Đại học Bách Khoa Hà Nội
Trường Công nghệ thông tin và truyền thông

BÁO CÁO MINI PROJECT

Traditional game: Cờ gánh

lập trình hướng đối tượng

Link git repository

Nhóm 36
Nguyễn Xuân Tùng - 20215162
Nguyễn Chiêu Văn - 20215163
tung.nx215162@sis.hust.edu.vn
van.nc215163@sis.hust.edu.vn

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Hà Nội, 06/01/2024

TABLE OF CONTENTS

1 Assignment of menbers

1.1 Nguyễn Xuân Tùng

- Thiết kế UML (*usecase diagram, class diagram*)
- Thiết kế UI (javafx)
- viết mã các chức năng giao tiếp giữa UI và user
- Quản lí kết nối giữa các model và ứng dụng
- Thiết kế chức năng cơ bản cho bàn cờ
- Viết báo cáo

Các file và hàm cụ thể:

- Piece.java
- Player.java
- PlaySide.java
- CoGanhApp.java
- GameController.java
  - initialize()
  - ClickSurrender()
  - menuButtonEnterEffect()
  - menuButtonLeaveEffect()
  - undoEnterEffect()
  - undoLeaveEffet()
  - mouseEnterEffect()
  - mouseLeaveEffect()
  - clickHelpButton()
  - clickExitButton()
  - enterPlayerName()
  - makeMove()

  - glowEffect()
  - reset()
- ExitDialog.java
- HelpController.java
- HelpPopUp.java
- surrenderPopUp.java
- gameUI.fxml
- helpUI.fxml

1.2 Nguyễn Chiêu Văn

- Đề xuất ý kiến và tham gia vào quá trình xây dựng UML
- Nghiên cứu về cấu trúc và quy tắc cho cờ gánh
- Viết mã lập trình xử lý các tình huống và luật chơi
- Quản lý việc sửa lỗi và debug
- Slide thuyết trình

Các file và hàm cụ thể:

- Move.java
  - Undo()
- WinPopUp.java
- GameController.java
  - ClickSurrender()
  - makeMove()
  - clickUndo()
  - copyintersectionPoint()
  - getBoard()
  - makeVay()
  - checkVay()
  - changeColor()

1.3 Guarantee

Nhóm chúng em cam kết rằng mọi ý tưởng thiết kế trong mini project cờ gánh của chúng em là tự sáng tạo và không phản ánh bất kỳ sự sao chép hay sửa đổi từ bất kỳ nguồn nào khác ngoài ý tưởng của thành viên trong nhóm.
Mọi dòng code trong mini project đều được viết từ đầu và không sử dụng source code từ bất kỳ nguồn hay thư viện nào khác. Mọi sự trùng lặp nếu có, sẽ chỉ là trùng hợp.

2 Mini-project description

2.1 Mô tả ứng dụng

Trò chơi dân gian Cờ Gánh là một trò chơi tương đối phổ biến ở các vùng nông thôn Việt Nam. App game Cờ Gánh là ứng dụng game đơn giản, được xây dựng bằng javaFX và nền rảng kiến thức lập trình hướng đối tượng , đáp ứng nhu cầu chơi cờ truyền thống đơn giản cùng 1 số chức năng, tiện ích bổ sung.

2.1.1 Play

- **Bàn cờ:** Hiển thị bàn cờ Cờ Gánh với 25 vị trí và 16 quân cờ. Quân cờ được thiết kế rõ ràng, dễ nhận biết giữa hai bên.
- **Tính Điểm:** có hệ thống tính điểm theo số ván đấu và có hiển thị số lượng quân cờ của 2 bên trong ván đấu.
- Hiện thị rõ đâu là lươt đi của người chơi nào.
- **Đi lại Nước Cờ:** Nếu người chơi có nhu cầu đi lại nước cờ đi nhầm, có thể click undo.
- **Chịu Thua**: Nếu muốn dừng ván đấu có thể ấn nút chịu thua, người chơi còn lại nhận được điểm.
- **Thông Báo Chiến Thắng/Thua:** Hiển thị thông báo khi kết thúc ván đấu hoặc khi người chơi chịu thua.

2.1.2 Help

Cung cấp hướng dẫn chi tiết về cách chơi Cờ Gánh. Giải thích các quy tắc và chiến thuật cơ bản.

2.1.3 Exit

- Hiển thị hộp thoại xác nhận trước khi thoát ứng dụng
- Thoát ứng dụng

2.2 Usecase diagram

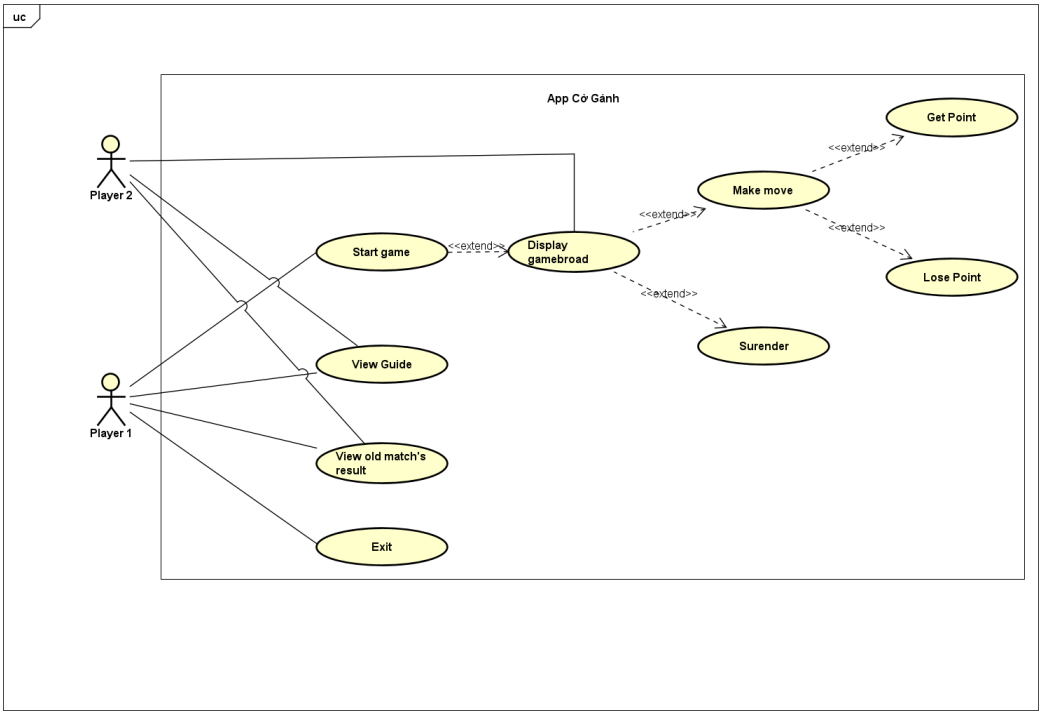


Figure 2.1 Usecase diagram

STT	Tên Actor	Vai trò
1	Player 1	Người điều khiển ứng dụng và chơi trò chơi
2	Player 2	Người chơi trò chơi

Table 2.1 Danh sách actor

STT	Tên usecase	Ý nghĩa
1	Start game	bắt đầu trò chơi
2	Display gamebroad	hiển thị bàn cờ
3	Make move	thực hiện nước đi trong ván cờ
4	Get Point	thắng ván cờ và được cộng điểm
5	Lose Point	Thua ván đấu và sẽ bị trừ điểm
6	Surrender	Đầu hàng ván đang đấu, đối thủ sẽ được điểm
7	View score	Xem điểm số
8	Show guide	Xem hướng dẫn chơi
9	View old match's result	Xem kết quả các ván đấu trước đó
10	Exit	Thoát ứng dụng

Table 2.2 Danh sách usecase

3 Design

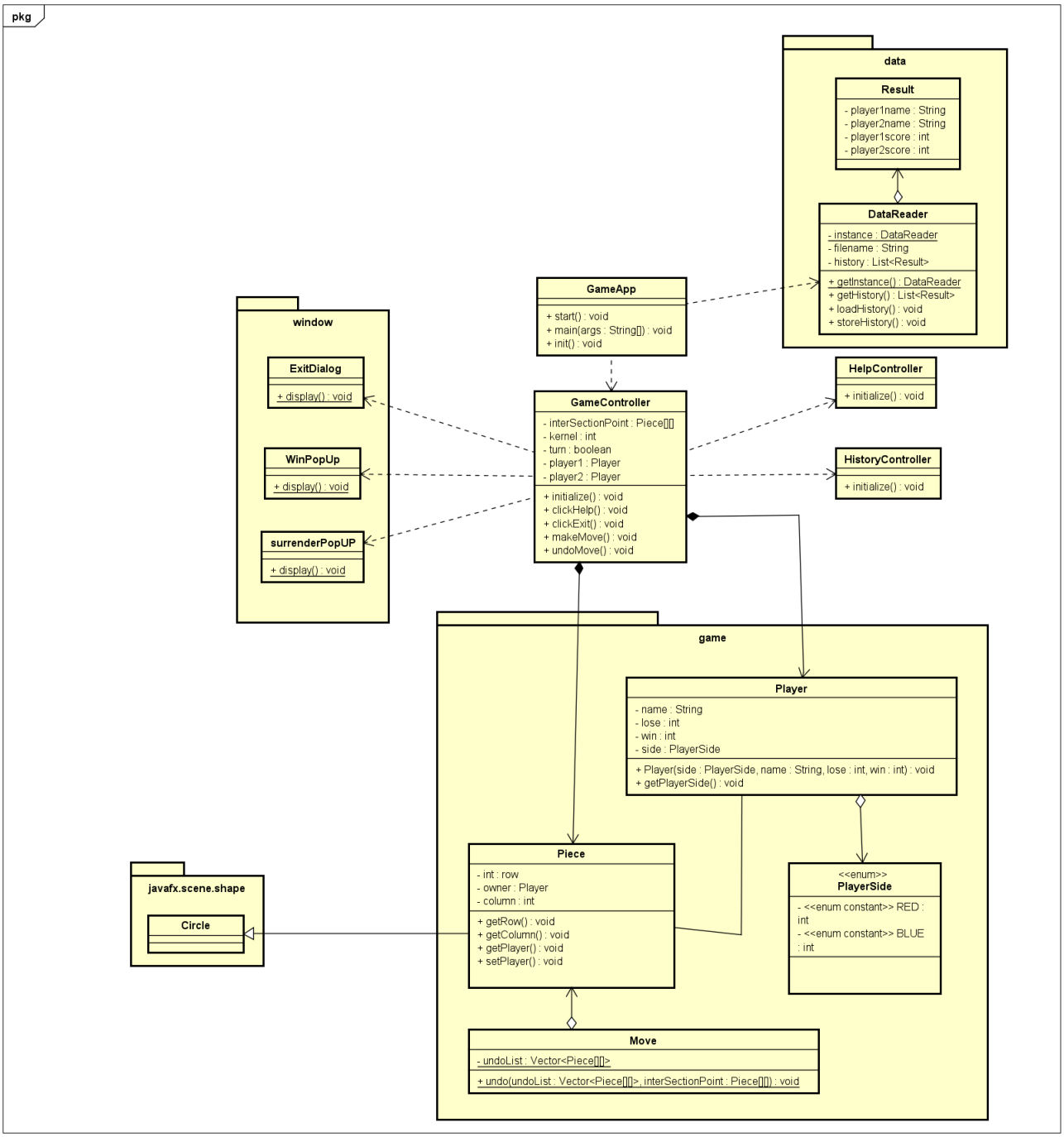


Figure 3.2 Class diagram

3.1 Package window

Chứa toàn bộ các class xử lí việc hiện lên các dialog trong ứng dụng. Tất cả các class đều có 1 static method display thực hiện chức năng hiển thị stage thông báo

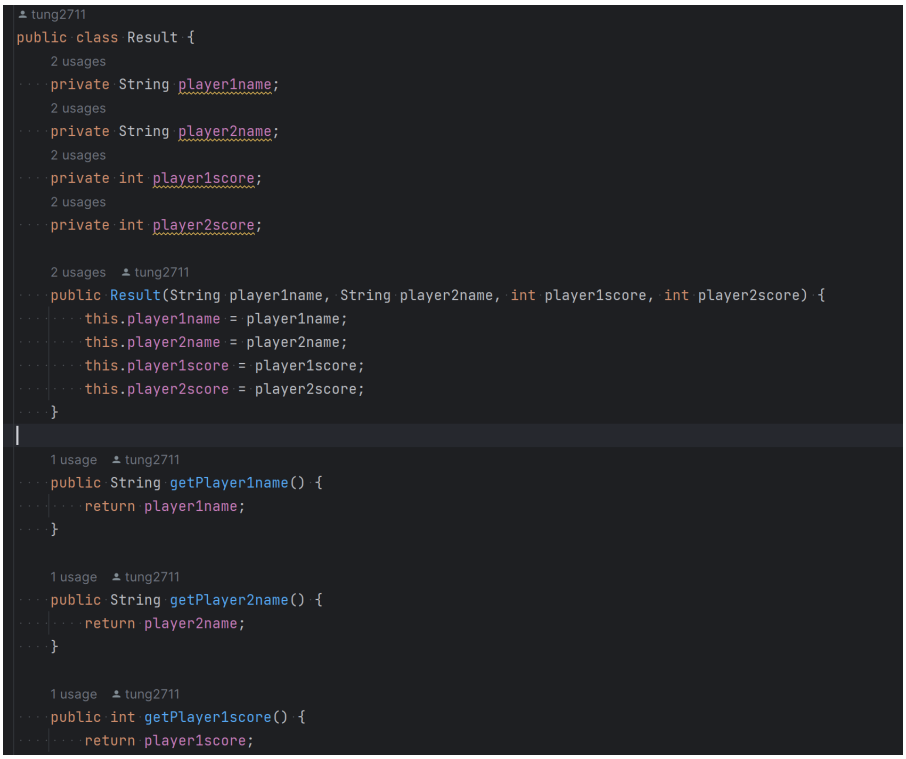
- **ExitDialog:** Hiện thị thông báo xác nhận thoát chương trình
- **WinPopUp:** Hiển thị thông báo người chiến thắng
- **SurrenderPopUp:** Hiển thị thông báo khi người chơi bỏ cuộc

3.2 Package data

Package thực hiện chức năng lưu lại kết quả trận đấu vào lịch sử.

3.2.1 Result

class có chức năng như 1 record lưu lại tên player và tỉ số của trận đấu.



```
tung2711
public class Result {
    2 usages
    private String player1name;
    2 usages
    private String player2name;
    2 usages
    private int player1score;
    2 usages
    private int player2score;

    2 usages  tung2711
    public Result(String player1name, String player2name, int player1score, int player2score) {
        this.player1name = player1name;
        this.player2name = player2name;
        this.player1score = player1score;
        this.player2score = player2score;
    }

    1 usage  tung2711
    public String getPlayer1name() {
        return player1name;
    }

    1 usage  tung2711
    public String getPlayer2name() {
        return player2name;
    }

    1 usage  tung2711
    public int getPlayer1score() {
        return player1score;
```

3.2.2 DataReader

class dùng để đọc và lưu giữ liệu được thiết kế theo desgin pattern *singleton*. Dữ liệu sẽ được lưu vào file "historyData.txt" (nếu không có sẽ tự động tạo)

```
tung2711
public class DataReader {
    1 usage
    private static DataReader instance = new DataReader();
    2 usages
    private static String filename = "historyData.txt";

    5 usages
    private List<Result> historyData;

    4 usages  tung2711
    public static DataReader getInstance() {
        return instance;
    }
```

- Phương thức đọc dữ liệu

```
1 usage  tung2711
public void loadHistory() throws IOException {

    historyData = FXCollections.observableArrayList();
    Path path = Paths.get(filename);
    BufferedReader br = Files.newBufferedReader(path);

    String input;

    try {
        while ((input = br.readLine()) != null) {
            String[] DataPieces = input.split( regex: "\t");

            String firstPlayerName = DataPieces[0];
            String secondPlayerName = DataPieces[1];
            String firstPlayerScore = DataPieces[2];
            String secondPlayerScore = DataPieces[3];
            Result result = new Result(firstPlayerName,secondPlayerName,
                    Integer.parseInt(firstPlayerScore),Integer.parseInt(secondPlayerScore));
            historyData.add(result);
        }

    } finally {
        if(br != null) {
            br.close();
        }
    }
}
```

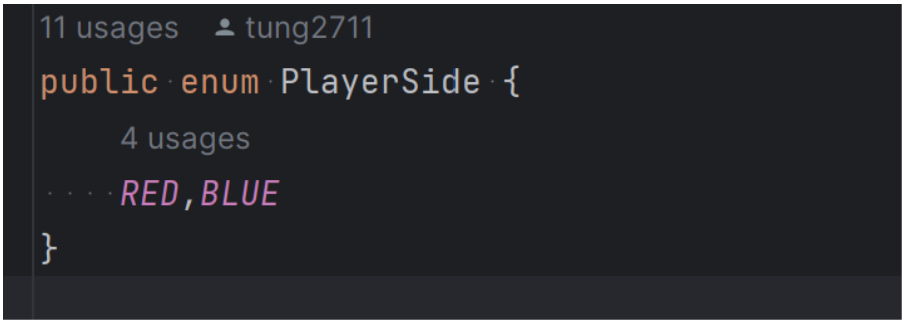
- Phương thức ghi dữ liệu

```
public void storeHistory() throws IOException {

    Path path = Paths.get(filename);
    BufferedWriter bw = Files.newBufferedWriter(path);
    try {
        Iterator<Result> iter = historyData.iterator();
        while (iter.hasNext()) {
            Result item = iter.next();
            bw.write(String.format("%s\t%s\t%s\t%s",
                    item.getPlayer1name(),
                    item.getPlayer2name(),
                    item.getPlayer1score(),
                    item.getPlayer2score())
            );
            bw.newLine();
        }

    } finally {
        if (bw != null) {
            bw.close();
        }
    }
}
```

3.3 Package game

Package gồm các class thực hiện các chức năng phục vụ cho trò chơi

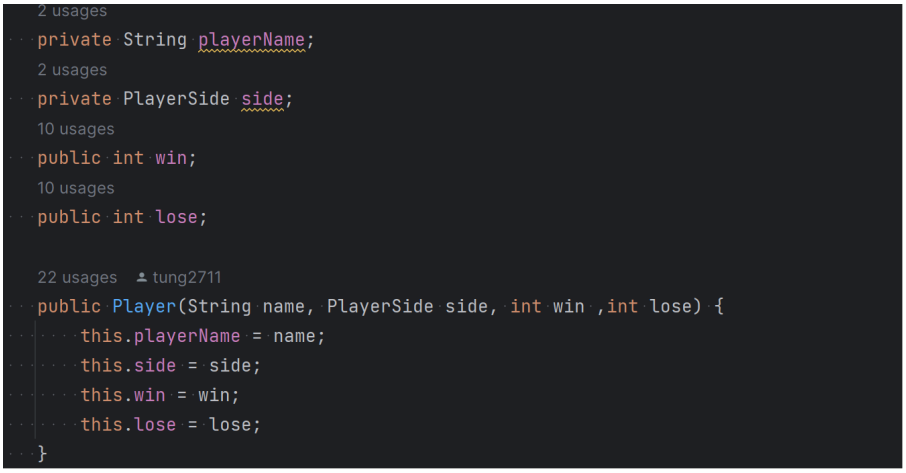
3.3.1 *Enum PlayerSide*



Biểu thị phe của người chơi. Gồm 2 giá trị RED,BLUE

3.3.2 *Player*

Class đại diện cho người chơi



Các atribute của Player

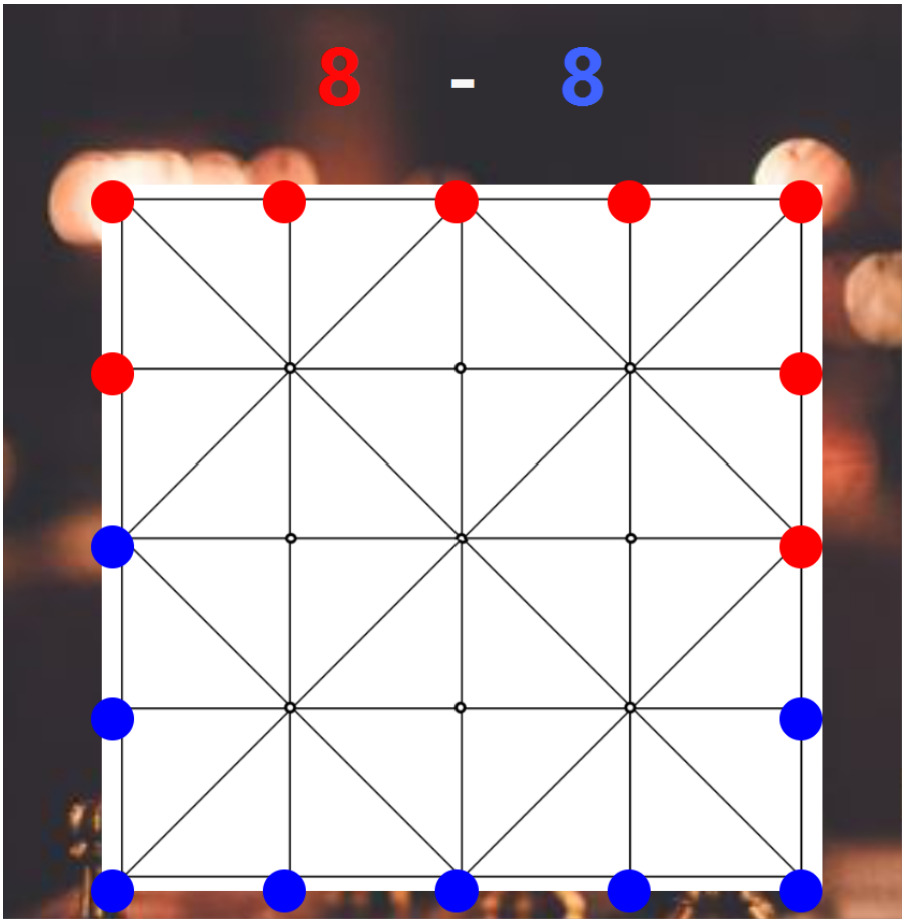
- *playerName*: tên của player
- *side*: phe của play
- *win*: số lần thắng
- *lose*: số lần thua

Player cũng có các phương thức getter và setter cho attribute cần dùng đến.

3.3.3 *Piece*

Class đại diên cho các quân cờ trong ván cờ. Vì các Piece được hiển thị là các chấm tròn trong UI nên Piece kế thừa từ class javafx.scene.shape.Circle

```java
import javafx.scene.shape.Circle;

21 usages  tung2711
public class Piece extends Circle {
    2 usages
    private int row;
    2 usages
    private int column;

    3 usages
    private Player player;
```



Các atribute của peice:

- *row,column*: biển diễn hàng và cột của quân cờ trong bàn cờ
- *player*: thể hiện Player đang sở hữu quân cờ
- và các thuộc tính sẵn có của javafx.scence.shape.Circle

Constructor của Piece:

Màu của piece thay đổi theo Player sở hữu Piece. Nếu người sở hữu quân cờ có side là RED

thì quân cờ sẽ là màu đỏ hoặc xanh nếu ide là BLUE. Nếu không có player nào sở hữu piece đó thì setOpacity cho quân cờ thành 0.

```
public Piece(int row, int column, Player player) {
    super( v: 80+80*column, v1: 113+80*row, v2: 10);
    if(player == null) {
        setOpacity(0);
    } else {
        if (player.getSide() == PlayerSide.RED) {
            setFill(Color.RED);
        } else {
            setFill(Color.BLUE);
        }
    }
    this.row = row;
    this.column = column;
    this.player = player;
}
```

Peice cũng có các phương thức getter và setter cho attribute cần dùng đến.

3.3.4 Move

Class lưu trữ lại mọi thay đổi của ván cờ sau các lượt di chuyển của hai người chơi.

```
package com.example.coganhapp.game;

import javafx.scene.paint.Color;

import java.util.Vector;

public class Move {
    2 usages
    public static Vector<Piece[][]> undoList= new Vector<>();

    1 usage
```

Các atribute của Move:

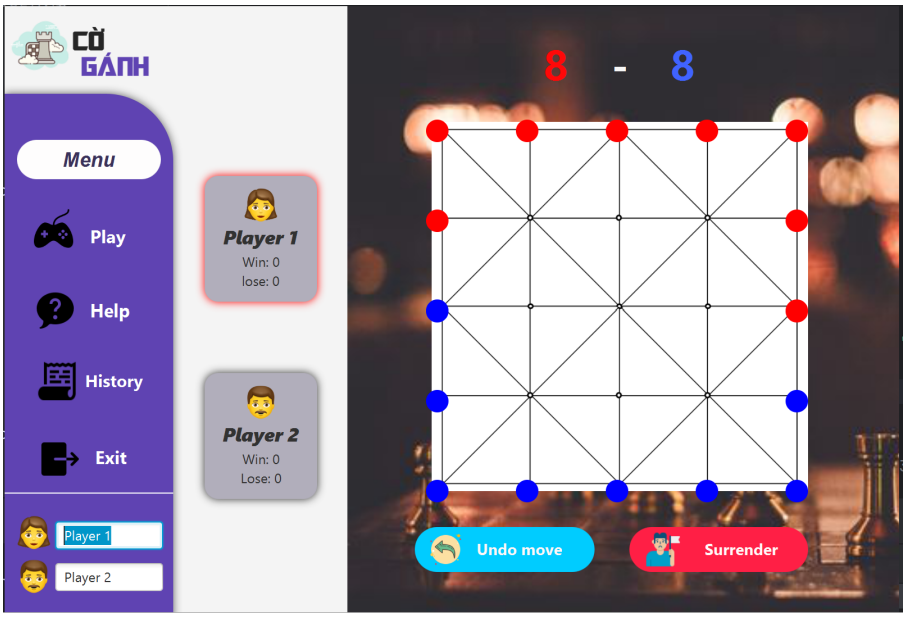
- *undoList*: lưu trữ tất cả sự thay đổi của ván cờ sau những lượt di chuyển

```java
1 usage
public static void Undo(Vector<Piece[][]> undoList, Piece[][] intersectionPoint) {
    if (undoList.size() != 0) {
        for(int i = 0;i<5;i++) {
            for(int j = 0;j<5;j++) {
                if (undoList.get(undoList.size()-1)[i][j].getPlayer() != null) {
                    intersectionPoint[i][j].setPlayer(undoList.get(undoList.size()-1)[i][j].getPlayer());
                    intersectionPoint[i][j].setFill(undoList.get(undoList.size()-1)[i][j].getFill());
                    intersectionPoint[i][j].setOpacity(1);
                } else {
                    intersectionPoint[i][j].setPlayer(null);
                    intersectionPoint[i][j].setFill(Color.BLUE);
                    intersectionPoint[i][j].setOpacity(0);
                }
            }
        }
        undoList.remove( index: undoList.size()-1);
    }
}
}
```

Ngoài ra Move còn có phương thức Undo cho phép cả hai người chơi có thể quay trở lại những lượt đi trước đó.

3.4 GameController

Controller cho GameUI.fxml



Các attribute của GameController:

- 2 biến player1 và player2 thuộc class Player.

```java
14 usages
private Player player1 = new Player( name: "Player 1", PlayerSide.RED, win: 0, lose: 0);
14 usages
private Player player2 = new Player( name: "Player 2", PlayerSide.BLUE, win: 0, lose: 0);
```

- turn: biến boolean để xác định đang là lượt đi của người chơi nào
- intersectionPoint: mảng 2 chiều thuộc kiểu Piece[5][5] để xây dựng bàn cờ 5x5 quân cờ.

- Các biến FXML

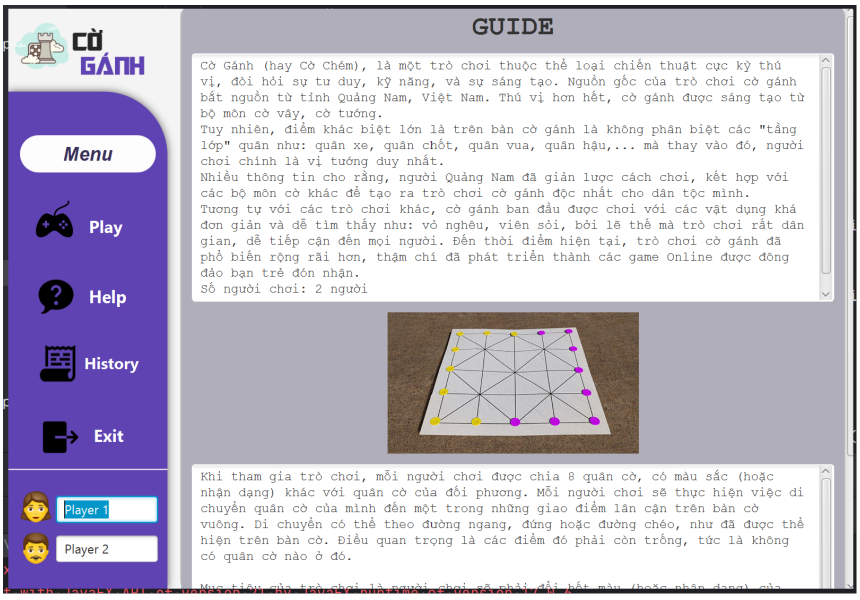
Giải thích các hàm trong GameController.java:

- Phương thức *initialize()*

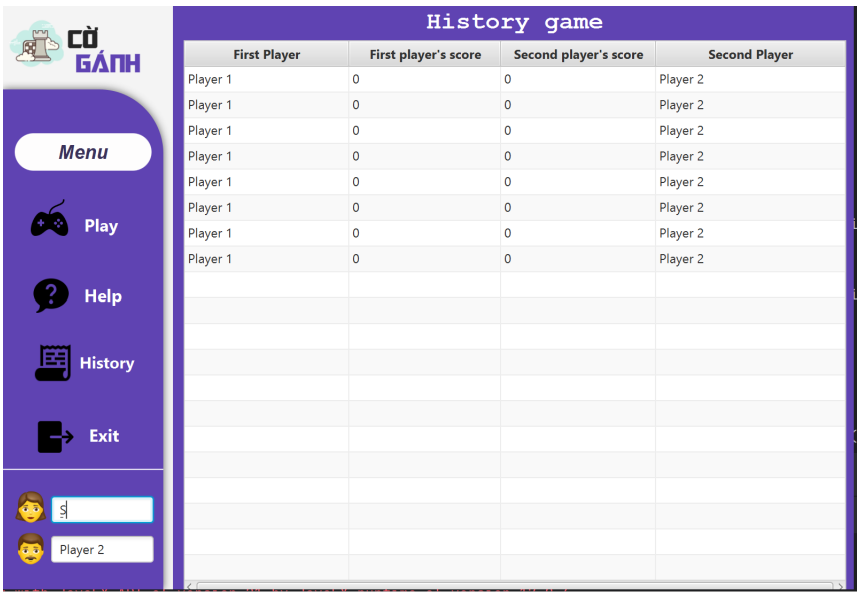
```
tung2711 *
@FXML
public  void initialize() {
    glowEffect();
    for(int i = 0;i<5;i++) {
        for(int j = 0;j<5;j++) {
            if(i == 0 || ((i == 1 || i == 2) && j == 4) || (i == 1 && j == 0)) {
                intersectionPoint[i][j] = new Piece(i,j,player1);
            } else if(i == 4 || ((i == 2 || i == 3) && j == 0) || (i == 3 && j == 4)) {
                intersectionPoint[i][j] = new Piece(i,j,player2);
            } else {
                intersectionPoint[i][j] = new Piece(i,j, player: null);
            }
            gameBroad.getChildren().add(intersectionPoint[i][j]);
            tung2711
            intersectionPoint[i][j].addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
                tung2711
                @Override
                public void handle(MouseEvent mouseEvent) {
                    Piece source = (Piece) mouseEvent.getSource();
                    int row = source.getRow();
                    int columm = source.getColumn();
                    makeMove(row,columm);
                }
            });
        }
    }
}
```

Tạo 25 instance Piece và gán vào các phần tử trong mảng intersectionPoint. ta để attribute row, column, player của các piece đó theo vị trí bàn cờ lúc đầu. Add các Piece vào anchorPane cha chứa hình ảnh bàn cờ. Gán cho 25 Piece EventHandler khi ấn vào sẽ thực hiện hàm makeMove()

- Hàm *clickOptionButton():*
  Có nhiệm vụ switch giữa các UI của ứng dụng



UI hiển thị hướng dẫn chơi



UI hiển thị lịch sử các ván đấu

```
@FXML
protected void clickOptionButton(MouseEvent event) throws IOException {
    if(event.getSource() == playBtn) {
        if(helpKernel == 1) {
            ancherRoot.getChildren().remove( index: ancherRoot.getChildren().size()-1);
            helpKernel = 0;
        } else if(historyKernel == 1) {
            ancherRoot.getChildren().remove( index: ancherRoot.getChildren().size()-1);
            historyKernel = 0;
        }
    } else if(event.getSource() == helpButton){
        if(helpKernel == 0) {
            if (historyKernel == 1) {
                ancherRoot.getChildren().remove( index: ancherRoot.getChildren().size()-1);
                historyKernel = 0;
            }
            Parent root =  FXMLLoader.load(Objects.requireNonNull(getClass().getResource( name: "HelpUI.fxml"))
            ancherRoot.getChildren().add(root);
            helpKernel = 1;
        }
    } else {
        if (historyKernel == 0) {
            if (helpKernel == 1) {
                ancherRoot.getChildren().remove( index: ancherRoot.getChildren().size() - 1);
                helpKernel = 0;
            }
            Parent root = FXMLLoader.load(Objects.requireNonNull(getClass().getResource( name: "history.fxml"))
            ancherRoot.getChildren().add(root);
            historyKernel = 1;
        }
    }
}
```

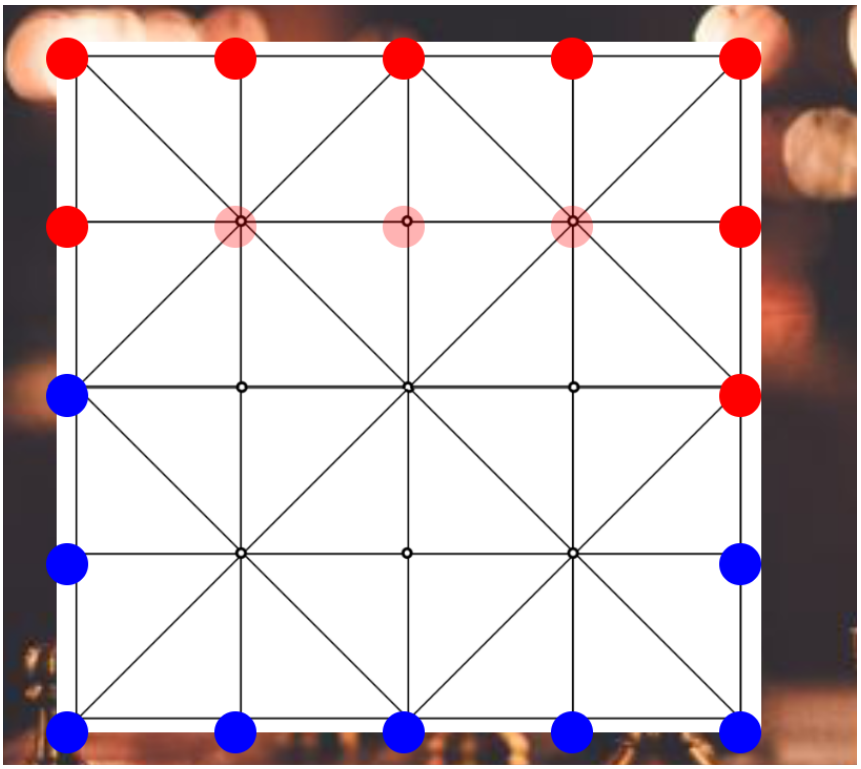
Hàm được thực thi khi click vào các Lựa chọn trong thanh menu bên trái. Tạo 2 biến helpKernel và historyKernel kiểm tra xem UI help và UI history có đang được hiển thị không. dựa vào việc kiểm tra các kernel, anchorpane cha chứa toàn bộ nội dung bên phải sẽ add và remove các UI.

- Hàm *makeMove()*:

  Khi di chuyển cần 2 lần click chuột: lần thứ nhất vào quân cờ ta muốn di chuyển và lần thứ hai vào điểm ta muốn quân cờ đó di chuyển đến. Đòi hỏi ta phải 1 biến int kernel khởi tạo bằng 0.

  Khi kernel bằng 0, ta phải kiểm tra xem quân ta click vào có player bằng null không và

tùy thuộc vào biến turn ta chỉ có thể ấn vào quân cờ đỏ hoặc xanh. Nếu tất cả các điều kiện thỏa mãn, ta tăng kernel lên 1 và hiện mờ các điểm quân cờ đó có thể di chuyển đến. và lưu lại row và column của quân cờ ta vừa click vào biến tmpRow, tmpColumn.



Khi ấn vào quân cờ giữa ở hàng đầu tiên, 3 quân cờ giữa hàng thứ 2 sẽ hiện mờ, thể hiện cho người chơi vị trí họ có thể đi

Khi kernel bằng 1, ta phải kiểm tra xem quân cờ được click chuột lần này có opacity > 0 và < 1 không (tức là có thể di chuyển được từ quân của lần click chuột trước). Ta setPlayer của quân cờ lần này thành player của quân cờ được lưu vào tmpRow và tmpColumn (quân cờ ta click vào khi kernel bằng 0) và đổi màu cho quân cờ hiện tại cùng màu với quân cờ đó. Còn quân cờ của lần click trước sẽ được setPlayer về null và setOpacity về 0. Cuối cùng, ta chuyển kernel về 0, turn = !turn và thực hiện kiểm tra gánh và vây.

- *getBoard():*
  Phương thức khi chạy sẽ kiểm tra và đếm lại số quân cờ của mỗi đội rồi đưa ra màn hình. Nếu số quân cờ của một đội nào đó bằng 16 thì sẽ xác định được người thắng

- *makeVay():*
  Mỗi khi người chơi di chuyển xong thì phương thức sẽ duyệt lại cả bàn cờ và nếu có quân cờ bị vây thì sẽ đổi màu quân cờ đó

- *checkVay():*
  Phương thức sẽ kiểm tra xem một hướng của quân cờ có bị vây hay không

- *changeColor():*
  Phương thức đổi màu và người chơi của một quân cờ

- *makeGanh():*
  Hàm sẽ thực hiện gánh hai quân cờ nếu quân cờ vừa di chuyển đi vào giữa hai quân cờ khác

- *isSurrounded():*
  Hàm kiểm tra xem 1 quân cờ có bị bao vây không. Sử dụng thuật toán duyệt chiều sâu.

3.5 HelpController

class controller cho phần hiển thị helpUI.fxml.

Các phương thức trong HelpController:

Phương thức initialize(): SetText cho các textArea trong phần hướng dẫn. setEditable là false.

3.6 HistoryController

class controller cho phần hiển thị history.fxml.

Các phương thức trong HistoryController:

Phương thức initialize(): Truyền các data vào tableview để hiện thị cho người dùng.

3.7 CoGanhApp

Lớp chính của một ứng dụng được mở rộng từ Application.

Các phương thức trong CoGanhApp:

- start(): Phương thức này được gọi khi ứng dụng bắt đầu chạy, sử dụng nó để tạo ra cửa sổ chính.
- init(): Phương thức chạy trước start() ,dùng để đọc dữ liệu từ file historyData.txt
- stop(): Phương thức chạy khi dừng ứng dụng, dùng để ghi dữ liệu trận đấu vào file historyData.txt

3.8 Giải thích quan hệ trong class diagram

- Piece - Circle : Piece kế thừa Circle -> **inheritance**
- GameController - Player: 2 player được khởi tạo khi trò chơi bắt đầu và biến mất khi trò chơi mất đi -> **composition**
- GameController - Piece: 25 quân cờ được tạo ra khi trờ chơi bắt đầu và biến mất khi trờ chơi kết thúc -> **composition**
- Player - PlayerSide: PlayerSide là 1 thuộc tính cần thiết để xây dựng Player, PlayerSide vẫn tồn tại nếu không có Player -> **aggregation**
- Player - Piece: 1 player có thể sở hữu nhiều quân cờ -> **association**
- Result - DataReader: Result là 1 thuộc tính để xây dựng DataReader -> **aggregation**
- Move - Piece: Piece là thuộc tính để xây dựng Move -> **aggregation**