và giao diện của trò chơi.

- **Thuộc tính:**


- Ui::Game *ui: thuộc tính ui là kiểu Game thuộc namespace UI đã cài đặt ở trên, chứa các đối tượng hiển thị giao diện của trò chơi.
- std::vector<Player*> player: thuộc tính nắm giữ hai người chơi bên đen và trắng.
- Player* playerTurn: thuộc tính nắm giữ người chơi hiện tại đang đi.
- Board* board: bàn cờ logic của trò chơi.
- std::vector<Spot*> validSpot: các nước đi hợp lệ trên bàn cờ logic.

- Spot* spotStart: ô cờ được người chơi chọn trên bàn cờ để đi.
- bool undo: thuộc tính kiểm tra có người chơi được undo, redo.
- Spot* positionOfKing: vị trí của quân vua của người chơi đối phương để hiển thị vua có bị chiếu hay không.

- **Phương thức:**

- Game(bool isContinue, QWidget *parent = nullptr): phương thức khởi tạo lớp Game với tham số truyền vào đầu tiên là biến bool isContinue kiểm tra xem có phải trò chơi mới hay tiếp tục chơi ván cờ cũ.
- void displayValidSpot(): phương thức trả về kiểu void hiển thị các nước cờ có thể đi được.
- void displayBoard(): phương thức trả về kiểu void hiển thị bàn cờ.
- void changePlayerTurn(): phương thức trả về kiểu void hiển thị lượt đi của người chơi.
- void changeGameStatus(): phương thức trả về kiểu void thay đổi giao diện trạng thái của trò chơi.
- void promote(Move* move): phương thức trả về kiểu void hiển thị cửa sổ cho người chơi phong tót.
- ~Game(): Phương thức hủy của lớp Game.
- void handleCellBoardviewClicked(int row, int column): phương thức trả về kiểu void xử lý hành động nhấn chuột vào một ô trên bàn cờ.
- void handleActionUnmute(): phương thức trả về kiểu void xử lý hành động nhấn vào nút mở âm.
- void handleActionSave(): phương thức trả về kiểu void xử lý hành động nhấn nút lưu trò chơi.
- void handleActionNewGame(): phương thức trả về kiểu void xử lý hành động nhấn nút bắt đầu trò chơi mới.
- void handleActionMute(): phương thức trả về kiểu void xử lý hành động nhấn nút tắt âm.
- void handleUndo(): phương thức trả về kiểu void xử lý hành động nhấn nút undo.

- void handleRedo(): phương thức trả về kiểu void xử lý hành động nhấn nút redo.
- void handleActionLoad(): phương thức trả về kiểu void xử lý hành động nhấn nút chơi lại trò chơi đã lưu.
- void handleExit(): phương thức trả về kiểu void xử lý hành động nhấn nút thoát.


 **Lớp UI_Lobby:** sử dụng các đối tượng trong thư viện thiết kế giao diện của QT để quản lý các đối tượng trong cửa sổ sảnh chờ của trò chơi.


- **Thuộc tính:**

- QWidget *widget: khung giao diện chính của sảnh chờ trò chơi.
- QPushButton *playButton: nút bắt đầu trò chơi.
- QCheckBox *speaker: nút bật tắt âm.
- QLabel *background: hình nền của trò chơi.
- QPushButton *continueButton: nút tiếp tục trò chơi.
- QPushButton *newGameButton: nút bắt đầu trò chơi mới.

- **Phương thức:**

- void setupUi(QMainWindow *Lobby): phương thức trả về kiểu void, hiển thị các thuộc tính của lớp UI_Lobby lên màn hình.
- void retranslateUi(QMainWindow *Lobby): phương thức trả về kiểu void, hỗ trợ phương thức setupUI hiển thị các đối tượng lên màn hình.

 **Lớp Lobby thuộc namespace UI:** kế thừa lớp UI_Lobby và đặt trong namespace UI.

 **Lớp Lobby:** cài đặt các thuộc tính, phương thức hiển thị giao diện sảnh chờ của trò chơi.

- **Thuộc tính:**

- Ui::Lobby *ui: thuộc tính ui là kiểu UI::Lobby thuộc namespace UI đã cài đặt ở trên, chứa các đối tượng hiển thị giao diện sảnh chờ của trò chơi.
- Game* game: thuộc tính kiểu Game* quản lý logic và giao diện trò chơi.

- **Phương thức:**

- Lobby(QWidget *parent = nullptr): phương thức khởi tạo lớp Lobby.

- ~Lobby(): phương thức huỷ lớp Lobby.
- void handlePlayButtonClicked(): phương thức trả về kiểu void, xử lý hành động nhấn vào nút Play.
- void handleSpeakerClicked(): phương thức trả về kiểu void, xử lý hành động nhấn vào nút tắt bật âm.
- void handleNewGameButtonClicked(): phương thức trả về kiểu void, xử lý hành động nhấn vào nút bắt đầu trò chơi mới.
- void handleContinueButtonClicked(): phương thức trả về kiểu void, xử lý hành động nhấn vào nút tiếp tục trò chơi cũ.

🚩 **Lớp Ui_PromotionDialog** sử dụng các đối tượng trong thư viện thiết kế giao diện của QT để quản lý các đối tượng trong cửa sổ phong tốt.

- **Thuộc tính:**

- QPushButton *queenButton: nút bấm phong tốt thành quân hậu.
- QPushButton *rookButton: nút bấm phong tốt thành quân xe.
- QPushButton *bishopButton: nút bấm phong tốt thành quân tượng.
- QPushButton *knightButton: nút bấm phong tốt thành quân mã.
- QLabel *label: khung giao diện của cửa sổ phong tốt.

- **Phương thức:**

- void setupUi(QMainWindow * PromotionDialog): phương thức trả về kiểu void, hiển thị các thuộc tính của lớp UI_PromotionDialog lên màn hình.
- void retranslateUi(QMainWindow * PromotionDialog): phương thức trả về kiểu void, hỗ trợ phương thức setupUI hiển thị các đối tượng lên màn hình.

🚩 **Lớp PromotionDialog thuộc namespace UI:** kế thừa lớp UI_PromotionDialog và đặt trong namespace UI.


🚩 **Lớp PromotionDialog: cài đặt các thuộc tính, phương thức hiển thị giao diện cửa sổ phong tốt.**

- **Thuộc tính:**

- `Ui::PromotionDialog *ui:` thuộc tính `ui` là kiểu `Ui::PromotionDialog` thuộc namespace `UI` đã cài đặt ở trên, chứa các đối tượng hiển thị giao diện trong cửa sổ phong tốt.
- `std::string namePromotedPiece:` thuộc tính nắm giữ tên của quân cờ được phong thành.

- **Phương thức:**

- `PromotionDialog(QWidget *parent = nullptr):` phương thức khởi tạo.
- `std::string getNamePromotedPiece():` phương thức trả về kiểu `std::string` lấy thuộc tính `std::string namePromotedPiece`.
- `~PromotionDialog():` phương thức hủy.
- `void promoteToQueen():` phương thức trả về kiểu `void`, xử lý hành động nhấn vào nút phong hậu.
- `void promoteToBishop():` phương thức trả về kiểu `void`, xử lý hành động nhấn nút phong tượng.
- `void promoteToRook():` phương thức trả về kiểu `void`, xử lý hành động nhấn phong xe.
- `void promoteToKnight():` phương thức trả về kiểu `void`, xử lý hành động nhấn nút phong mã.

 **Lớp SoundController:** quản lý các đối tượng phát âm thanh của trò chơi.

- **Thuộc tính:**

- `static bool isMute:` thuộc tính giữ trạng thái bật tắt âm.
- `static QMediaPlayer* backgroundMusic:` thuộc tính nắm giữ âm thanh nền của trò chơi.
- `static QAudioOutput* audioBackground:` thuộc tính nắm giữ thiết bị phát âm thanh nền.
- `static QMediaPlayer* moveSoundEffect:` thuộc tính nắm giữ âm thanh bước đi của quân cờ.
- `static QAudioOutput* audio:` thuộc tính nắm giữ thiết bị phát âm thanh quân cờ.

- **Phương thức:**

- static void setBackgroundMusic(): phương thức tĩnh trả về kiểu void thiết lập âm thanh nền cho trò chơi.
- static void deleteSoundController(): phương thức tĩnh trả về kiểu void huỷ các thuộc tính tĩnh của lớp SoundController.
- static void playPieceMoveSoundEffect(): phương thức tĩnh trả về kiểu void phát âm thanh bước đi của quân cờ.

VI. ĐÁNH GIÁ VÀ HOÀN THIỆN

1. Các chức năng đã làm được

Tính năng cơ bản:

- Cài đặt logic trò chơi
- Lưu và chơi lại ván cờ trước đó
- Hiện thị bàn cờ lên màn hình

Tính năng nâng cao:

- Thao tác chuột trên console
- Thao tác Undo/Redo đối với nước đi
- Thiết kế giao diện người dùng
- Âm thanh cho trò chơi

2. Các chức năng chưa làm được

- Xem lại trận đấu vừa chơi
- Đấu với máy.

TÀI LIỆU THAM KHẢO

- [1] "Qt Documentation," [Online]. Available: <https://doc.qt.io/>.
- [2] "Qt Multimedia," [Online]. Available: <https://doc.qt.io/qt-6/qtmultimedia-index.html>.
- [3] "Qt Widgets C++ Classes," [Online]. Available: <https://doc.qt.io/qt-6/qtwidgets-module.html>.
- [4] shashipk11, "Design a Chess Game," [Online]. Available: <https://www.geeksforgeeks.org/design-a-chess-game/>.
- [5] "Object Oriented Design for a Chess game," [Online]. Available: <https://stackoverflow.com/questions/4168002/object-oriented-design-for-a-chess-game>.
- [6] "Design a chess game using object-oriented principles," [Online]. Available: <https://codereview.stackexchange.com/questions/71790/design-a-chess-game-using-object-oriented-principles>.
- [7] "Deploying Qt Applications," [Online]. Available: <https://doc.qt.io/qt-6/deployment.html>.

-----HẾT-----