ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

Xây dựng game Cờ Caro

Môn: Nhập môn trí tuệ nhân tạo

Nhóm 27

Lê Minh Hiếu 20215048

Đoàn Thái Việt 20215164

Phạm Minh Thắng 20215139

Dương Văn Nhất 20215106

Hoàng Đức Dương 20215021

Hà Nội, ngày 09 tháng 06 năm 2024

I. GIỚI THIỆU ĐỀ TÀI:

Cờ ca-rô là một trò chơi nhân gian dùng viết trên bàn cờ giấy với số lượng ô không giới hạn. Hai người chơi, người dùng ký hiệu **X**, người kia dùng ký hiệu **O**, lần lượt điền ký hiệu của mình vào các ô. Người thắng là người có thể tạo được đầu tiên một chuỗi liên tục gồm 5 ô hàng ngang (có thể là dọc hoặc chéo).

Từ các kiến thức sẵn có và dựa vào đồ án game Tic-tac-toe, nhóm mình quyết định sẽ tạo ra một chương trình chơi cờ ca-rô.

II.MỤC TIÊU:

Mục tiêu của nhóm là tạo ra một ứng dụng chơi cờ ca-ro từ ngôn ngữ JAVA với sự hỗ trợ của công cụ lập trình Intellij cùng một số tài liệu tham khảo trên mạng. Nhóm hi vọng trò chơi sẽ mang lại những phút giây thư giãn để xả stress.

Phân chia công việc:

Nhóm Sinh viên 1: Dương Văn Nhất, Lê Minh Hiếu

- Xây dựng các nút chức năng
- Thiết kế giao diện trò chơi
- Thiết kế, tạo EasyBot (bot dễ)
- Tập hợp các hàm để hoàn thiện sản phẩm
- Mức độ hoàn thành: 100%

Nhóm Sinh viên 2: Hoàng Đức Dương, Đoàn Thái Việt

- Xây dựng thuật toán Heuristic, tạo HeuristicBot (bot thường)
- Kiểm tra lỗi, chỉnh sửa
- Mức đô hoàn thành: 100%

Sinh viên 3: Phạm Minh Thắng

- Thiết kế chế độ chơi 2 người
- Kiểm tra lỗi.
- Mức đô hoàn thành: 100%

Sinh viên 4: Hoàng Đức Dương

- Viết báo cáo, trình bày powerpoint.
- Kiểm tra lỗi.
- Mức độ hoàn thành: 100%

III. NỘI DUNG:

1. Các chức năng trong ứng dụng:

Trò chơi Ca-rô gồm các chức năng sau:

- **Nhập tên người chơi:** Người chơi cần nhập tên vào để chơi game, nếu không nhập tên sẽ không vào được giao diện chơi game.
- **Chọn số ô trong bàn cờ Caro:** người dùng có thể chọn số ô trong bàn cờ Ca-rô bằng cách nhập vào số dòng=cột với giá trị thấp nhất là 5 (25 ô) và giá trị cao nhất là 20 (400 ô).
- **Chơi 2 người:** 2 người chơi với nhau, người 1 xong lượt mình thì tới người 2. Muốn chơi 2 người cần nhập tên đăng nhập cho cả 2 người.
- **Chơi với máy:** Người chơi đấu với Bot, chỉ cần nhập tên đăng nhập cho người chơi thứ nhất. Bot có 2 độ khó là **dễ** và **thường**.
  - **Easy Bot:** Bot đi ngẫu nhiên.
  - **Normal Bot:** Bot sử dụng thuật giải Heuristic.
- **Chế độ đi lại và bỏ đi lại:** trong 1 bàn ca-rô, khi lỡ đánh một nước sai thì có thể chọn đi lại để xóa nước đó và đánh lại nước khác. Còn nếu thấy ưng thuận với nước đã đánh mà lỡ chọn đi lại thì chỉ cần chọn bỏ đi lại là xong.

2. Phân tích ứng dụng:

a) Danh sách các Class chủ yếu sử dụng trong ứng dụng:

- Các Class trong gói Java Swing được sử dụng cho ứng dụng. Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng window-based. Nó được xây dựng trên API AWT (Abstract Windowing Toolkit) và được viết hoàn toàn bằng Java. Không giống như AWT, Java Swing cung cấp các thành phần không phụ thuộc vào nền tảng và nhẹ hơn.

- Main: Lớp chính của ứng dụng, chạy lớp này để khởi động ứng dụng.

- Play2Players: Tạo bảng trò chơi, thực hiện chế độ chơi với người.

- DrawCanvas: Được tạo trong lớp Play2PLayers. Thực hiện việc vẽ các thành phần trong bảng trò chơi.

- Play2PlayersCaro: Kế thừa từ lớp Play2PLayers, xử lý thuật toán thắng thua khi xài bàn cờ 5 dòng 5 cột trở lên.

- PlayWithAi: Kế thừa từ lớp Play2Players, thực hiện chế độ chơi với máy.
  - HeuristicBot: Được tạo trong lớp PlayWithAi, thực hiện việc xử lý thuật toán Bot thường.

- PlayWithAiCaro: Kế thừa từ lớp PlayWithAi, xử lý thuật toán thắng thua.

- EasyBot: Xử lý thuật toán Bot dễ.

- JFrameMain: Hiển thị giao diện khởi đầu. Thực hiện việc hiển thị các giao diện khác thông qua một số buttons.

b) Nội dung Class:

- Các lớp trong gói Java Swing được sử dụng cho giao diện của ứng dụng:

- JFrame: Là khung của ứng dụng, chứa các thành phần như labels, buttons, textfields, tables,… để tạo ra một GUI. JFrame của Swing là một phiên bản kế thừa từ java.awt.Frame mà bổ sung các hỗ trợ cho cấu trúc thành phần JFC/Swing.

- JPanel: Là một container (thùng chứa) dùng để chứa các đối tượng tương tự như JFrame tuy nhiên nó không phải là 1 JFrame. Dễ hiểu hơn thì bạn có thể hình dùng ngôi nhà của chúng ta là 1 JFrame, còn phòng ngủ, phòng khách, phòng ăn đó là các JPanel, tức là trong một JFrame chứa các JPanel, trong mỗi JPanel lại có thể chứa các đối tượng hoặc thậm chí là các JPanel khác.

- JLabel: Thường được dùng để hiển thị text hoặc hình ảnh để tạo các chỉ dẫn, hướng dẫn trên giao diện người dùng.

➢ JButton: Được sử dụng để tạo một nút button mà có trình triển khai là độc lập nền tảng. Thành phần này có một label và tạo một sự kiện (event) khi được nhấn. Nó cũng có thể có Image.

➢ JTextField: Là một điều khiển văn bản cơ bản cho phép người dùng có thể điền một lượng văn bản nhỏ như họ tên, email, điện thoại, … Chẳng hạn, khi người dùng điền xong dữ liệu thì sẽ Text Field sẽ kích hoạt một Action Event.

➢ JCheckBox: Là đối tượng cho phép chúng ta chọn nhiều thuộc tính.

➢ JOptionPane: Là một thành phần cung cấp các phương thức chuẩn để popup một hộp thoại dialog chuẩn cho một giá trị hoặc thông báo người dùng về một cái gì đó.

➢ JTable: Được sử dụng để hiển thị dữ liệu trên các ô của bảng hai chiều.

➢ JSpinner: là một thành phần cho phép người dùng lựa chọn một số hoặc một giá trị đối tượng từ một dãy đã qua sắp xếp bởi sử dụng một trường đầu vào.

- Main:

```
public class Main {

    public static  JFrameMain jFrameMain;
    public static void main(String[] args){
        jFrameMain= new JFrameMain();
        jFrameMain.CreateAndShow();
    }
}
```

➢ Run Main để khởi động ứng dụng. Khi khởi động sẽ hiển thị giao diện khởi đầu của JFrameMain.

- Play2Players:

```
public class Play2Players extends JFrame
```

➢ Kế thừa lớp JFrame.

```
public static int newRow =3;
protected Seed PlayerReRun;
public static int rowPreSelected = -1;
public static int colPreSelected = -1;
public static int rowPreDiLai;
public static int colPreDiLai;
// Named-constants for the game board
public static  int ROWS = 3;  // ROWS by COLS cells
public static  int COLS = 3;
public static String Player1Name;
public static String Player2Name;
public static boolean Player1TwoMove;
public static int STEPS=0;
```

- newRow: chỉ số dòng cột của bảng Caro, có mặc định bằng 3.

- PlayerReRun: chỉ người chơi cần đi lại.

- rowPreSelected, colPreSelected: chỉ vị trí của một ô trong bảng đã đánh dấu trong lượt trước của người chơi, được sử dụng để tìm vị trí đi trước đó của người chơi.

- rowPreDiLai, colPreDiLai: chỉ vị trí của một ô trong bảng đã đánh dấu trong lượt trước của người chơi, 2 biến này được sử dụng cho chức năng **Đi lại** và **Bỏ đi lại**.

- Khai báo biến và hằng số cho bảng Tic-tac-toe: ROWS và COLS cho biết số dòng và cột của bảng, 2 biến Player1Name và Player2Name là tên của người chơi 1 và người chơi 2 và biến STEPS chỉ lượt đánh đầu của người chơi.

```
// Named-constants of the various dimensions used for graphics drawing
public static  int CELL_SIZE = 100; // cell width and height (square)
public static  int CANVAS_WIDTH = CELL_SIZE * COLS;  // the drawing canvas
public static  int CANVAS_HEIGHT = CELL_SIZE * ROWS;
public static  int GRID_WIDTH = 2;                   // Grid-line's width
public static  int GRID_WIDHT_HALF = GRID_WIDTH / 2; // Grid-line's half-width
// Symbols (cross/nought) are displayed inside a cell, with padding from border
public static  int CELL_PADDING = CELL_SIZE / 6;
public static  int SYMBOL_SIZE = CELL_SIZE - CELL_PADDING * 2; // width/height
public static  int SYMBOL_STROKE_WIDTH = 8; // pen's stroke width
```

➢ Khai báo hằng số dùng cho giao diện đồ họa: CELL_SIZE chỉ độ rộng và cao của mỗi ô vuông trong bảng, CANVAS_WIDTH và CANVAS_HEIGHT chỉ độ rộng và độ cao của khoảng trống hình chữ nhật (nơi ứng dụng có thể vẽ hoặc bẫy các sự kiện đầu vào từ người dùng), GRID_WIDTH chỉ độ rộng đường kẽ giữa các ô vuông và GRID_WIDHT_HALF chỉ độ rộng một nửa của GRID_WIDTH.

➢ Khai báo hằng số cho việc hiển thị ký hiệu **X**/**O** trong ô vuông: CELL_PADDING chỉ độ cao rộng của khoảng trống giữa kí hiệu và viền ô vuông, SYMBOL_SIZE chỉ độ cao và rộng của ký hiệu và SYMBOL_STROKE_WIDTH chỉ độ rộng của nét vẽ cho ký hiệu.

```
// Use an enumeration (inner class) to represent the various states of the game
public enum GameState {
    PLAYING, DRAW, CROSS_WON, NOUGHT_WON
}
protected GameState currentState;  // the current game state

// Use an enumeration (inner class) to represent the seeds and cell contents
public enum Seed {
    EMPTY, CROSS, NOUGHT
}
protected Seed currentPlayer;  // the current player


public Seed[][] board   ; // Game board of ROWS-by-COLS cells
protected DrawCanvas canvas; // Drawing canvas (JPanel) for the game board
protected JLabel statusBar;  // Status Bar
protected JPanel pnButton;
protected Button btnDiLai;
protected Button btnBoDiLai;
```

➢ GameState chỉ trạng thái của game, được định nghĩa bởi enum với 4 giá trị: PLAYING (thắng), DRAW (hòa), CROSS_WON (X thắng), NOUGHT_WON (O thắng).

➢ currentState: Chỉ trạng thái hiện tại của game.

➢ Seed đại diện cho thứ ô đang chứa, được định nghĩa từ enum với 3 giá trị: EMPTY (ô trống), CROSS (ô đánh X), NOUGHT(ô đánh O).

➢ currentPlayer: Chỉ lượt người chơi hiện tại.

➢ Board: Tên mảng 2 chiều dùng để tạo bảng dòng theo cột.

➢ Tạo đối tượng canvas từ lớp DrawCanvas để vẽ Canvas cho bảng trò chơi.

```
/** Constructor to setup the game and the GUI components */
public Play2Players(String name1, String name2) {
    STEPS=0;
    PlayGame(name1,name2);
}
```

➢ Khởi tạo để cài đặt trò chơi và thành phần GUI.

```
public void SetUpBoard(int row){
    ROWS = row;
    COLS = row;
    CELL_SIZE = (20/row) *30;
    CANVAS_WIDTH = CELL_SIZE * COLS;
    CANVAS_HEIGHT = CELL_SIZE * ROWS;
    CELL_PADDING = CELL_SIZE / 6;
    SYMBOL_SIZE = CELL_SIZE - CELL_PADDING * 2;
}
```

➢ SetUpBoard(): dùng để set lại kích cỡ bàn cờ, từ việc nhập số dòng/cột ở giao diện khởi đầu rồi set lại bàn cờ sao cho hợp lý.

```
/** Initialize the game-board contents and the status */
public void initGame() {
    btnDiLai.setEnabled(false);
    btnBoDiLai.setEnabled(false);
    STEPS = 0;
    for (int row = 0; row < ROWS; ++row) {
        for (int col = 0; col < COLS; ++col) {
            board[row][col] = Seed.EMPTY; // all cells empty
        }
    }
    currentState = GameState.PLAYING; // ready to play
    currentPlayer = Seed.CROSS;       // cross plays first
}
```

➢ initGame(): Hàm này dùng để khởi tạo nội dung bảng trò chơi và trạng thái của game. Set button đi lại và bỏ đi lại tạm không sư dụng được, tạo STEPS = 0, làm tất cả ô vuông đều trống, current state sẽ là PLAYING và người chơi ký hiệu **X** sẽ đánh lượt đầu tiên.

```
public void updateGame(Seed theSeed, int rowSelected, int colSelected) {  3 usages   DuongHoang673
    if (hasWon(theSeed, rowSelected, colSelected)) {  // check for win
        currentState = (theSeed == Seed.CROSS) ? GameState.CROSS_WON : GameState.NOUGHT_WON;

        if(theSeed == Seed.CROSS){
            JOptionPane.showMessageDialog( parentComponent: null, message: Player1Name+" thắng rồi! Click chuột để chơi lại");
        }
        else {
            JOptionPane.showMessageDialog( parentComponent: null,  message: Player2Name + " thắng rồi! Click chuột để chơi lại");
            }
    } else if (isDraw()) {  // check for draw
        currentState = GameState.DRAW;
        JOptionPane.showMessageDialog( parentComponent: null, message: "Hòa rồi! Click chuột để chơi lại");
    }
    // Otherwise, no change to current state (still GameState.PLAYING).
}
```

➢ updateGame(): hàm dùng để cập nhật trạng thái của game. Nếu vẫn còn chơi được thì không thay đổi gì. Tuy nhiên, nếu một bên đạt điều kiện thắng (**X** hoặc **O**) thì chuyển trạng thái CROSS_WON hay NOUGHT_WON tùy theo người thắng là ai. Nếu trên bảng không còn ô vuông trống mà không có ai thắng cả thì chuyển trạng thái DRAW.

```
public boolean isDraw() {
    for (int row = 0; row < ROWS; ++row) {
        for (int col = 0; col < COLS; ++col) {
            if (board[row][col] == Seed.EMPTY) {
                return false; // an empty cell found, not draw, exit
            }
        }
    }
    return true;  // no more empty cell, it's a draw
}
```

➢ isDraw(): hàm boolean kiểm tra trận đấu có hòa không. Nếu đánh hết tất cả các ô trong bảng mà không phân được người thắng thì trả về true.

```java
public boolean hasWon(Seed theSeed, int rowSelected, int colSelected) {
    return (board[rowSelected][0] == theSeed  // 3-in-the-row
            && board[rowSelected][1] == theSeed
            && board[rowSelected][2] == theSeed
            || board[0][colSelected] == theSeed      // 3-in-the-column
            && board[1][colSelected] == theSeed
            && board[2][colSelected] == theSeed
            || rowSelected == colSelected            // 3-in-the-diagonal
            && board[0][0] == theSeed
            && board[1][1] == theSeed
            && board[2][2] == theSeed
            || rowSelected + colSelected == 2  // 3-in-the-opposite-diagonal
            && board[0][2] == theSeed
            && board[1][1] == theSeed
            && board[2][0] == theSeed);
}
```

➢ HasWon(): Là hàm boolean dùng để kiểm tra coi có người chơi nào đạt diều kiện thắng chưa.

```java
protected void PlayGame(String name1, String name2){ 1 usage  1 override  DuongHoang673
    SetUpBoard(newRow);
    Player1Name = name1;
    Player2Name = name2;
    canvas = new DrawCanvas();  // Construct a drawing canvas (a JPanel)
    canvas.setPreferredSize(new Dimension(CANVAS_WIDTH, CANVAS_HEIGHT));

    // The canvas (JPanel) fires a MouseEvent upon mouse-click
    canvas.addMouseListener((MouseAdapter) mouseClicked(e) -> {  // mouse-clicked handler
            int mouseX = e.getX();
            int mouseY = e.getY();
            // Get the row and column clicked
            int rowSelected = mouseY / CELL_SIZE;
            int colSelected = mouseX / CELL_SIZE;

            if (currentState == GameState.PLAYING) {
                if (rowSelected >= 0 && rowSelected < ROWS && colSelected >= 0
                        && colSelected < COLS && board[rowSelected][colSelected] == Seed.EMPTY) {
                    rowPreSelected = rowSelected;
                    colPreSelected = colSelected;
                    board[rowSelected][colSelected] = currentPlayer; // Make a move
                    updateGame(currentPlayer, rowSelected, colSelected); // update state
                    // Switch player
                    currentPlayer = (currentPlayer == Seed.CROSS) ? Seed.NOUGHT : Seed.CROSS;
                    STEPS++;
                }
                btnDiLai.setEnabled(true);
                btnBoDiLai.setEnabled(false);
            } else {       // game over
                initGame(); // restart the game
            }
            // Refresh the drawing canvas
            repaint();  // Call-back paintComponent().
    });
```

```
// Set up the status bar (JLabel) to display status message
statusBar = new JLabel( text: "  ");
statusBar.setFont(new Font(Font.DIALOG_INPUT, Font.BOLD,  size: 15));
statusBar.setBorder(BorderFactory.createEmptyBorder( top: 2,  left: 5,  bottom: 4,  right: 5));

//Thêm Button
btnDiLai = new Button( label: "Đi lại");
btnDiLai.setFont(new Font(Font.DIALOG_INPUT, Font.BOLD,  size: 15));
btnDiLai.setEnabled(false);
//btnDiLai.setSize(10,10);

btnDiLai.addActionListener(new ActionListener() {  DuongHoang673
    @Override  DuongHoang673
    public void actionPerformed(ActionEvent e) {
        if(currentState != GameState.PLAYING) return;
        if(STEPS==0) return;
        rowPreDiLai =rowPreSelected;
        colPreDiLai =colPreSelected;
        PlayerReRun = board[rowPreSelected][colPreSelected];
        currentPlayer = PlayerReRun;
        board[rowPreSelected][colPreSelected] = Seed.EMPTY;
        btnBoDiLai.setEnabled(true);
        btnDiLai.setEnabled(false);
        repaint();
    }
});
```

```
btnBoDiLai = new Button( label: "Bỏ đi lại");
btnBoDiLai.setFont(new Font(Font.DIALOG_INPUT, Font.BOLD,  size: 15));
btnBoDiLai.setEnabled(false);
//btnDiLai.setSize(10,10);

btnBoDiLai.addActionListener(new ActionListener() {  DuongHoang673
    @Override  DuongHoang673
    public void actionPerformed(ActionEvent e) {
        if(currentState != GameState.PLAYING) return;
        if(STEPS == 0 || PlayerReRun == null ) return;
        board[rowPreDiLai][colPreDiLai] = PlayerReRun;
        currentPlayer = (currentPlayer == Seed.CROSS) ? Seed.NOUGHT : Seed.CROSS;
        btnBoDiLai.setEnabled(false);
        repaint();
    }
});
```

```
    pnButton = new JPanel();
    pnButton.setLayout(new FlowLayout(FlowLayout.CENTER));
    pnButton.add(btnDiLai);
    pnButton.add(btnBoDiLai);

    Container cp = getContentPane();
    cp.setLayout(new BorderLayout());
    cp.add(canvas, BorderLayout.CENTER);
    cp.add(statusBar, BorderLayout.PAGE_END); // same as SOUTH
    cp.add(pnButton, BorderLayout.PAGE_START);


    pack();  // pack all the components in this JFrame
    setTitle("Tic Tac Toe 2 người");
    setLocationRelativeTo(null);
    setVisible(true);  // show this JFrame

    addWindowListener((WindowAdapter) windowClosing(e) -> { jFrame.setVisible(true); });

    board = new Seed[ROWS][COLS]; // allocate array
    initGame(); // initialize the game board contents and game variables
}
```

- PlayGame(): Hàm này dùng để thực hiện việc chơi game. Đầu tiên hàm sẽ tạo giao diện bảng trò chơi. Người chơi sẽ nhấp chuột vào một ô vuông để đánh kí hiệu, sau đó đến người chơi 2 (hoặc máy). Khi đang đánh thì 2 button đi lại sẽ sáng lên và có thể sử dụng được. Chơi tới khi nào 1 trong 2 thắng hoặc không còn ô trống nào thì ngừng. Muốn chơi tiếp thì nhấp chuột vào giao diện game để khởi tạo lại trò chơi.

- setPreferredSize(): Nằm trong lớp JComponent của gói javax.swing., phương thức này dùng để thiết lập flowLayout theo ý thích của mình vì mặc định flowLayout sẽ thiết lập kích thước cho các component con vừa đủ để bao bọc nội dung bên trong component đó. Dùng phương thức này để tạo bảng trò chơi với độ cao và rộng theo kích thước riêng của mình.

- addMouseListener(): Nằm trong lớp Component của gói java.awt.. Mouse Listener dùng để quản lý và xử lý các sự kiện liên quan đến chuột. addMouseListener() là phương thức dùng dể đăng ký sử dụng đối tượng.

- mouseClicked(): Được override từ lớp MouseAdapter của gói java.awt.. phương thức này được gọi hồi khi nút chuột đã được click (được nhấn và nhả ra) trên một thành phần. Trong trường hợp này thì khi click chuột sẽ xuất được ký hiệu X hay O.

- addWindowListener: Nằm trong lớp Window của gói java.awt.. WindowListener dùng để xử lí sự kiện trên Window. addWindowListener() là phương thức dùng dể đăng ký sử dụng đối tượng.

- windowClosing(): Được override từ lớp WindowAdapter của gói java.awt., dùng để xử lý việc khi đóng cửa sổ. Nếu đóng bảng trò chơi sẽ cho hiện lại giao diện khởi đầu.

- btnDiLai: Nút này được tạo cùng lúc với bảng trò chơi. Khi đánh dấu trong bàn cờ, nút này có thể sử dụng được. Khi nhấn vào sẽ xóa dấu mình mới đánh vào, làm nút **đi lại** không sử dụng được và làm nút **bỏ đi lại** sử dụng được.

- btnBoDiLai: Nút này được tạo cùng lúc với bảng trò chơi. Khi nhấn nút **đi lại**, nút này có thể sử dụng được. Khi nhấn vào sẽ bỏ đi hiệu ứng của nút **đi lại** và disable cả 2 nút **bỏ đi lại** và **đi lại.**

- DrawCanvas:

```java
public class DrawCanvas extends JPanel {
    @Override
    public void paintComponent(Graphics g) {  // invoke via repaint()
        super.paintComponent(g);    // fill background
        setBackground(Color.WHITE); // set its background color

        // Draw the grid-lines
        g.setColor(Color.LIGHT_GRAY);
        for (int row = 1; row < ROWS; ++row) {
            g.fillRoundRect( x: 0,  y: CELL_SIZE * row - GRID_WIDHT_HALF,
                    width: CANVAS_WIDTH-1, GRID_WIDTH, GRID_WIDTH, GRID_WIDTH);
        }
        for (int col = 1; col < COLS; ++col) {
            g.fillRoundRect( x: CELL_SIZE * col - GRID_WIDHT_HALF,  y: 0,
                    GRID_WIDTH,  height: CANVAS_HEIGHT-1, GRID_WIDTH, GRID_WIDTH);
        }
```

```java
        // Draw the Seeds of all the cells if they are not empty
        // Use Graphics2D which allows us to set the pen's stroke
        Graphics2D g2d = (Graphics2D)g;
        g2d.setStroke(new BasicStroke(SYMBOL_STROKE_WIDTH, BasicStroke.CAP_ROUND,
                BasicStroke.JOIN_ROUND));  // Graphics2D only
        for (int row = 0; row < ROWS; ++row) {
            for (int col = 0; col < COLS; ++col) {
                int x1 = col * CELL_SIZE + CELL_PADDING;
                int y1 = row * CELL_SIZE + CELL_PADDING;
                if (board[row][col] == Play2Players.Seed.CROSS) {
                    g2d.setColor(Color.RED);
                    int x2 = (col + 1) * CELL_SIZE - CELL_PADDING;
                    int y2 = (row + 1) * CELL_SIZE - CELL_PADDING;
                    g2d.drawLine(x1, y1, x2, y2);
                    g2d.drawLine(x2, y1, x1, y2);
                } else if (board[row][col] == Play2Players.Seed.NOUGHT) {
                    g2d.setColor(Color.BLUE);
                    g2d.drawOval(x1, y1, SYMBOL_SIZE, SYMBOL_SIZE);
                }
            }
        }
```

```
        // Print status-bar message
        if (currentState == Play2Players.GameState.PLAYING) {
            statusBar.setForeground(Color.BLACK);
            if (currentPlayer == Play2Players.Seed.CROSS) {
                statusBar.setText("Lượt của "+Player1Name);
            } else {
                statusBar.setText("Lượt của "+Player2Name);
            }
        } else if (currentState == Play2Players.GameState.DRAW) {
            statusBar.setForeground(Color.RED);
            statusBar.setText("Hòa rồi! Click chuột để chơi lại");
        } else if (currentState == Play2Players.GameState.CROSS_WON) {
            statusBar.setForeground(Color.RED);
            statusBar.setText(Player1Name+" thắng rồi! Click chuột để chơi lại");
        } else if (currentState == Play2Players.GameState.NOUGHT_WON) {
            statusBar.setForeground(Color.RED);
            statusBar.setText(Player2Name+" thắng rồi! Click chuột để chơi lại");
        }
    }
}
```

➢ DrawCanvas kế thừa lớp JPanel - là một container chung và gọn nhẹ của lớp Container. Từ đó ta có thể tạo một bề mặt dùng để vẽ theo ý thích của mình thông qua paintComponent().

➢ paintComponent(): Bất cứ đoạn code nào bạn dùng để vẽ ra thứ gì cũng được đặt trong phương thức này, được override từ lớp JPanel. Với phương thức này, ta vẽ ra được các thành phần trong bảng trò chơi Tic-tac-toe.

- PlayWithAI:

```
if (currentPlayer == Seed.NOUGHT && currentState == GameState.PLAYING) {
   if(GameBot == Bot.EASY_BOT) {
      EasyBot botRun = new EasyBot();
      String run = botRun.getPosFrBrd(board);
      String[] splStr = run.split( regex: " ");
      rowSelected = Integer.parseInt(splStr[0]);
      colSelected = Integer.parseInt(splStr[1]);
   }
   else if(GameBot == Bot.HEURISTIC_BOT){
      HeuristicBot botRun = new HeuristicBot(ROWS,COLS, Seed.NOUGHT, Seed.CROSS);
      String run = botRun.getPoint(board);
      String[] splStr = run.split( regex: " ");
      rowSelected = Integer.parseInt(splStr[0]);
      colSelected = Integer.parseInt(splStr[1]);
   }

   if (rowSelected >= 0 && rowSelected < ROWS && colSelected >= 0
         && colSelected < COLS && board[rowSelected][colSelected] == Seed.EMPTY) {
      //Luu dòng cột đã chon của máy
      rowBotPreSelected = rowSelected;
      colBotPreSelected = colSelected;

      board[rowSelected][colSelected] = currentPlayer; // Make a move
      updateGame(currentPlayer, rowSelected, colSelected); // update state
      // Switch player
      currentPlayer = (currentPlayer == Seed.CROSS) ? Seed.NOUGHT : Seed.CROSS;
   }
}
```

➢ playGame(): phương thức này được override từ lớp Play2Players. Phương thức chơi không đổi nhưng người chơi 1 sẽ đánh với người chơi 2 là máy thuộc lớp EasyBot hay HeuristicBot.

- HeuristicBot:

```
static int dong, cot;
static Seed bot;
static Seed player; // kiểu của bot, người chơi, test với int, có thể thay lại thành enum
static int[] mangTC = new int[]{0, 10, 600, 3500, 40000000, 70000, 1000000};
static int[] mangPN = new int[]{0, 7, 700, 4000, 10000, 67000, 500000};
static long MAX_INT = 100000000;
// int dongg, int cott, enum bott, enum playerr
public HeuristicBot(int dongg, int cott, Seed Bott, Seed Playerr){
   dong = dongg;
   cot = cott;
   bot = Bott;
   player = Playerr;
}
```

➢ dong,cot: chỉ dòng cột mà máy có thể tương tác.

➢ bot, player: kiểu của bot, người chơi.

➢ mangTC: mảng chứa các điểm tấn công của Bot.

➢ mangPN: mảng chứa các điểm phòng ngự của Bot.

```
public HeuristicBot(int dongg, int cott, Seed Bott, Seed Playerr){
    dong = dongg;
    cot = cott;
    bot = Bott;
    player = Playerr;
}
```

➢ HeuristicBot(): hàm tạo của HeuristicBot.

```
public static String getPoint(Seed[][] board){
   long checkTC = 0;
   long checkPT = 0;
   long max = 0;
   String vTri = new String();
   List<String> list = new ArrayList<>();
   System.out.println();
   for (int i = 0; i < dong; i++) {
      for (int ii = 0; ii < cot; ii++) {
         if (board[i][ii] == Seed.EMPTY) {
            checkTC = CheckDoc(ii, i, board, bot) + CheckNgang(ii, i, board, bot) + CheckCheoPhai(i, ii, board, bot) + CheckCheoTrai(i, ii, board, bot);
            checkPT = PTDoc(ii, i, board, player) + PTNgang(ii, i, board, player) + PTPhai(i, ii, board, player) + PTTrai(i, ii, board, player);
            long tmp = checkPT + checkTC;
            if (tmp > max) {
               list = new ArrayList<String>();
               max = tmp;
               vTri = i + " " + ii;
               list.add(vTri);
            }
            if (tmp == max) {
               vTri = i + " " + ii;
               list.add(vTri);
            }
         }
      }
   }
   Random rand = new Random();
   String s = list.get(rand.nextInt(list.size()));
   return s;
}
```

➢ getPoint(): Duyệt toàn bộ bàn cờ. Nếu có ô trống nào thì sẽ check tính tấn công và phòng thủ của nó, từ đó lập ra một danh sách vị trí thích hợp có thể đánh dấu được nhằm tạo điều kiện thắng lợi và giảm bớt xác xuất thua lớn nhất có thể. Hàm trả về là một vị trí ô có trong danh sách (lấy ngẫu nhiên trong danh sách).

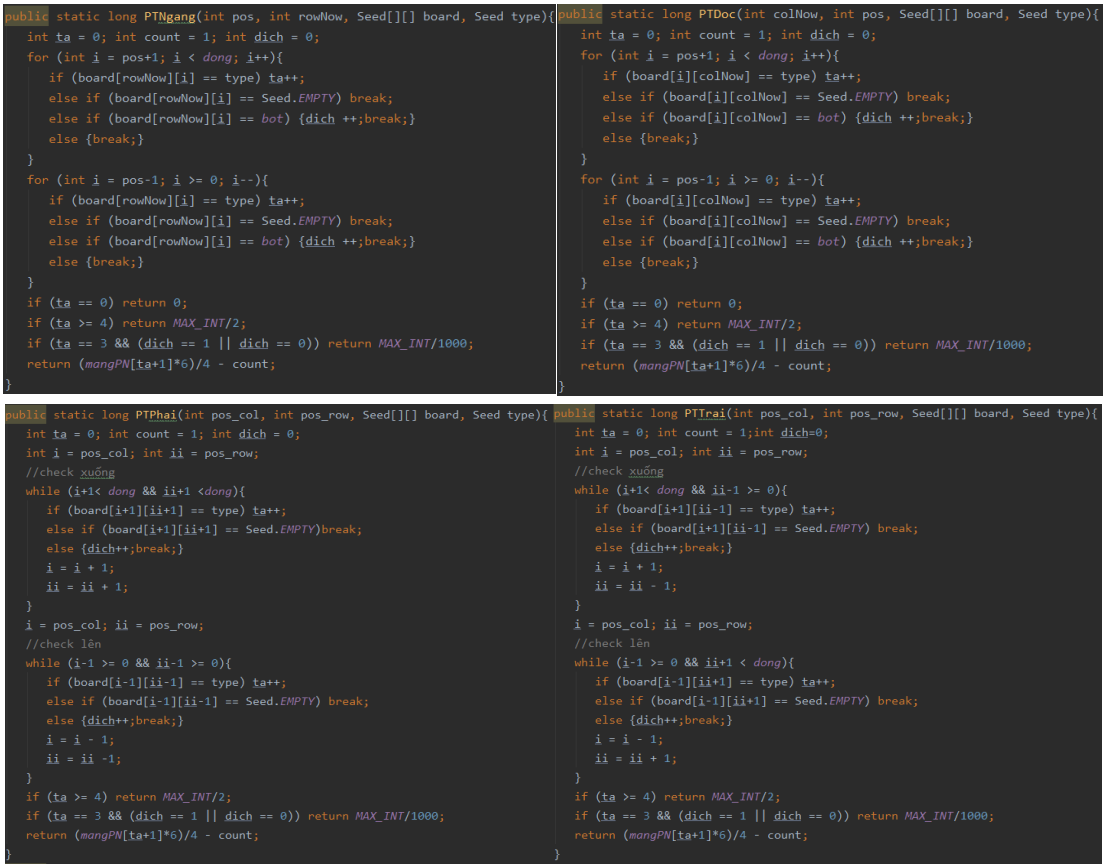
```
public static long CheckNgang(int pos, int rowNow, Seed[][] board, Seed type){
   int ta = 0; int count = 0; int dich = 0;
   boolean flag = false;
   for (int i = pos+1; i < dong; i++){
      if (board[rowNow][i] == type) ta++;
      else if (board[rowNow][i] == Seed.EMPTY && flag == false){
         count++; flag = true; break;
      }
      else if (board[rowNow][i] == player) {dich ++;break;}
      else {break;}
   }
   flag = false;
   for (int i = pos-1; i >= 0; i--){
      if (board[rowNow][i] == type) ta++;
      else if (board[rowNow][i] == Seed.EMPTY && flag == false){
         count++; flag = true; break;
      }
      else if (board[rowNow][i] == player) {dich ++;break;}
      else {break;}
   }
   if (ta == 0) return 0;
   if (ta == 3 && dich == 2 && (count == 0 || count == 1)) return 0;
   if (ta == 2 && dich == 2 && (count == 0 || count == 1 || count == 2)) return 0;
   if (ta <= 3 && dich == 1 && count == 0) return 0;
   if (ta == 3 && (dich == 0 || dich == 1)) return MAX_INT/250;
   if (ta >= 4) return MAX_INT;
   return (mangTC[ta]*3)/2 - count*100;
}
```

```
public static long CheckDoc(int colNow, int pos, Seed[][] board, Seed type){
   int ta = 0; int count = 0; int dich = 0;
   boolean flag = false;
   for (int i = pos+1; i < dong; i++){
      if (board[i][colNow] == type) ta++;
      else if (board[i][colNow] == Seed.EMPTY && flag == false){
         count++; flag = true; break;
      }
      else if (board[i][colNow] == player) {dich ++;break;}
      else {break;}
   }
   flag = false;
   for (int i = pos-1; i >= 0; i--){
      if (board[i][colNow] == type) ta++;
      else if (board[i][colNow] == Seed.EMPTY && flag == false){
         count++;flag = true; break;
      }
      else if (board[i][colNow] == player) {dich ++;break;}
      else {break;}
   }
   if (ta == 0) return 0;
   if (ta == 3 && dich == 2 && (count == 0 || count == 1)) return 0;
   if (ta == 2 && dich == 2 && (count == 0 || count == 1 || count == 2)) return 0;
   if (ta <= 3 && dich == 1 && count == 0) return 0;
   if (ta == 3 && (dich == 0 || dich == 1)) return MAX_INT/250;
   if (ta >= 4) return MAX_INT;
   return (mangTC[ta]*3)/2 - count*100;
}
```

```
public static long CheckCheoPhai(int pos_col, int pos_row, Seed[][] board, Seed type){
   int ta = 0; int count = 0; int dich = 0;
   boolean flag = false;
   int i = pos_col; int ii = pos_row;
   //check xuống
   while (i+1< dong && ii+1 <dong){
      if (board[i+1][ii+1] == type) ta++;
      else if (board[i+1][ii+1] == Seed.EMPTY && flag == false){ count++; flag = true; break; }
      else if (board[i+1][ii+1] == player) {dich ++; break;}
      else {break;}
      i = i + 1; ii = ii + 1;
   }
   i = pos_col; ii = pos_row;
   flag = false;
   //check lên
   while (i-1 >= 0 && ii-1 >= 0){
      if (board[i-1][ii-1] == type) ta++;
      else if (board[i-1][ii-1] == Seed.EMPTY && flag == false) { count++; flag = true; break; }
      else if (board[i-1][ii-1] == player) {dich ++; break;}
      else {break;}
      i = i - 1; ii = ii -1;
   }
   if (ta == 0) return 0;
   if (ta == 3 && dich == 2 && (count == 0 || count == 1)) return 0;
   if (ta == 2 && dich == 2 && (count == 0 || count == 1 || count == 2)) return 0;
   if (ta <= 3 && dich == 1 && count == 0) return 0;
   if (ta == 3 && (dich == 0 || dich == 1)) return MAX_INT/250;
   if (ta >= 4) return MAX_INT;
   return (mangTC[ta]*3) - count*100;
}
```

```
public static long CheckCheoTrai(int pos_col, int pos_row, Seed[][] board, Seed type){
   int ta = 0; int count = 0; int dich = 0;
   boolean flag = false;
   int i = pos_col; int ii = pos_row;
   //check xuống
   while (i+1< dong && ii-1 >= 0){
      if (board[i+1][ii-1] == type) ta++;
      else if (board[i+1][ii-1] == Seed.EMPTY && flag == false) { count++; flag = true; break; }
      else if (board[i+1][ii-1] == player) {dich ++;break;}
      else {break;}
      i = i + 1; ii = ii - 1;
   }
   flag = false;
   i = pos_col; ii = pos_row;
   //check lên
   while (i-1 >= 0 && ii+1 < dong){
      if (board[i-1][ii+1] == type) ta++;
      else if (board[i-1][ii+1] == Seed.EMPTY && flag == false) { count++; flag = true; break; }
      else if (board[i-1][ii+1] == player) {dich ++;break;}
      else {break;}
      i = i - 1; ii = ii + 1;
   }
   if (ta == 0) return 0;
   if (ta == 3 && dich == 2 && (count == 0 || count == 1)) return 0;
   if (ta == 2 && dich == 2 && (count == 0 || count == 1 || count == 2)) return 0;
   if (ta <= 3 && dich == 1 && count == 0) return 0;
   if (ta == 3 && (dich == 0 || dich == 1)) return MAX_INT/250;
   if (ta >= 4) return MAX_INT;
   return (mangTC[ta]*3) - count*100;
}
```

➢ CheckNgang(), CheckDoc(), CheckCheoPhai() và CheckCheoTrai(): Các hàm này dùng để tính điểm tấn công của Bot. Các Hàm trả về là một điểm tấn công thích hợp qua việc xử lý thông tin các ô Bot đã đánh dấu, và các ô người chơi đã đánh dấu trên phạm vi ngang, dọc, chéo.



```
public static long PTNgang(int pos, int rowNow, Seed[][] board, Seed type){
   int ta = 0; int count = 1; int dich = 0;
   for (int i = pos+1; i < dong; i++){
      if (board[rowNow][i] == type) ta++;
      else if (board[rowNow][i] == Seed.EMPTY) break;
      else if (board[rowNow][i] == bot) {dich ++;break;}
      else {break;}
   }
   for (int i = pos-1; i >= 0; i--){
      if (board[rowNow][i] == type) ta++;
      else if (board[rowNow][i] == Seed.EMPTY) break;
      else if (board[rowNow][i] == bot) {dich ++;break;}
      else {break;}
   }
   if (ta == 0) return 0;
   if (ta >= 4) return MAX_INT/2;
   if (ta == 3 && (dich == 1 || dich == 0)) return MAX_INT/1000;
   return (mangPN[ta+1]*6)/4 - count;
}
```

```
public static long PTDoc(int colNow, int pos, Seed[][] board, Seed type){
   int ta = 0; int count = 1; int dich = 0;
   for (int i = pos+1; i < dong; i++){
      if (board[i][colNow] == type) ta++;
      else if (board[i][colNow] == Seed.EMPTY) break;
      else if (board[i][colNow] == bot) {dich ++;break;}
      else {break;}
   }
   for (int i = pos-1; i >= 0; i--){
      if (board[i][colNow] == type) ta++;
      else if (board[i][colNow] == Seed.EMPTY) break;
      else if (board[i][colNow] == bot) {dich ++;break;}
      else {break;}
   }
   if (ta == 0) return 0;
   if (ta >= 4) return MAX_INT/2;
   if (ta == 3 && (dich == 1 || dich == 0)) return MAX_INT/1000;
   return (mangPN[ta+1]*6)/4 - count;
}
```

```
public static long PTPhai(int pos_col, int pos_row, Seed[][] board, Seed type){
   int ta = 0; int count = 1; int dich = 0;
   int i = pos_col; int ii = pos_row;
   //check xuống
   while (i+1< dong && ii+1 <dong){
      if (board[i+1][ii+1] == type) ta++;
      else if (board[i+1][ii+1] == Seed.EMPTY)break;
      else {dich++;break;}
      i = i + 1;
      ii = ii + 1;
   }
   i = pos_col; ii = pos_row;
   //check lên
   while (i-1 >= 0 && ii-1 >= 0){
      if (board[i-1][ii-1] == type) ta++;
      else if (board[i-1][ii-1] == Seed.EMPTY) break;
      else {dich++;break;}
      i = i - 1;
      ii = ii -1;
   }
   if (ta >= 4) return MAX_INT/2;
   if (ta == 3 && (dich == 1 || dich == 0)) return MAX_INT/1000;
   return (mangPN[ta+1]*6)/4 - count;
}
```

```
public static long PTTrai(int pos_col, int pos_row, Seed[][] board, Seed type){
   int ta = 0; int count = 1;int dich=0;
   int i = pos_col; int ii = pos_row;
   //check xuống
   while (i+1< dong && ii-1 >= 0){
      if (board[i+1][ii-1] == type) ta++;
      else if (board[i+1][ii-1] == Seed.EMPTY) break;
      else {dich++;break;}
      i = i + 1;
      ii = ii - 1;
   }
   i = pos_col; ii = pos_row;
   //check lên
   while (i-1 >= 0 && ii+1 < dong){
      if (board[i-1][ii+1] == type) ta++;
      else if (board[i-1][ii+1] == Seed.EMPTY) break;
      else {dich++;break;}
      i = i - 1;
      ii = ii + 1;
   }
   if (ta >= 4) return MAX_INT/2;
   if (ta == 3 && (dich == 1 || dich == 0)) return MAX_INT/1000;
   return (mangPN[ta+1]*6)/4 - count;
}
```

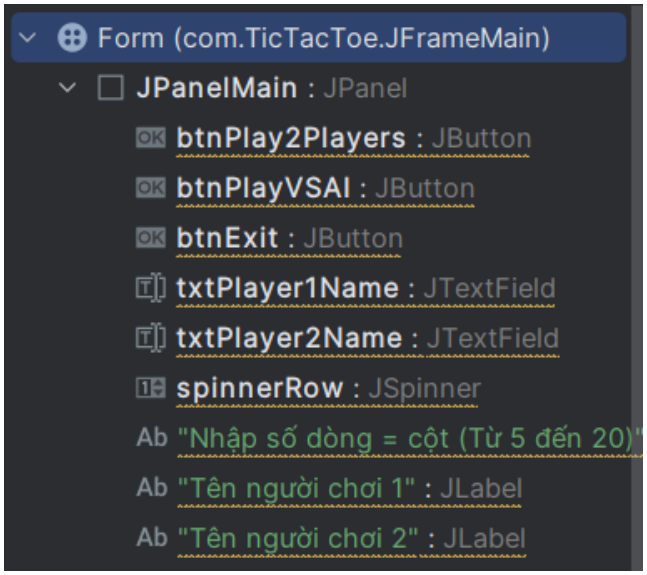
➢ PTNgang(), PTDoc(), PTPhai() và PTTrai(): Các hàm này dùng để tính điểm phòng ngự của Bot. Các Hàm trả về là một điểm phòng ngự thích hợp qua việc xử lý thông tin các ô Bot đã đánh dấu, và các ô người chơi đã đánh dấu trên phạm vi ngang, dọc, chéo.

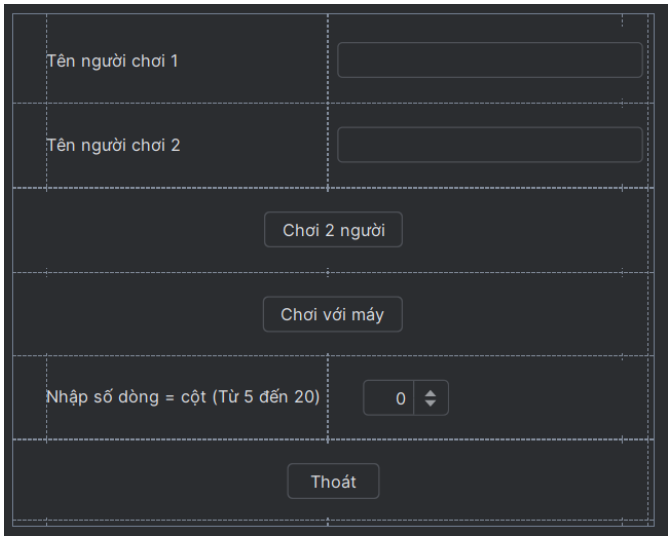
- EasyBot:

```
public String getPosFrBrd(Play2Players.Seed[][] board){
    String vTri = new String();
    int pos1, pos2;
    do {
        Random rand = new Random();
        pos1 = rand.nextInt( bound: Play2Players.COLS - 2) + 1;
        pos2 = rand.nextInt( bound: Play2Players.COLS - 2) + 1;
        vTri = pos1 + " " + pos2;
    }
    while (board[pos1][pos2] != Play2Players.Seed.EMPTY);
    return vTri;
}
```

➢ getPosFrBrd(): Hàm này thực hiện việc đánh ngẫu nhiên trong bàn cờ. Hàm sẽ lấy một vị trí ngẫu nhiên trong bàn cờ sau đó đánh dấu vào bàn cờ

- JFrameMain:





JFrameMain.form

```
public JFrameMain() {...}
```

➢ JFrameMain(): Hàm này thực hiện việc xử lí các buttons.

```
SpinnerModel spinnerModel =
        new SpinnerNumberModel( value: 10, //initial value
                5, //min
                20, //max
                1);//step
spinnerRow.setModel(spinnerModel);
```

✓ Cài đặt model cho spinner nhập dòng, cột.

```
btnExit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { System.exit( status: 0); }
});
```

✓ btnExit: Nút này dùng để thoát luôn ứng dụng.

```
btnPlayVSAI.addActionListener(new ActionListener() {  DuongHoang673
    @Override  DuongHoang673
    public void actionPerformed(ActionEvent e) {
        String Player1Name = txtPlayer1Name.getText();
        if(Player1Name.isEmpty()){
            JOptionPane.showMessageDialog( parentComponent: null, message: "Bạn chưa nhập tên người chơi!!!");
            return;
        }
        int row = (int)spinnerRow.getValue();
        if( row<5 || row >20){
            JOptionPane.showMessageDialog( parentComponent: null, message: "Bạn phải nhập số dòng, cột nằm trong khoảng" +
                    " giá trị từ 5 đến 20!!!");
            return;
        }
        // Chọn độ khó máy
        Object[] options1 = {"Dễ",
                "Bình thường"};
        int resultLevel = JOptionPane.showOptionDialog( parentComponent: null, message: "Chọn độ khó", title: "Chọn độ khó" +
                "",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE, icon: null,options1,options1[1]);


        if(resultLevel == JOptionPane.YES_OPTION){
            PlayWithAiCaro.GameBot = PlayWithAI.Bot.EASY_BOT;
        }
        else PlayWithAiCaro.GameBot = PlayWithAI.Bot.HEURISTIC_BOT;
        PlayWithAiCaro.newRow =row;
        PlayWithAiCaro heuristicBotCaro = new PlayWithAiCaro(Player1Name);


        jFrame.setVisible(false);
```

✓ btnPlayVSAI: Nút này dùng để chuyển giao diện chính thành vô hình, tạo bảng tùy vào số mình nhập ở SpinnerModel và chơi với máy nhưng yêu cầu phải nhập tên người chơi 1 ở txtPLayer1Name.

```
btnPlay2Players.addActionListener(new ActionListener() {  DuongHoang673 *
    @Override  DuongHoang673 *
    public void actionPerformed(ActionEvent e) {
        String Player1Name = txtPlayer1Name.getText();
        String Player2Name = txtPlayer2Name.getText();
        if(Player1Name.isEmpty() || Player2Name.isEmpty()) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Bạn chưa nhập đủ tên 2 người chơi!!!");
            return;
        }
        int row = (int)spinnerRow.getValue();
        if( row<5 || row >20){
            JOptionPane.showMessageDialog( parentComponent: null, message: "Bạn phải nhập số dòng, cột nằm trong khoảng" +
                    " giá trị từ 5 đến 20!!!");
            return;
        }
        Play2PlayersCaro.newRow = row;
        new Play2PlayersCaro(Player1Name,Player2Name);
        jFrame.setVisible(false);
    }
});
```

✓ btnPLay2PLayers: Nút này dùng để chuyển giao chính thành vô hình, tạo bảng tùy vào số mình nhập ở SpinnerModel và chơi 2 người nhưng yêu cầu phải nhập tên 2 người chơi ở txtPLayer1Name và txtPlayer2Name.

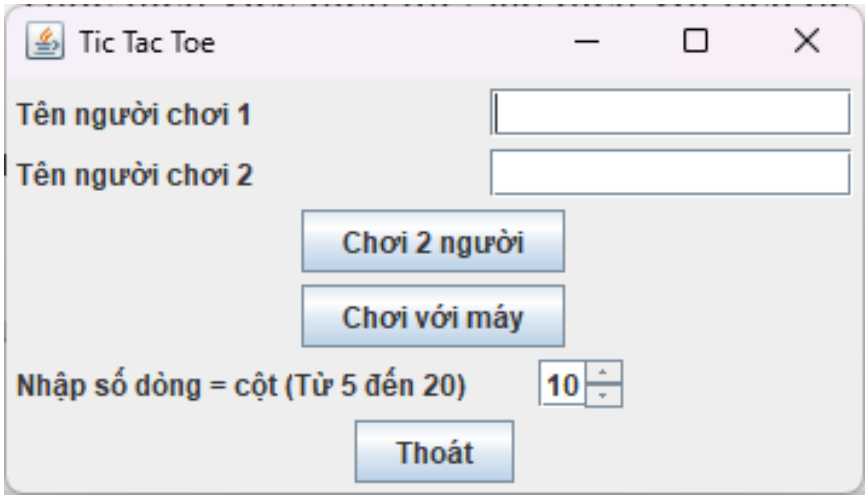
```
public void CreateAndShow(){
    SpinnerModel spinnerModel =
            new SpinnerNumberModel( value: 3, //initial value
                    minimum: 3, //min
                    maximum: 20, //max
                    stepSize: 1);//step
    spinnerRow.setModel(spinnerModel);
    jFrame =new JFrame( title: "Cờ Caro");
    jFrame.setContentPane(new JFrameMain().JPanelMain);
    jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jFrame.pack();
    jFrame.setVisible(true);
    jFrame.setLocationRelativeTo(null);

}
```

➢ CreateAndShow(): Thực hiện việc hiển thị giao diện với tiêu đề “Cờ Caro”.

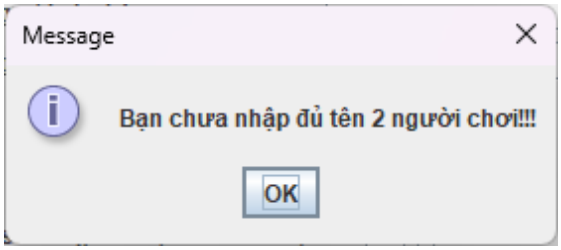
c) Giao diện ứng dụng:

- Giao diện khởi đầu:

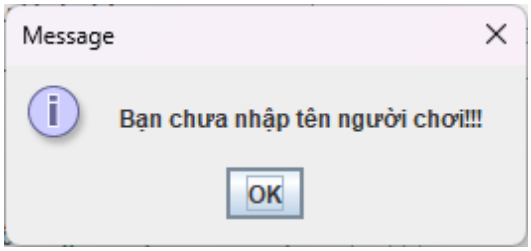


- Thông báo nhập tên:

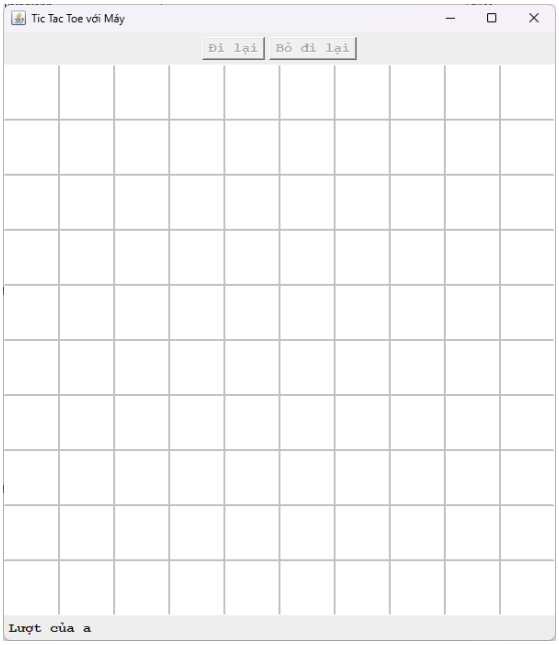
➢ Nếu chọn chơi 2 người:



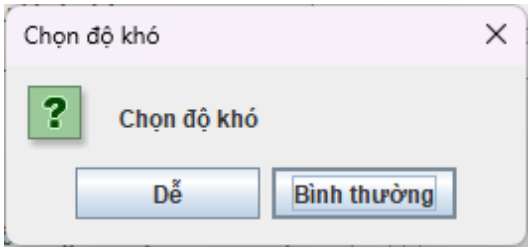
➢ Nếu chọn chơi với máy:



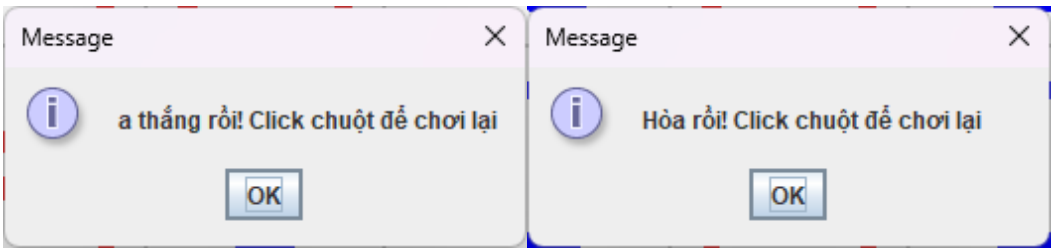
- Giao diện chơi Game:



- Giao diện chọn độ khó Bot:



- Giao diện thông báo thắng hoặc hòa:



IV. KẾT LUẬN:

Từ việc nghiên cứu ngôn ngữ Java, thư viện Java Swing và vận dụng thêm những gì đã học. Nhóm đã tạo ra 1 ứng dụng chơi cờ Ca-rô. Trò chơi sẽ giúp người dùng có những giây phút giải stress và thư giãn sau những giờ làm việc căng thẳng.

V. TÀI LIỆU THAM KHẢO:

- Wikipedia
- https://vietjack.com/java_swing/tong_quan_ve_java_swing.jsp
- https://viettuts.vn/java-swing
- https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaGame_TicTacToe.html?fbclid=IwAR1S5K5EJCq_LjkFuo62Qf9v0YbW0oFnfQHKMKv6i9f4S3hmcFctTaJ-wiA