**TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**LỚP CNTTC**

---------oOo---------



**BÁO CÁO ĐỒ ÁN**

**MÔN : CÔNG NGHỆ JAVA**

**CHỦ ĐỀ : BÁO CÁO GAME CỜ CARO**

**GVHD : LÊ TUẤN**

**Nhóm sinh viên:**

1. **Phan Huỳnh Văn Ngọc 43.01.104.116**
2. **Nguyễn Thanh Huy Hoàng 43.01.104.056**
3. **Trần Thiên Phúc 43.01.104.134**
4. **GIỚI THIỆU:**

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 **O**, người kia dùng ký hiệu **X**, lần lượt điền ký hiệu của mình vào các ô. Người thắng là người 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.

1. **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. Chúng tôi hi vọng trò chơi sẽ mang lại những phút giây thư giãn để xả tress.

1. **NỘI DUNG:**
2. **Các chức năng trong ứng dụng:**

Trò chơi Ca-rô gồm các chức năng sau:

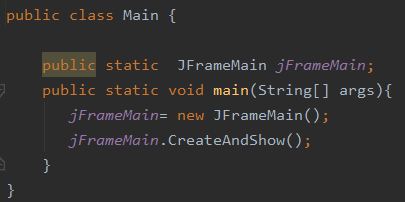
* **Nhập tên người chơi:** Người chơi cần nhập tên vào để chơi game, nếu không nhập tên sẽ không vào được giao diện chơi game.
* **Chọn số ô trong bàn cờ Caro:** người dùng có thể chọn số ô trong bàn cờ Ca-rô bằng cách nhập vào số dòng=cột với giá trị thấp nhất là 3 (9 ô) và giá trị cao nhất là 20 (400 ô). Nếu chọn 3 thì sẽ chơi Tic Tac Toe, 4 là game Connect 4, còn lớn hơn 5 là chơi Caro bình thường.
* **Campaign:** Là chế độ chơi thử thách, người chơi sẽ nhập tên và đánh liên tiếp 7 màn. Màn sau khó hơn màn trước, chỉ cần thua 1 màn thì sẽ chơi lại từ đầu.
* **Stage 1:** Dễ nhất. Mình được đi trước và lượt đầu được đánh 3 lần và Bot sẽ đánh ngẫu nhiên trong lượt đầu tiên của Bot. Các lượt sau thì Bot sẽ dùng thuật giải Heuristic để đánh.
* **Stage 2:** Mình được đi trước và lượt đầu tiên được đánh 2 lần và Bot sẽ đánh ngẫu nhiên trong lượt đầu tiên của Bot. Các lượt sau thì Bot sẽ dùng thuật giải Heuristic để đánh.
* **Stage 3:** Mình được đi trước và Bot sẽ đánh ngẫu nhiên trong lượt đầu tiên của Bot. Các lượt sau thì Bot sẽ dùng thuật giải Heuristic để đánh.
* **Stage 4:** Mình được đi trước và Bot hoàn toàn dùng thuật giải Heuristic để đánh.
* **Stage 5:** Giống Stage 4 nhưng Bot được đi trước.
* **Stage 6:** Bot được đi trước và trong lượt đầu sẽ được đánh 2 lần.
* **Stage 7:** Bot được đi trước và trong lượt đầu sẽ được đánh 3 lần.
* **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:** Sử dụng thuật giải Heuristic để viết nên Bot khá thông minh.
* **MOD chơi chấp trước 1 bước:** chơi với chế độ lượt đầu tiên của 1 trong 2 người chơi (người chơi 2 có thể là máy) được đánh 2 ký hiệu trong 1 lượt. Trong chế độ này lại có 2 lựa chọn là:
* **Đánh liền kề:** Người chơi hoặc máy chỉ được đánh kí hiệu vào 2 ô kề nhau ở lượt đầu tiên. Nếu đánh 2 ô không liền kề thì sẽ chặn lại và xuất thông báo.
* **Đánh không liền kề:** Người chơi hoặc máy chỉ được đánh 2 kí hiệu vào 2 ô không liền kề nhau ở lượt đầu tiên. Nếu đánh vào 2 ô liền kề thì sẽ chặn lại và xuất thông báo.
* **Chế độ đi lại và bỏ đi lại:** trong 1 bàn ca-rô (trừ Campaign), 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.
* **Xem High Score theo tên người chơi:** Xuất thông tin của các người chơi gồm tên, số lần thắng, lần thua.

1. **Phân tích ứng dụng:**
2. **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 nếu xài bàn cờ 5 dòng 5 cột trở lên.
* Play2PlayersConnect4: Kế thừa từ lớp Play2PLayersCaro, xử lý thuật toán thắng thua nếu xài bàn cờ 4 dòng 4 cột.
* 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 nếu xài bàn cờ 5 dòng 5 cột trở lên.
* PlayWithAiConnect4: Kế thừa từ lớp PlayWithAiCaro, xử lý thuật toán thắng thua nếu xài bàn cờ 4 dòng 4 cột.
* ModChap1Nuoc2Nguoi: Kế thừa từ lớp Play2Players, thực hiện việc chơi 2 người với chế độ chấp trước 1 bước.
* ModChap1Nuoc2NguoiCaro: Kế thừa từ lớp ModChap1Nuoc2Nguoi, xử lý thuật toán thắng thua nếu xài bàn cờ 5 dòng 5 cột trở lên.
* ModChap1Nuoc2NguoiConnect4: Kế thừa từ lớp ModChap1Nuoc2NguoiCaro, xử lý thuật toán thắng thua nếu xài bàn cờ 4 dòng 4 cột.
* ModChap1NuocVSMay: Kế thừa từ lớp Play2WithAi, thực hiện việc chơi với máy với chế độ chấp trước 1 bước.
* ModChap1NuocVSMayCaro: Kế thừa từ lớp ModChap1NuocVSMay, xử lý thuật toán thắng thua nếu xài bàn cờ 5 dòng 5 cột trở lên.
* ModChap1NuocVSMayConnect4: Kế thừa từ lớp ModChap1NuocVSMayCaro, xử lý thuật toán thắng thua nếu xài bàn cờ 4 dòng 4 cột.
* EasyBot: Xử lý thuật toán Bot dễ.
* Campaign: Thực hiện chế độ khởi tạo Campaign, mặt định của số dòng cột trong bàn cờ Ca-rô là 18.
* CampaignStage1: Kế thừa từ lớp PlayWithAiCaro, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* CampaignStage2: Kế thừa từ lớp CampaignStage1, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* CampaignStage3: Kế thừa từ lớp CampaignStage2, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* CampaignStage4: Kế thừa từ lớp CampaignStage3, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* CampaignStage5: Kế thừa từ lớp CampaignStage4, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* CampaignStage6: Kế thừa từ lớp CampaignStage5, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* CampaignStage7: Kế thừa từ lớp CampaignStage6, xử lý việc chơi Ca-rô với máy theo chế độ Campaign.
* GetAndSetHighScore: quản lý dữ liệu thắng thua của các người chơi.
* JFrameMain: Hiển thị giao diện khởi đầu. Thực hiện việc hiển thị các giao diện khác thông qua một số buttons.
* HighScoreForm: Hiển thị giao diện xem lịch sử và thực hiện việc lấy dữ liệu từ lớp GetAndSetHighScore đưa vào giao diện khi giao diện ấy được gọi lên.

1. **Nội dung Class:**

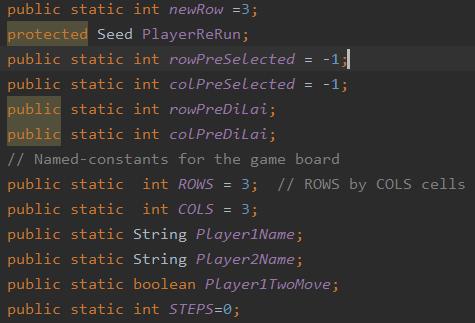
* Các lớp trong gói Java Swing được sử dụng cho giao diện của ứng dụng:
* JFrame: Là khung của ứng dụng, chứa các thành phần như labels, buttons, textfields, tables,… để tạo ra một GUI. JFrame của Swing là một phiên bản kế thừa từ java.awt.Frame mà bổ sung các hỗ trợ cho cấu trúc thành phần JFC/Swing.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



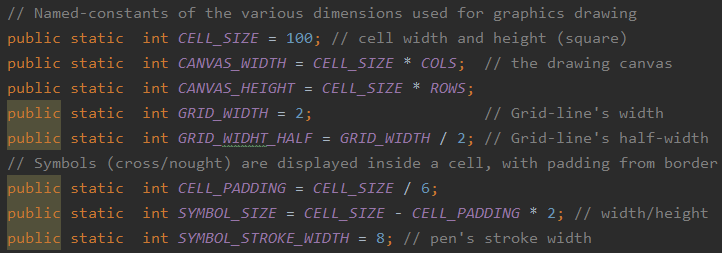
* Run Main để khởi động ứng dụng. Khi khởi động sẽ hiển thị giao diện khởi đầu của JFrameMain.
* Play2Players:



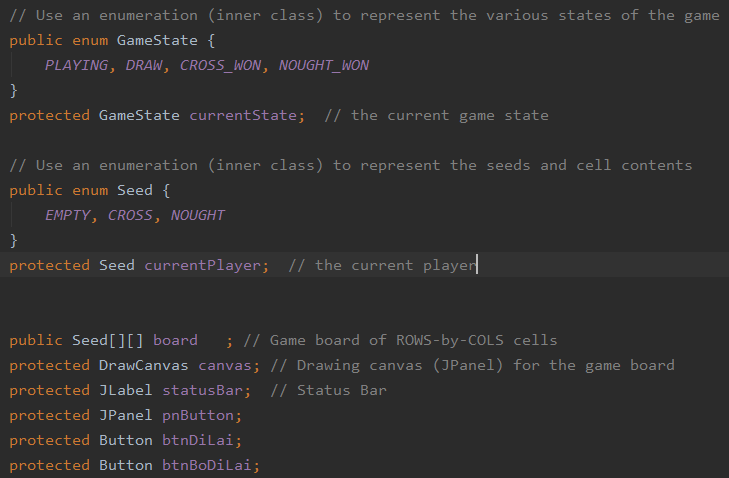
* Kế thừa lớp JFrame.



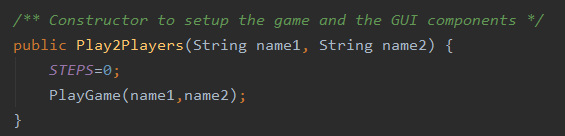
* newRow: chỉ số dòng cột của bảng Caro, có mặc định bằng 3.
* PlayerReRun: chỉ người chơi cần đi lại.
* rowPreSelected, colPreSelected: chỉ vị trí của một ô trong bảng đã đánh dấu trong lượt trước của người chơi, được sử dụng để tìm vị trí đi trước đó của người chơi.
* rowPreDiLai, colPreDiLai: chỉ 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, biến boolean Player1TwoMove kiểm tra coi có Mod chấp trước 1 bước không và biến STEPS chỉ lượt đánh đầu của người chơi.



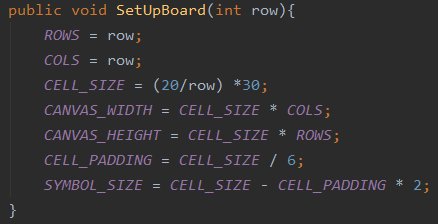
* Khai báo hằng số dùng cho giao diện đồ họa: CELL\_SIZE chỉ độ rộng và cao của mỗi ô vuông trong bảng, CANVAS\_WIDTH và CANVAS\_HEIGHT chỉ độ rộng và độ cao của khoảng trống hình chữ nhật (nơi ứng dụng có thể vẽ hoặc bẫy các sự kiện đầu vào từ người dùng), GRID\_WIDTH chỉ độ rộng đường kẽ giữa các ô 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.



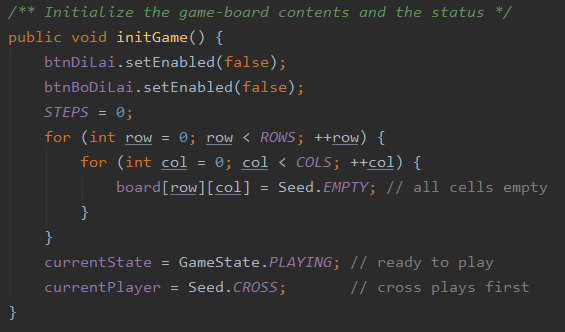
* GameState chỉ trạng thái của game, được định nghĩa bởi enum với 4 giá trị: PLAYING (thắng), DRAW (hòa), CROSS\_WON (X thắng), NOUGHT\_WON (O thắng).
* currentState: Chỉ trạng thái hiện tại của game.
* Seed đại diện cho thứ ô đang chứa, được định nghĩa từ enum với 3 giá trị: EMPTY (ô trống), CROSS (ô 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.



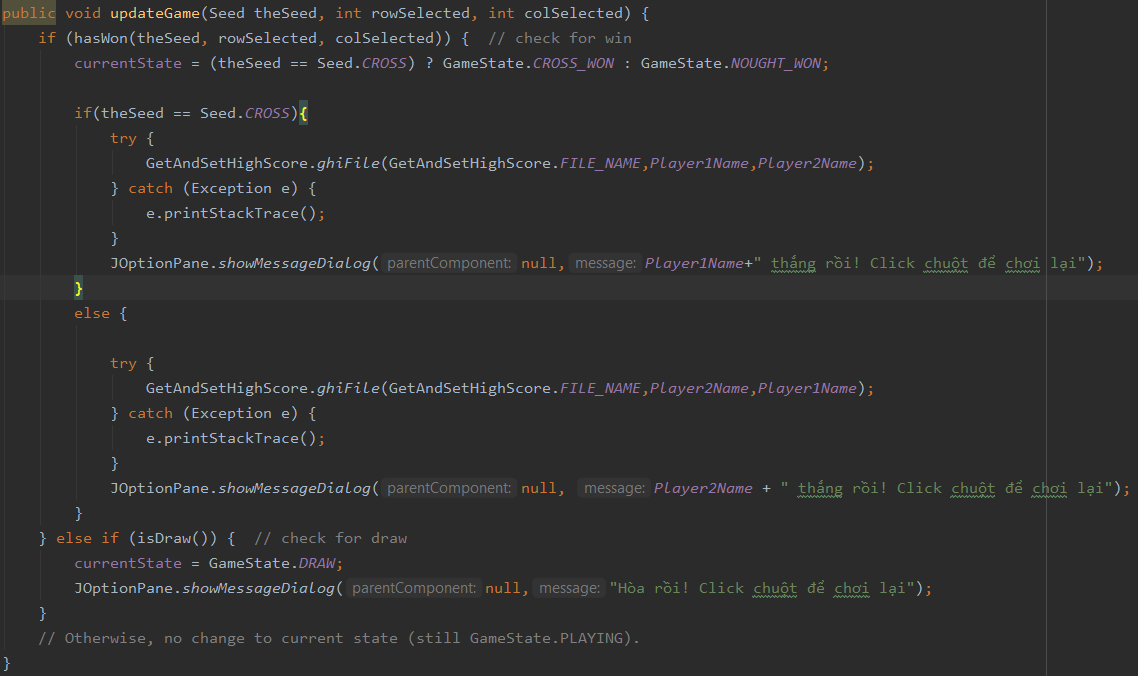
* Khởi tạo để cài đặt trò chơi và thành phần GUI.



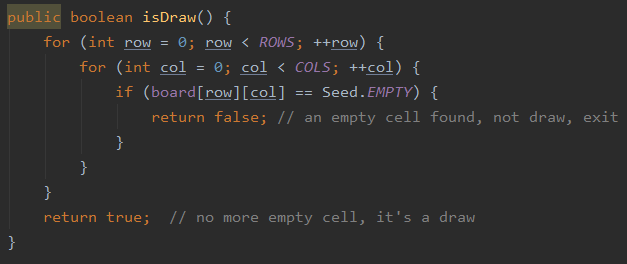
* SetUpBoard(): dùng để set lại kích cỡ bàn cờ, từ việc nhập số dòng/cột ở giao diện khởi đầu rồi set lại bàn cờ sao cho hợp lý.



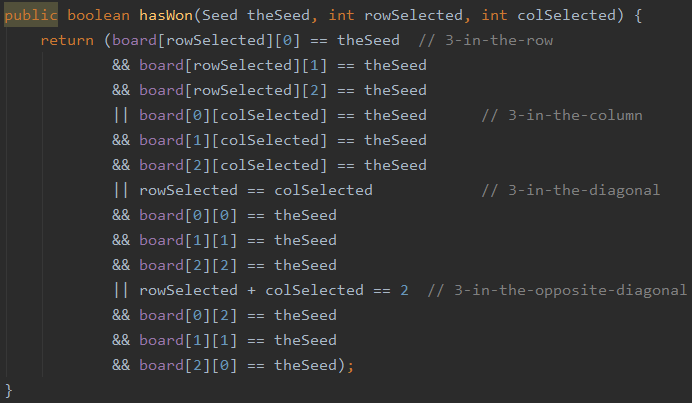
* initGame(): Hàm này dùng để khởi tạo nội dung bảng trò chơi và trạng thái của game. Set button đi lại và bỏ đi lại tạm không sư dụng được, tạo STEPS = 0, làm tất cả ô vuông đều trống, current state sẽ là PLAYING và người chơi ký hiệu X sẽ đánh lượt đầu tiên.



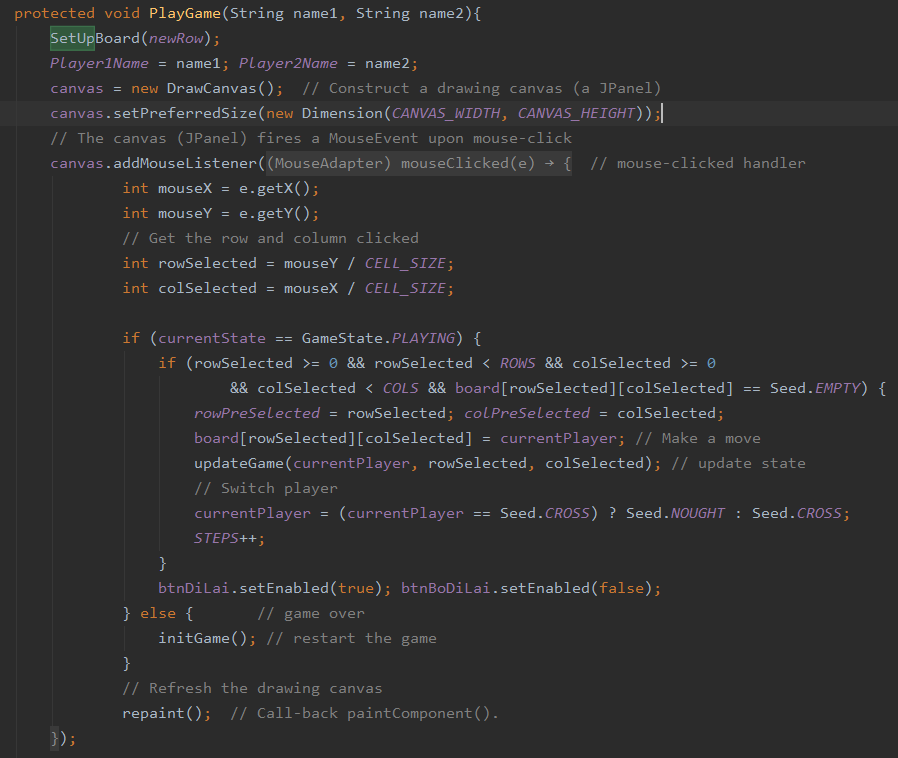
* updateGame(): hàm dùng để cập nhật trạng thái của game. Nếu vẫn còn chơi được thì không thay đổi gì. Tuy nhiên, nếu một bên đạt điều kiện thắng (X hoặc O) thì chuyển trạng thái CROSS\_WON hay NOUGHT\_WON tùy theo người thắng là ai đồng thời ghi lại lần thắng thua của 2 người đó trong lịch sử thông qua hàm ghiFile() của lớp GetAndSetHighScore. Nếu trên bảng không còn ô vuông trống mà không có ai thắng cả thì chuyển trạng thái DRAW.



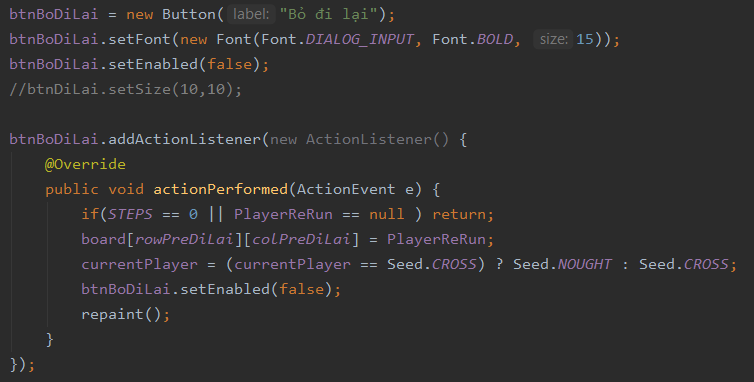
* isDraw(): hàm boolean kiểm tra trận đấu có hòa không. Nếu đánh hết tất cả các ô trong bảng mà không phân được người thắng thì trả về true.

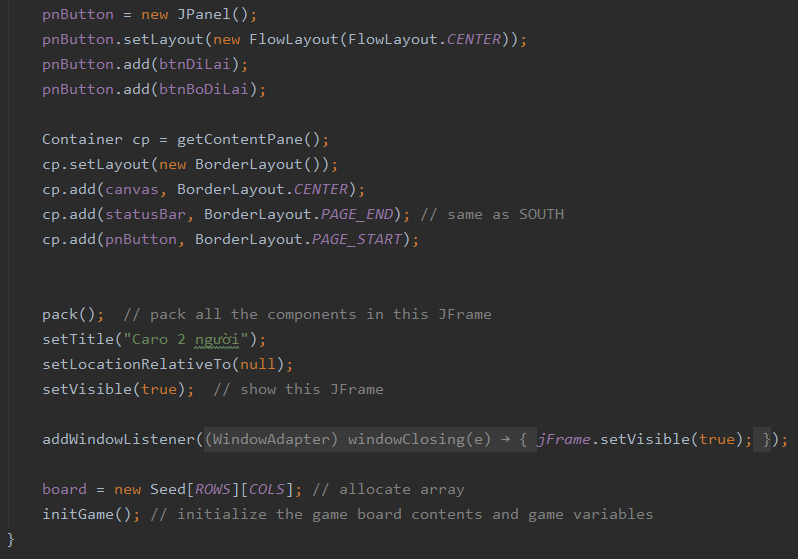


* HasWon(): Là hàm boolean dùng để kiểm tra coi có người chơi nào đạt diều kiện thắng chưa. Hàm dùng cho bàn cờ 9 ô, nếu người chơi đánh được 3 kí hiệu theo dãy ngang (chéo hay dọc đều được) thì sẽ trả về true.

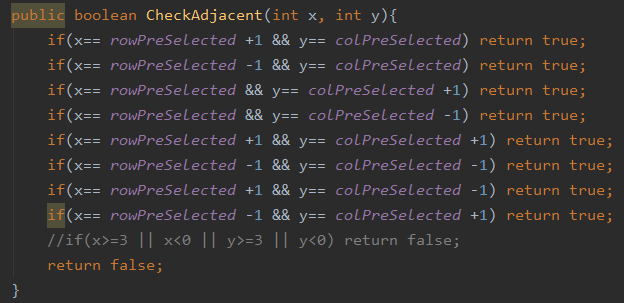




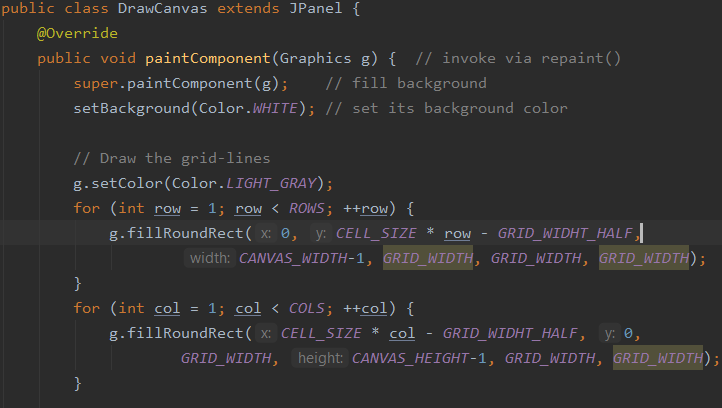


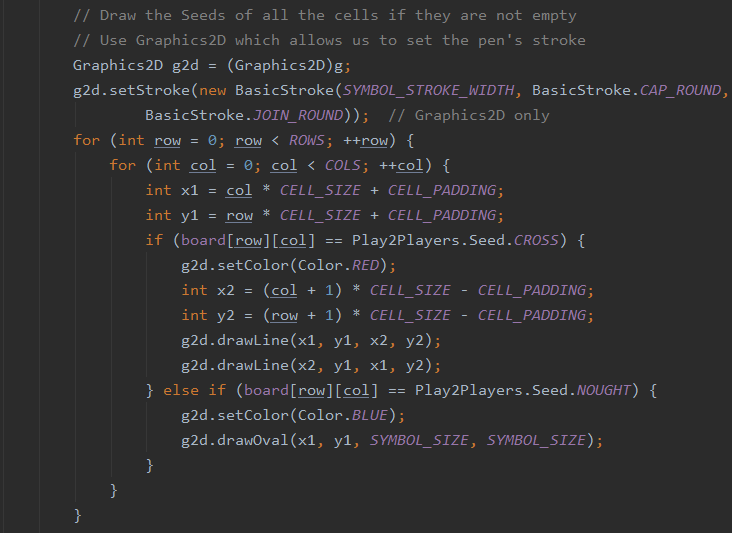


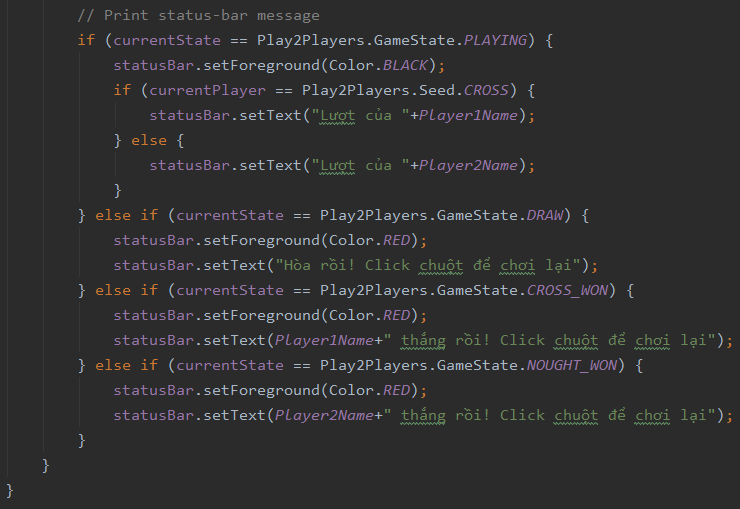
* PlayGame(): Hàm này dùng để thực hiện việc chơi game. Đầu tiên hàm sẽ tạo giao diện bảng trò chơi. Người chơi sẽ nhấp chuột vào một ô vuông để đánh kí hiệu, sau đó đến người chơi 2 (hoặc máy). Khi đang đánh thì 2 button đi lại sẽ sáng lên và có thể sử dụng được. Chơi tới khi nào 1 trong 2 thắng hoặc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



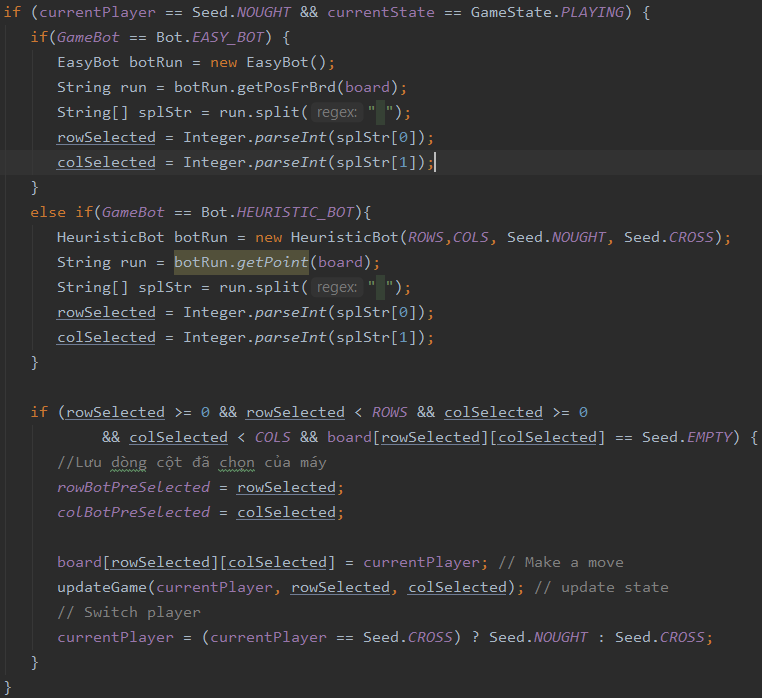
* CheckAdjacent(): hàm kiểm tra 2 ô được đánh kí hiệu trong bảng có liền kề ko.
* DrawCanvas:



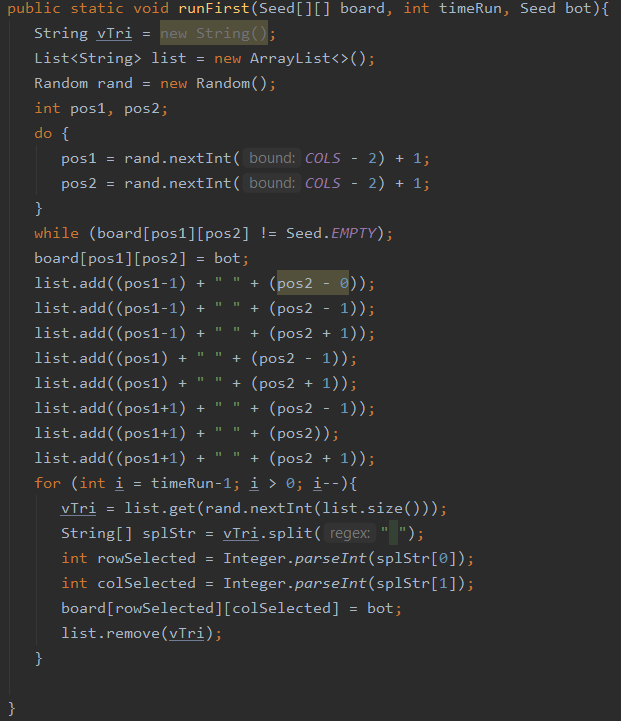




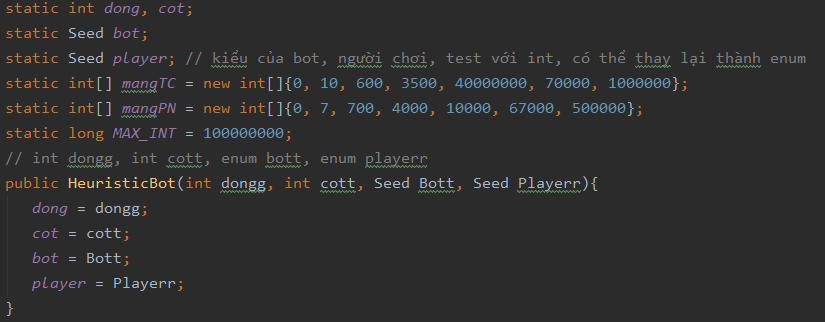
* DrawCanvas kế thừa lớp JPanel - là một container chung và gọn nhẹ của lớp Container. Từ đó ta có thể tạo một bề mặt dùng để vẽ theo ý thích của mình thông qua paintComponent().
* paintComponent(): Bất cứ đoạn code nào bạn dùng để vẽ ra thứ gì cũng được đặt trong phương thức này, được override từ lớp JPanel. Với phương thức này, ta vẽ ra được các thành phần trong bảng trò chơi Tic-tac-toe.
* PlayWithAI:



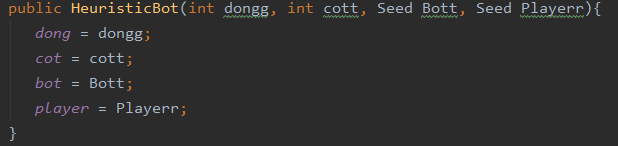
* playGame(): phương thức này được override từ lớp Play2Players. Phương thức chơi không đổi nhưng người chơi 1 sẽ đánh với người chơi 2 là máy thuộc lớp EasyBot hay HeuristicBot.



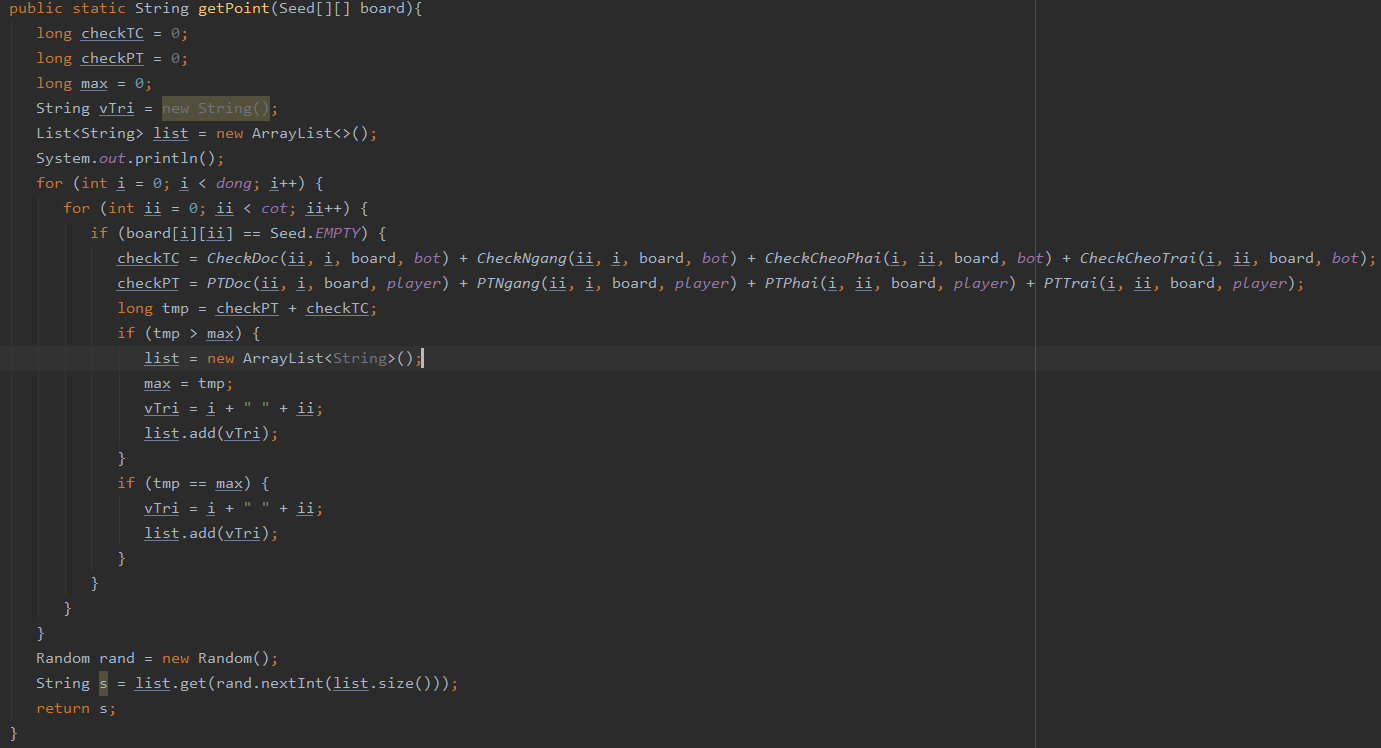
* runFirst(): Hàm dùng để xử lý việc lấy điểm ngẫu nhiên trên bàn cờ, sau đó tìm thêm vùng lân cận của nó theo số lần timeRun cần.
* HeuristicBot:



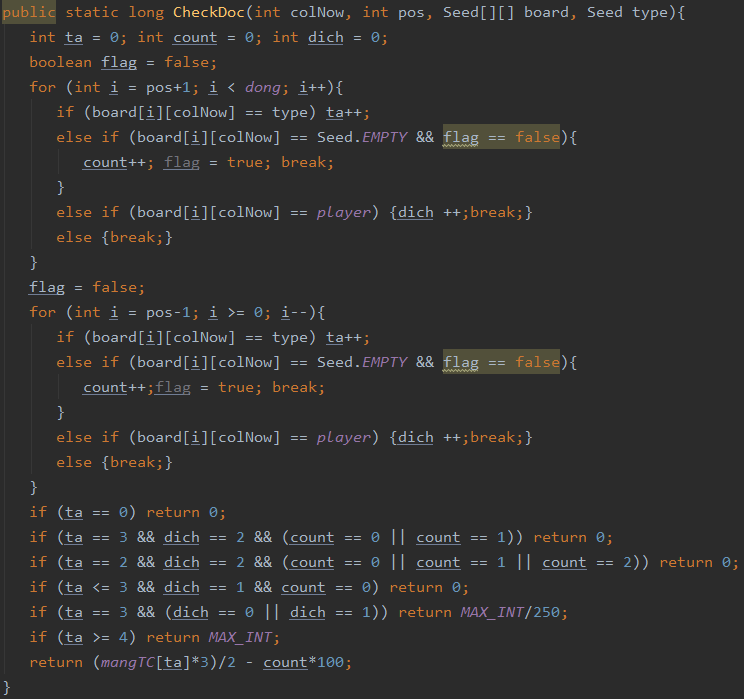
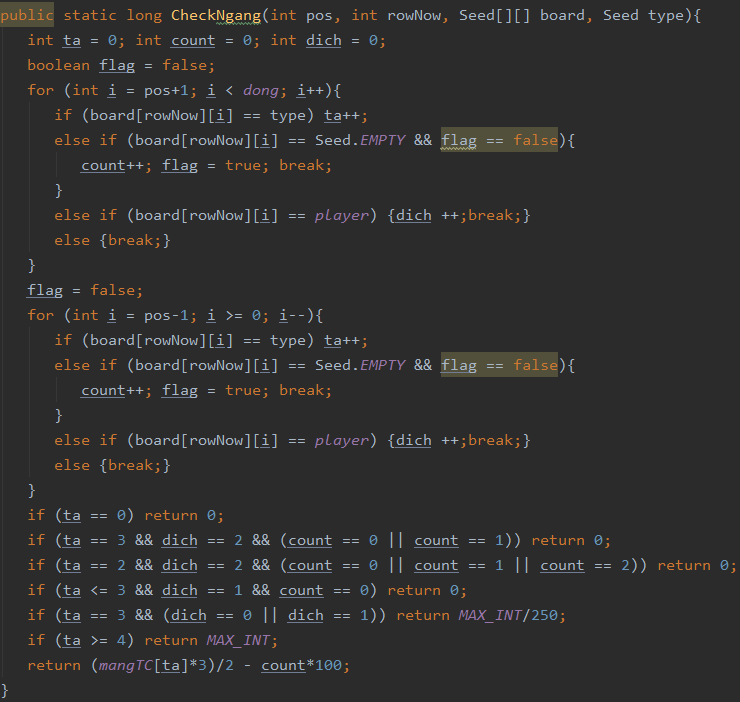
* dong,cot: chỉ dòng cột mà máy có thể tương tác.
* bot, player: kiểu của bot, người chơi.
* mangTC: mảng chứa các điểm tấn công của Bot.
* mangPN: mảng chứa các điểm phòng ngự của Bot.

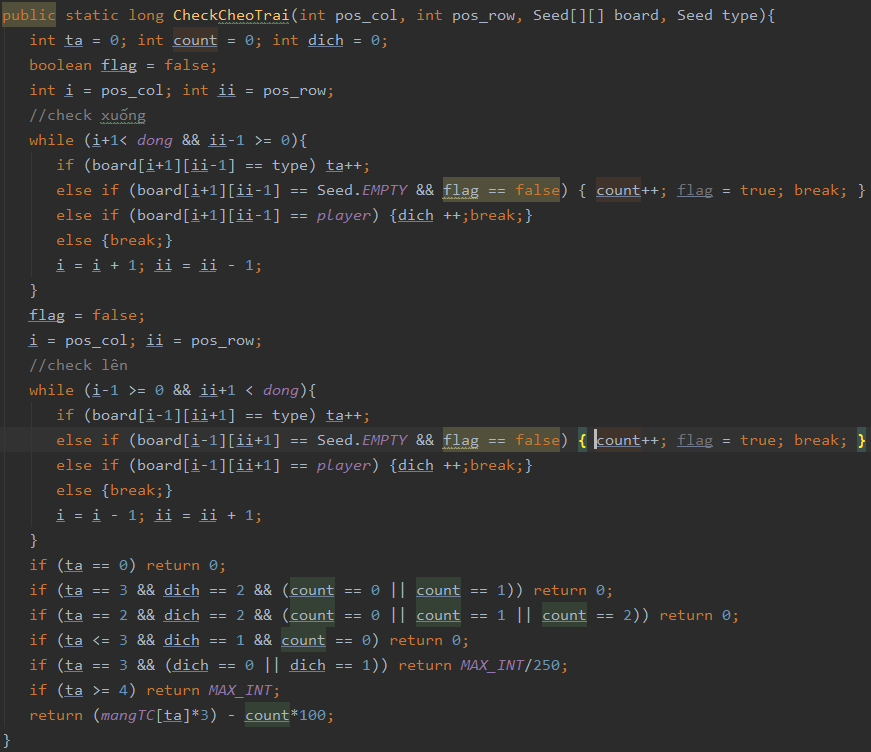
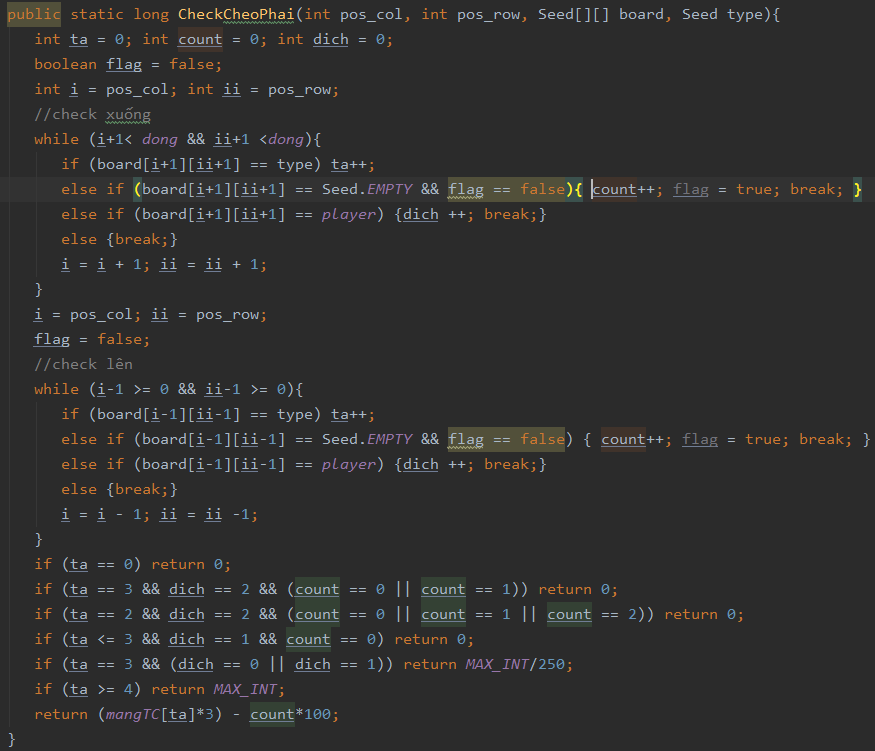


* HeuristicBot(): hàm tạo của HeuristicBot.

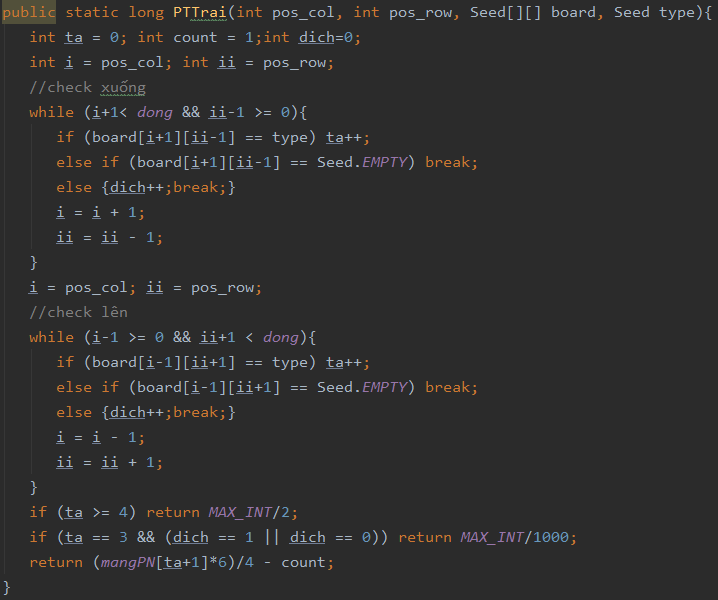
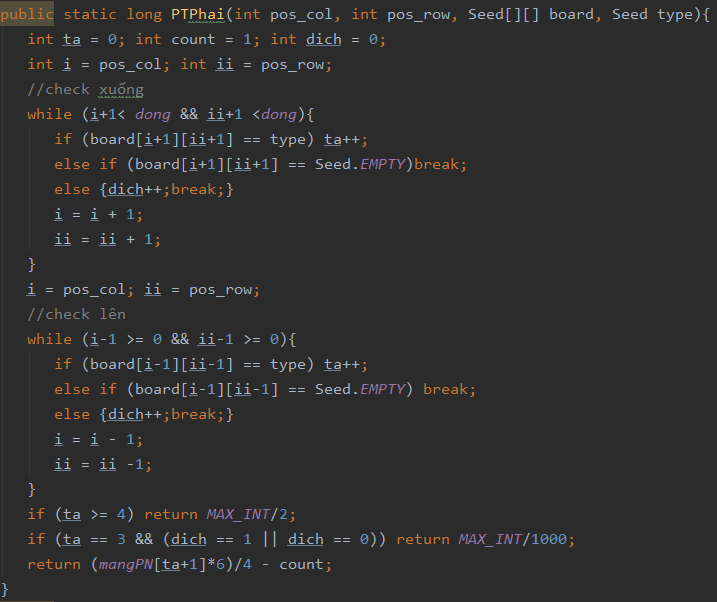
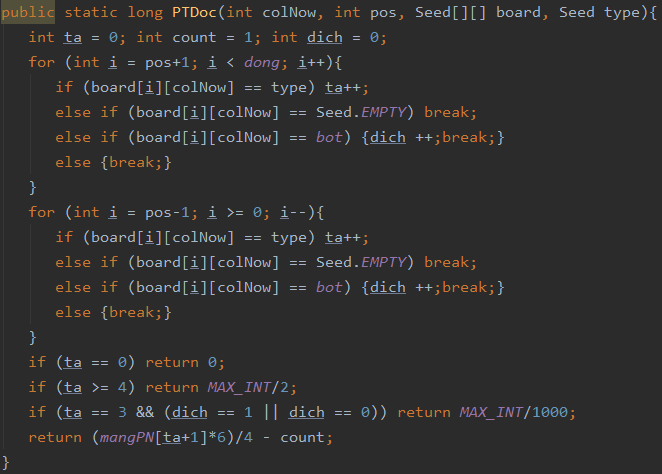
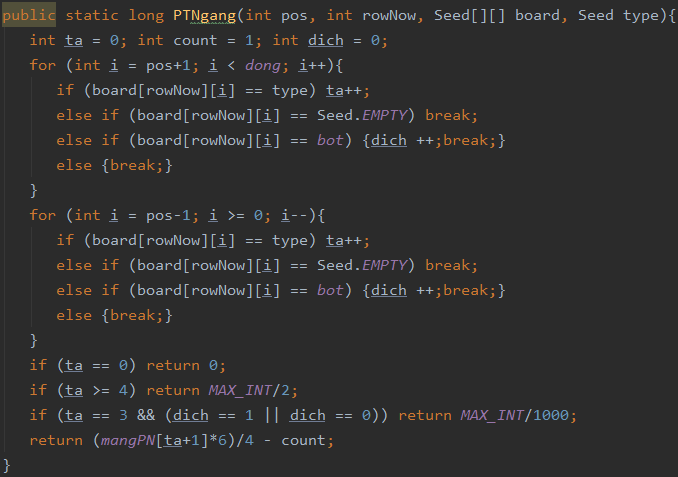


* getPoint(): Duyệt toàn bộ bàn cờ. Nếu có ô trống nào thì sẽ check tính tấn công và phòng thủ của nó, từ đó lập ra một danh sách vị trí thích hợp có thể đánh dấu được nhằm tạo điều kiện thắng lợi và giảm bớt xác xuất thua lớn nhất có thể. Hàm trả về là một vị trí ô có trong danh sách (lấy ngẫu nhiên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





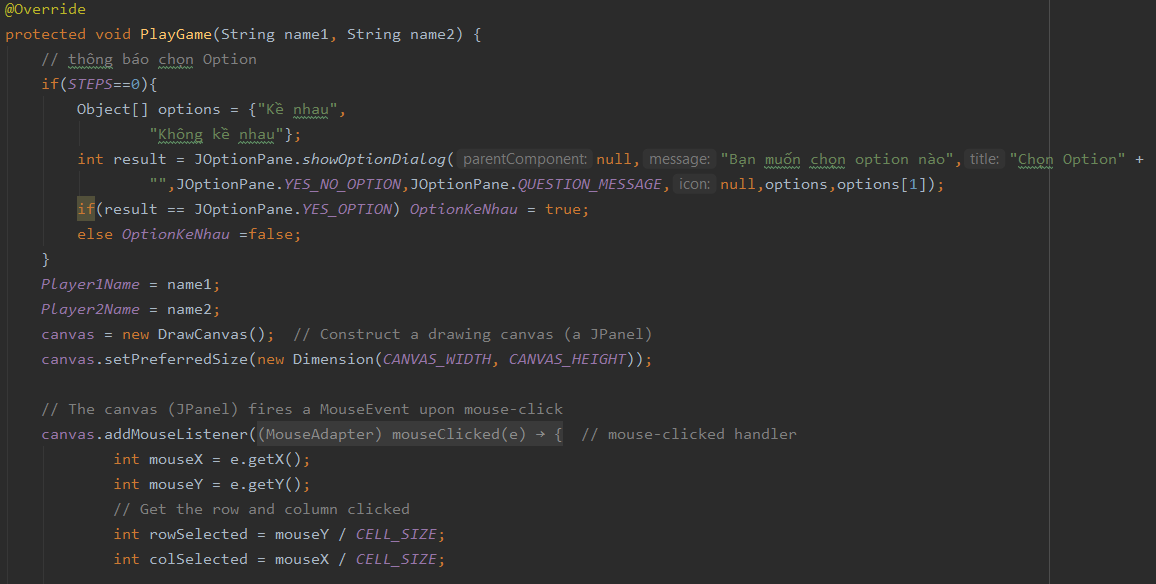
* CheckNgang(), CheckDoc(), CheckCheoPhai() và CheckCheoTrai(): Các hàm này dùng để tính điểm tấn công của Bot. Các Hàm trả về là một điểm tấn công thích hợp qua việc xử lý thông tin các ô Bot đã đánh dấu, và các ô người chơi đã đánh dấu trên phạm vi ngang, dọc, chéo.

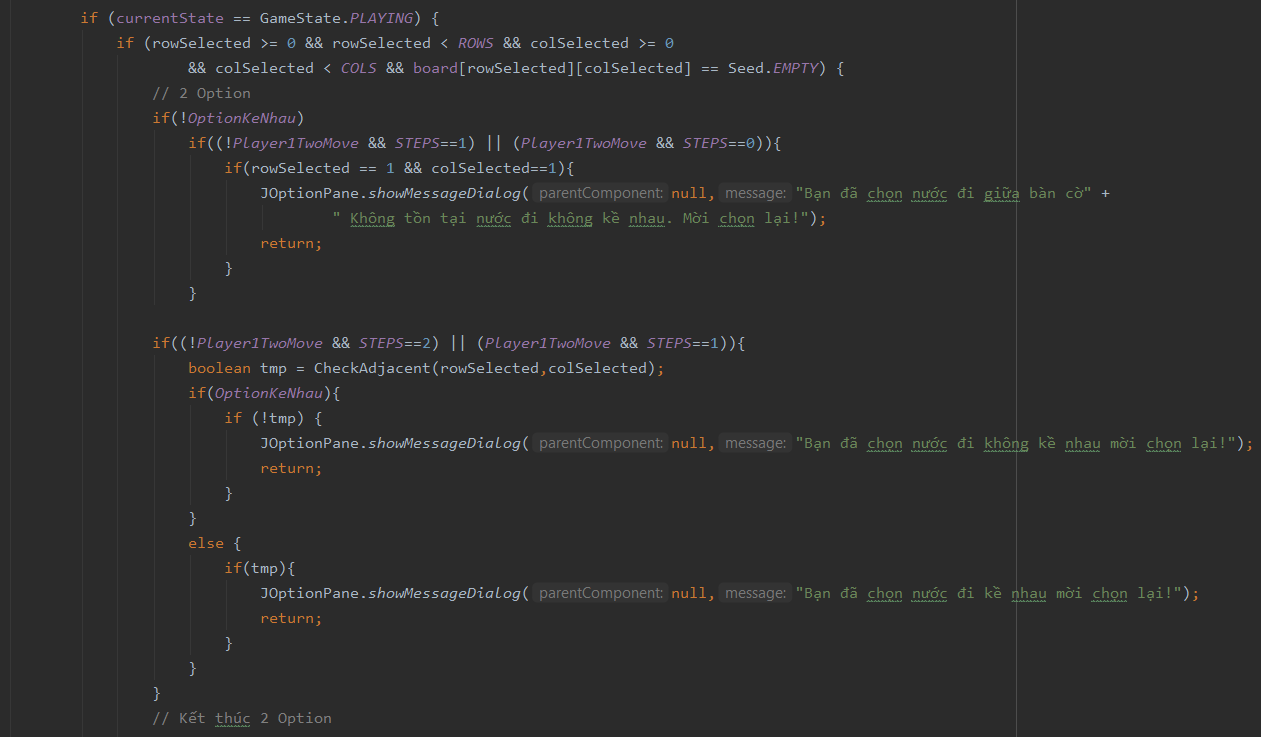


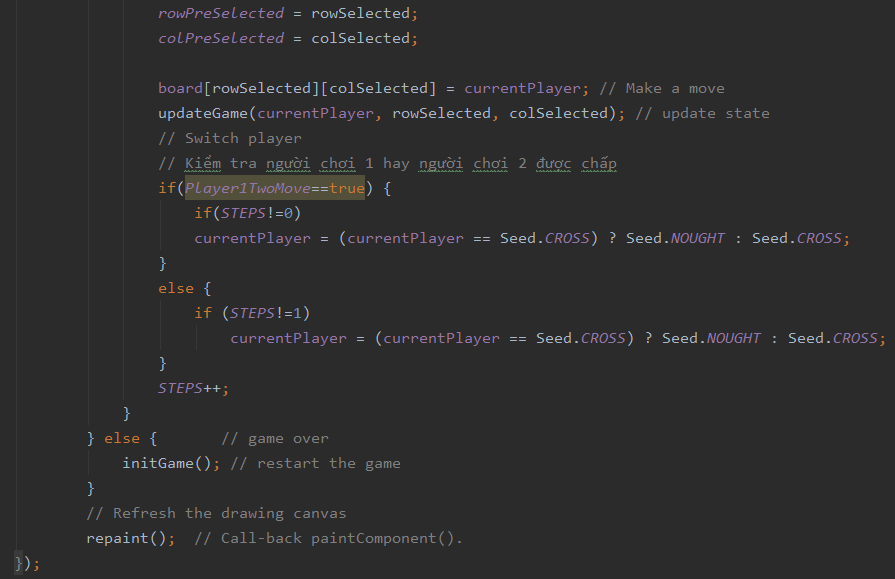
* PTNgang(), PTDoc(), PTPhai() và PTTrai(): Các hàm này dùng để tính điểm phòng ngự của Bot. Các Hàm trả về là một điểm phòng ngự thích hợp qua việc xử lý thông tin các ô Bot đã đánh dấu, và các ô người chơi đã đánh dấu trên phạm vi ngang, dọc, chéo.
* EasyBot:



* 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
* ModChap1Nuoc2Nguoi:

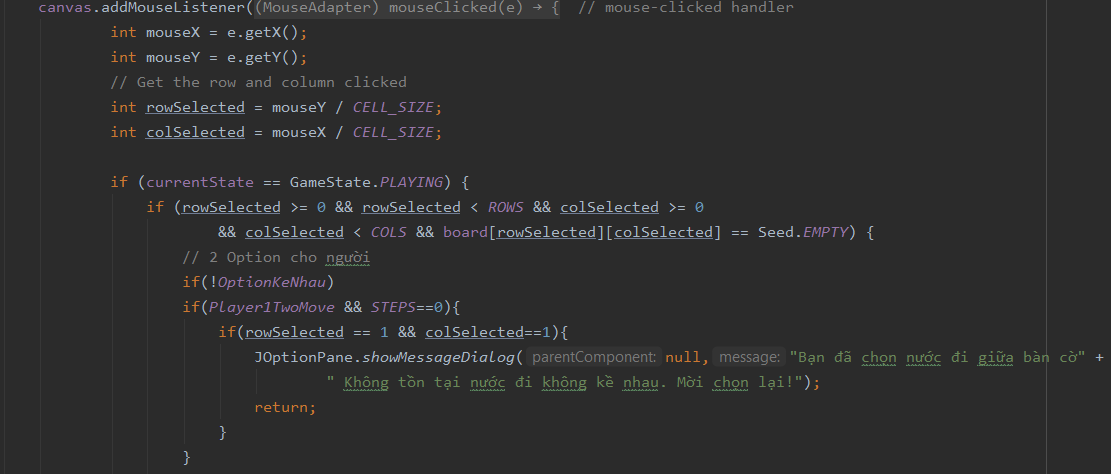


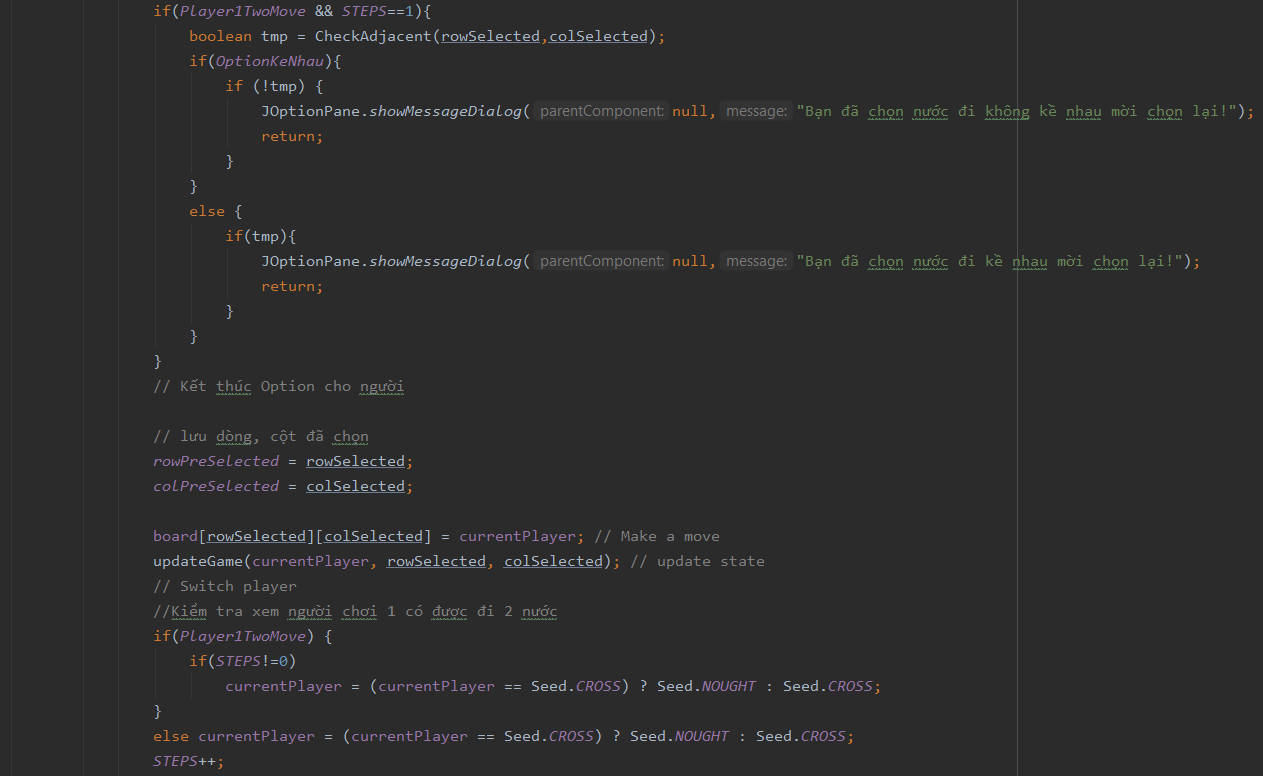




* playGame(): Phương thức này được override từ lớp Play2Players. Phương thức chơi không đổi nhưng sẽ cho 1 trong 2 người chơi được đánh 2 kí hiệu liên tiếp trong lượt đầu tiên với 1 trong 2 lựa chọn là đánh liền kề hay không liền kề.
* ModChap1NuocVSMay:

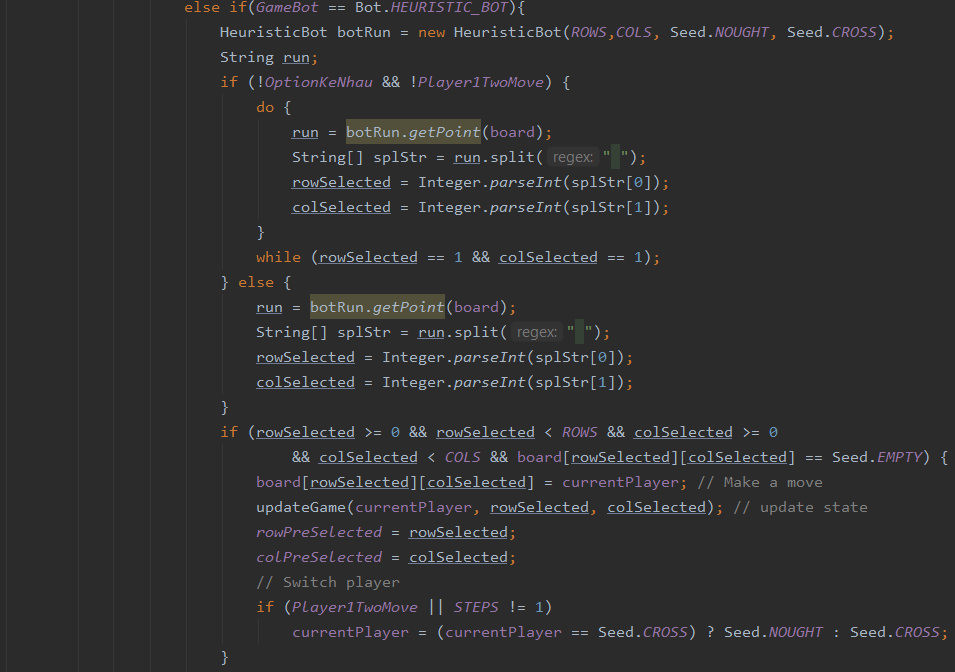


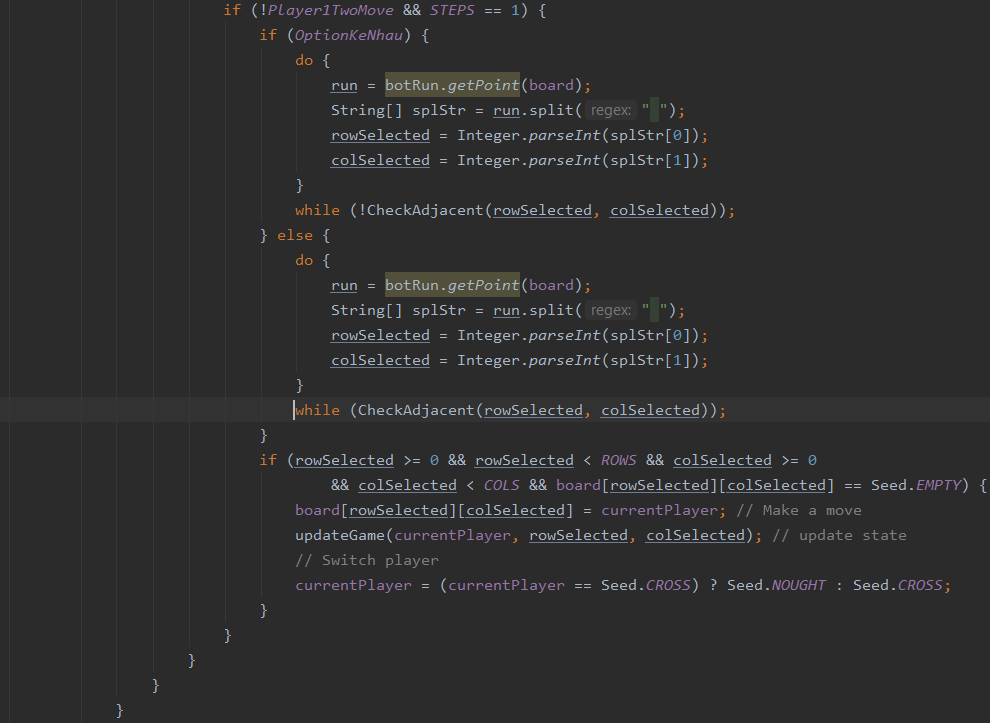


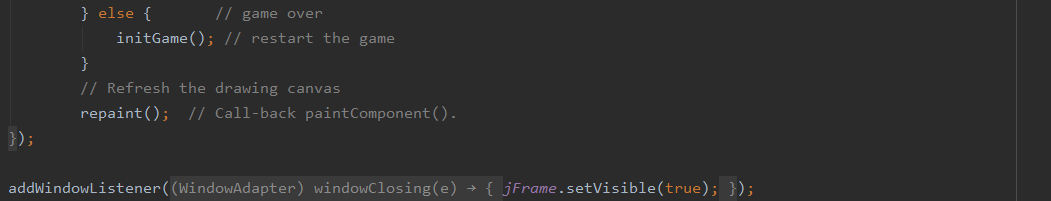




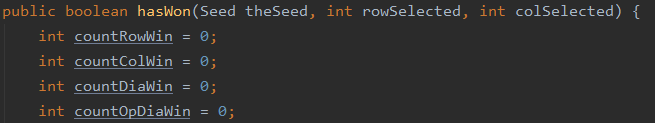


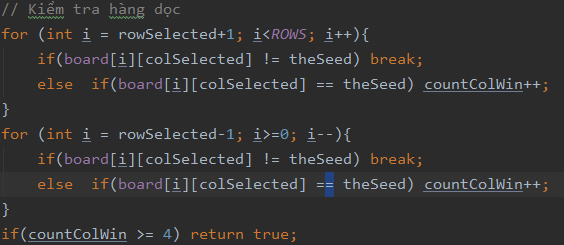
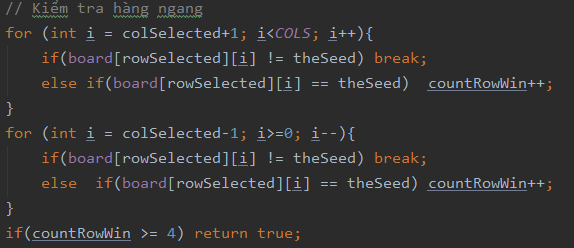


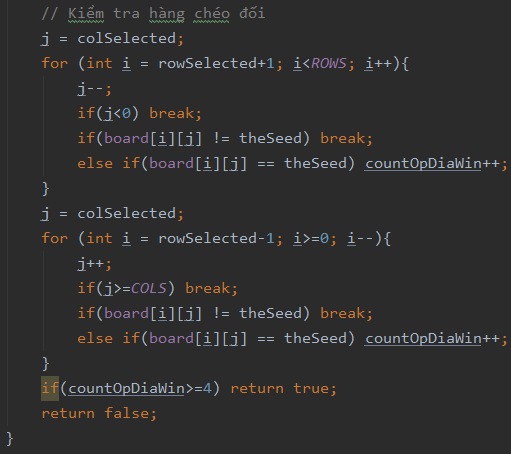
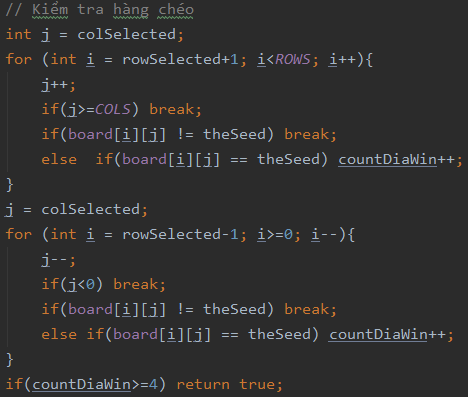




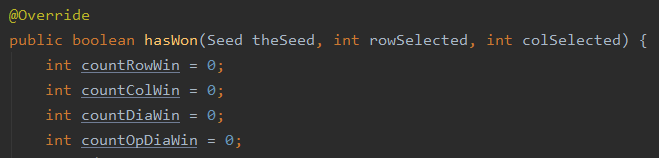
* + playGame(): Phương thức này được override từ lớp PlayWithAI. Phương thức chơi không đổi nhưng sẽ cho người chơi 1 đánh 2 kí hiệu liên tiếp trong lượt đi đầu với 1 trong 2 lựa chọn là đánh liền kề hay không liền kề.
* PLay2PlayersCaro, PlayWithAiCaro, ModChap1Nuoc2NguoiCaro và ModChap1NuocVSMayCaro:

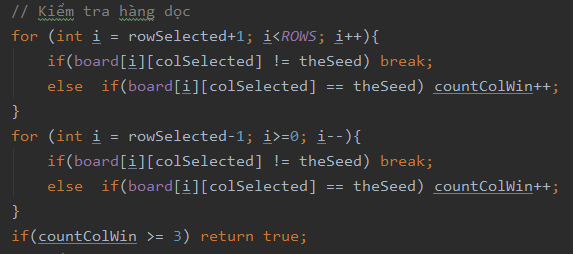
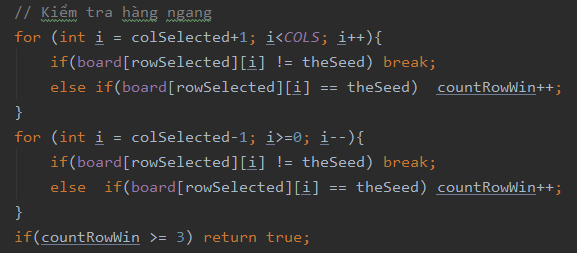


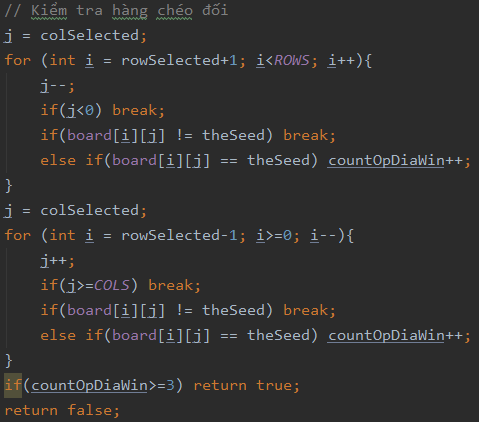
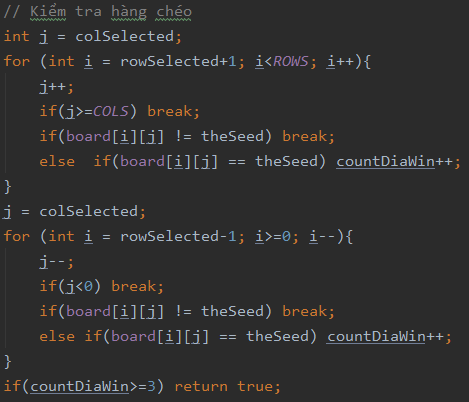




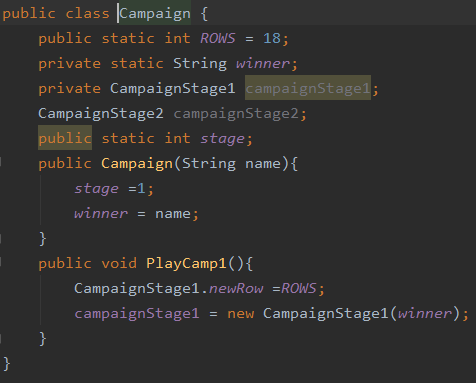
* HasWon(): Hàm đã được override. Dùng cho bàn cờ có số dòng = cột từ 5 trở lên. Trong bàn cờ, nếu có người chơi nào tạo nên được một dãy ký hiệu 5 liên tiếp theo hàng ngang (hoặc dọc hoặc chéo) thì trả về True.
* PLay2PlayersConnect4, PlayWithAiCaroConnect4, ModChap1Nuoc2NguoiConnect4 và ModChap1NuocVSMayConnect4:



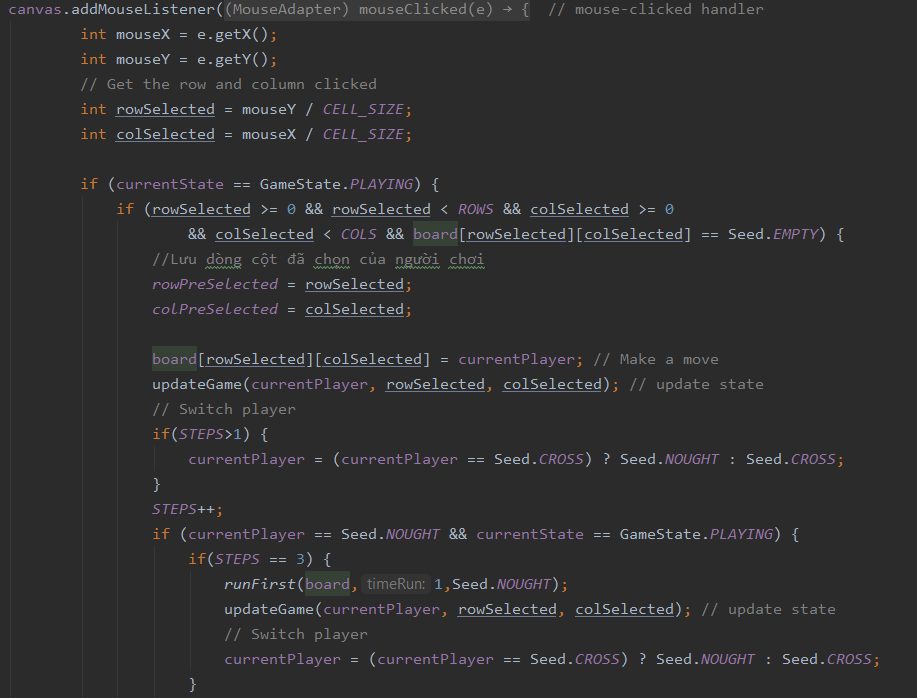


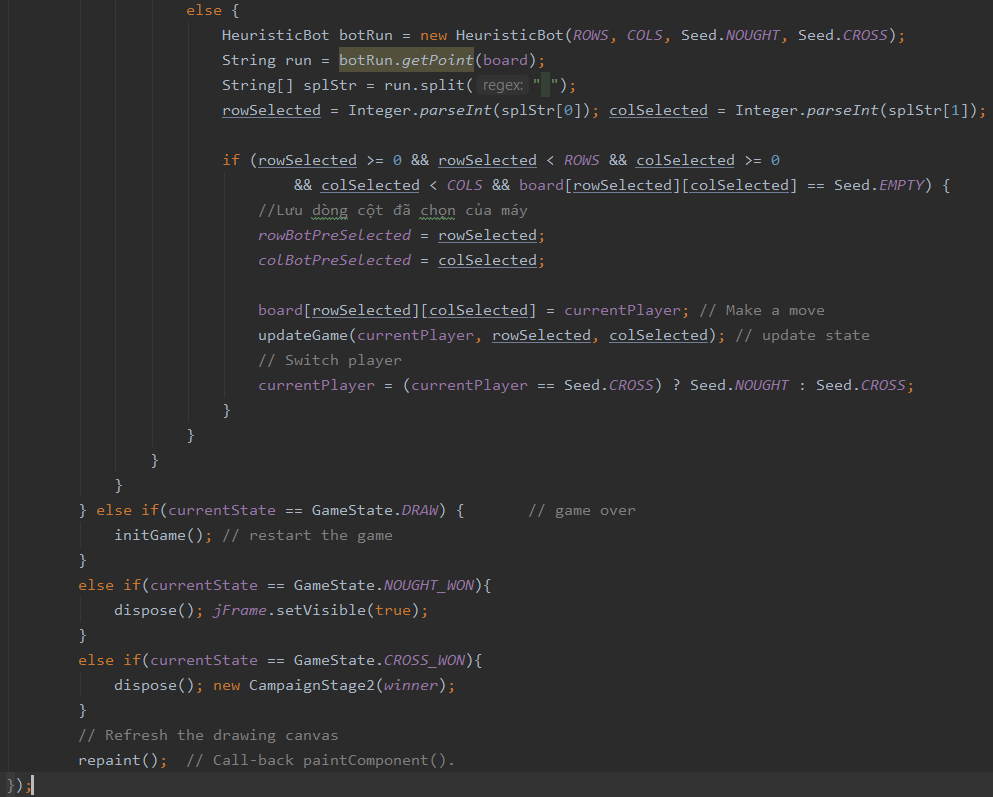


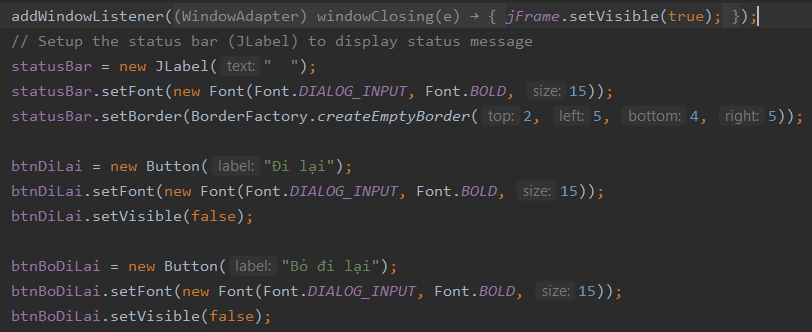
* HasWon(): Hàm đã được override. Dùng cho bàn cờ có số dòng = cột là 4 (tức 16 ô). Trong bàn cờ, nếu có người chơi nào tạo nên được một dãy ký hiệu 5 liên tiếp theo hàng ngang (hoặc dọc hoặc chéo) thì trả về True.
* Campaign:



* ROWS: chỉ số dòng (cũng là số cột). Trong Campaign thì số dòng cột là 18.
* Campaign(): Hàm khởi tạo cho Campaign.
* PlayCam1(): Hàm để tạo CampaignStage1.
* CampaignStage1:

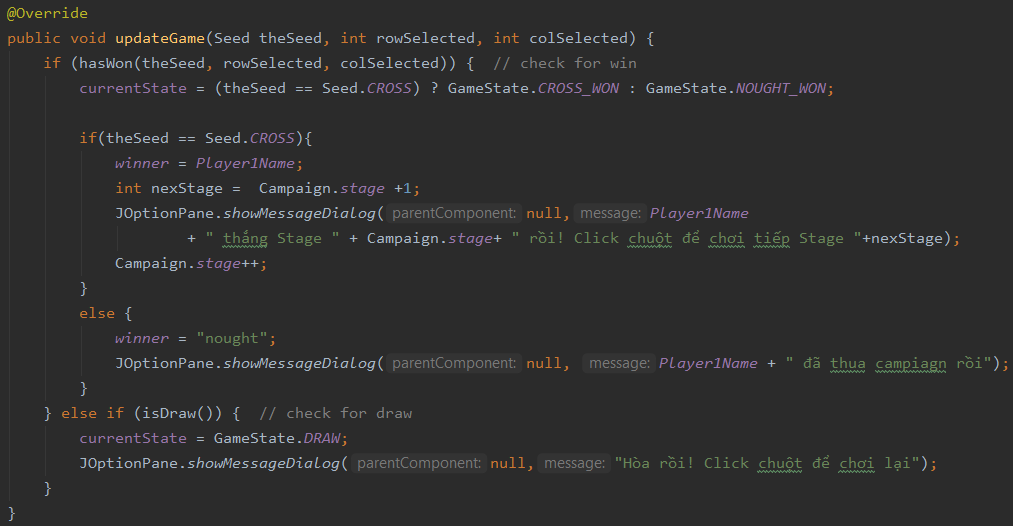




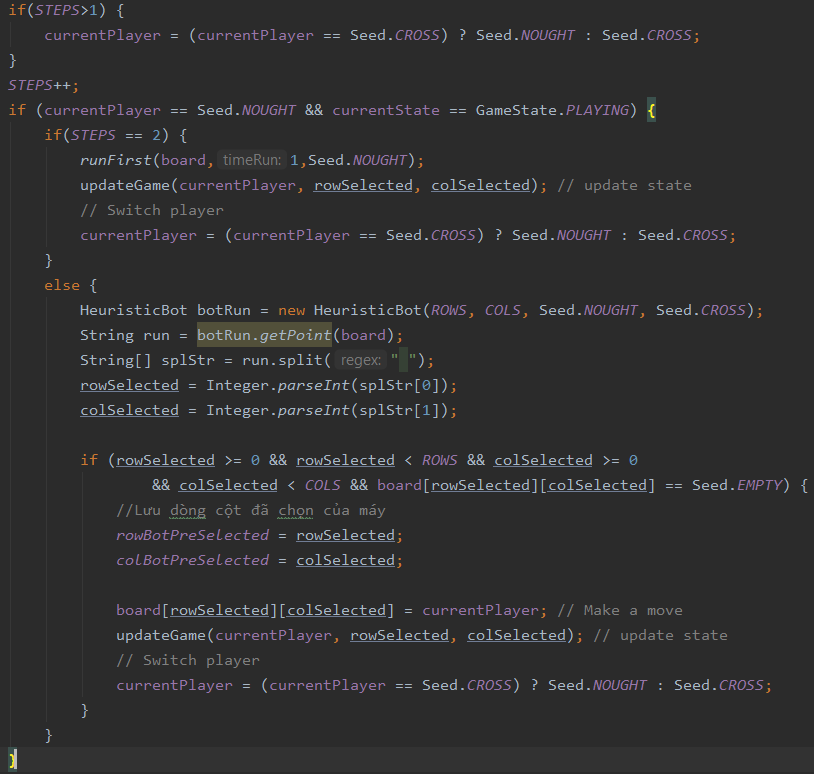




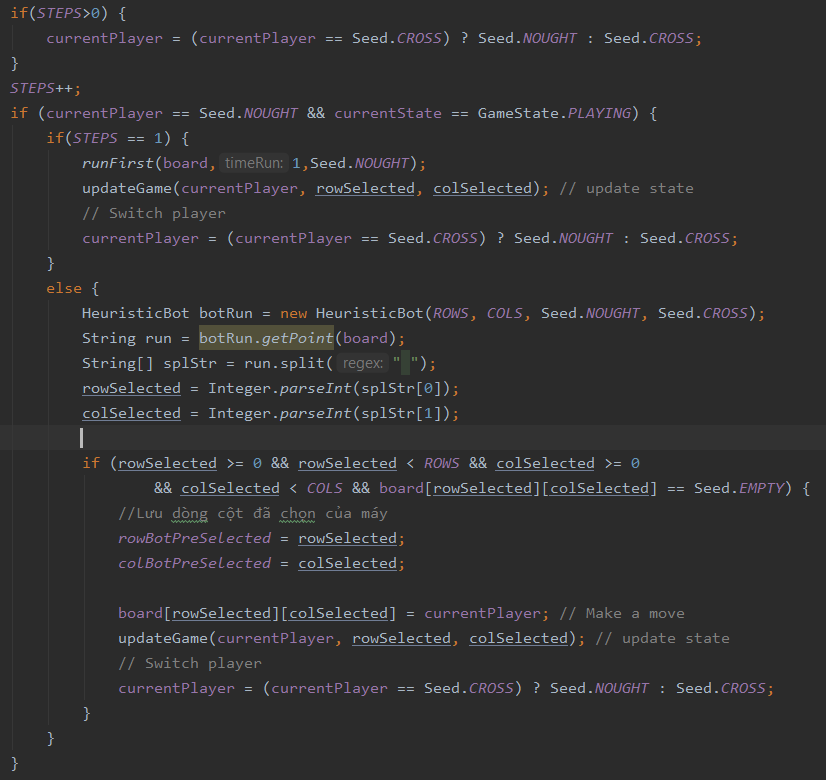
* PLayGame(): Được override từ lớp PlayWithAiCaro. Disable các nút **đi lại** và **bỏ đi lại.** Hàm dùng để chơi với máy theo chế độ Campaign Stage 1. Mình được đánh trước liên tiếp 3 dấu và Bot sẽ đi ngẫu nhiên một lần, về sau Bot sẽ chơi theo thuật giải Heuristic. Nếu chơi thắng thì qua Stage tiếp, còn thua thì chơi lại từ đầu và hòa thì chơi lại Stage đó.



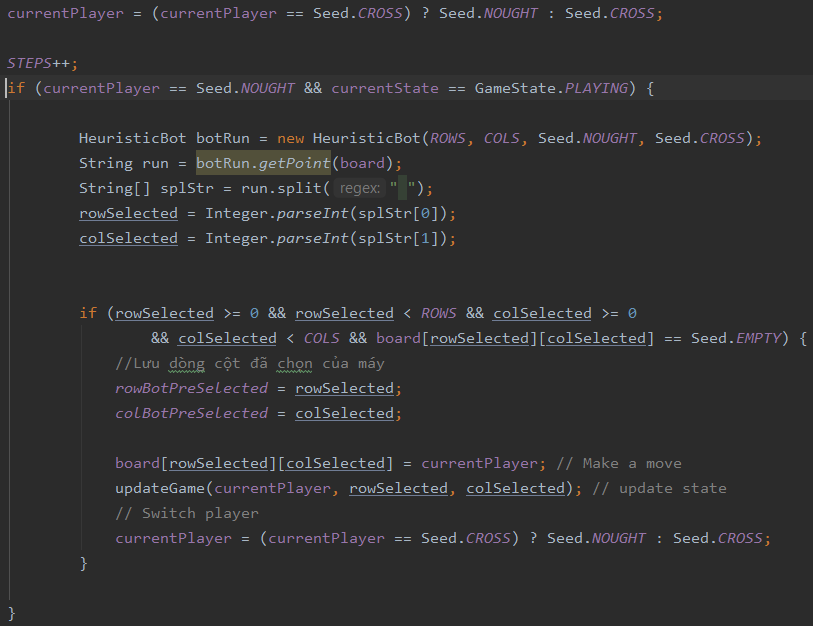
* updateGame(): Được override từ lớp PlayWithAiCaro. Hàm dùng để cập nhật tình hình bàn cờ. Nếu đủ điều kiện thắng thì xuất thông báo thắng, còn thua thì xuất thông báo thua và hòa xuất thông báo hòa.
* CampaignStage2:



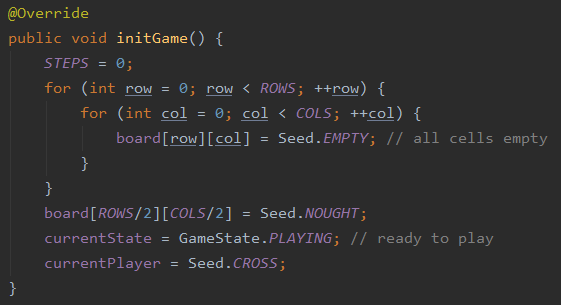
* PLayGame(): Được override từ lớp CampaignStage1.Hàm dùng để chơi với máy theo chế độ Campaign Stage 1. Mình được đánh trước liên tiếp 2 dấu và Bot sẽ đi ngẫu nhiên một lần, về sau Bot sẽ chơi theo thuật giải Heuristic. Nếu chơi thắng thì qua Stage tiếp, còn thua thì chơi lại từ đầu và hòa thì chơi lại Stage đó.
* CampaignStage3:



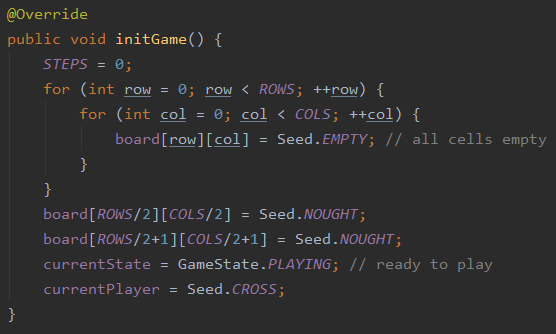
* PLayGame(): Được override từ lớp CampaignStage2.Hàm dùng để chơi với máy theo chế độ Campaign Stage 1. Mình được đánh trước và Bot sẽ đi ngẫu nhiên một lần, về sau Bot sẽ chơi theo thuật giải Heuristic. Nếu chơi thắng thì qua Stage tiếp, còn thua thì chơi lại từ đầu và hòa thì chơi lại Stage đó.
* CampaignStage4:



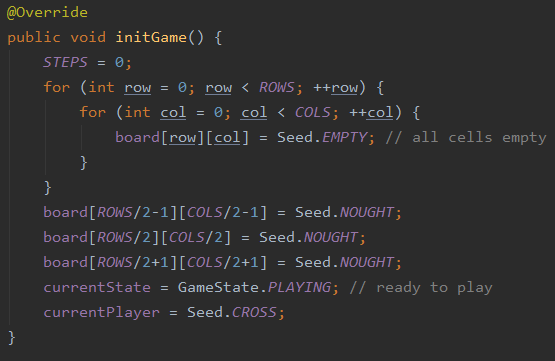
* PLayGame(): Được override từ lớp CampaignStage3.Hàm dùng để chơi với máy theo chế độ Campaign Stage 1. Mình được đánh trước và Bot chơi theo thuật giải Heuristic. Nếu chơi thắng thì qua Stage tiếp, còn thua thì chơi lại từ đầu và hòa thì chơi lại Stage đó.
* CampaignStage5:



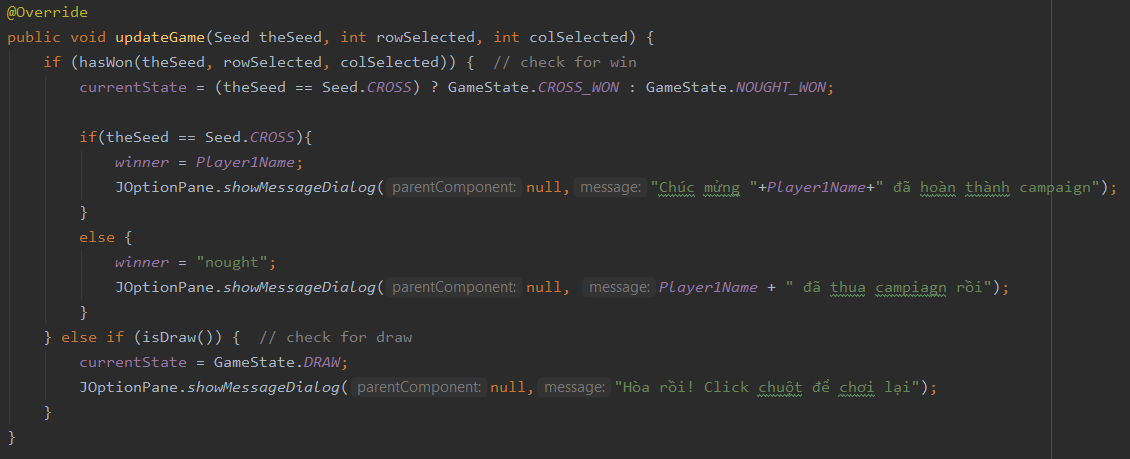
* initGame(): Được override từ lớp CampaignStage4. Hàm này dùng để khởi tạo nội dung bảng trò chơi và trạng thái của game. Làm tất cả ô vuông đều trống đồng thời cho một ký hiệu O ở vị trí giữa bảng, current state sẽ là PLAYING và người chơi ký hiệu X sẽ đánh lượt đầu tiên.
* CampaignStage6:



* initGame(): Được override từ lớp CampaignStage5. Hàm này dùng để khởi tạo nội dung bảng trò chơi và trạng thái của game. Làm tất cả ô vuông đều trống đồng thời cho 2 ký hiệu O ở vị trí giữa bảng, current state sẽ là PLAYING và người chơi ký hiệu X sẽ đánh lượt đầu tiên.
* CampaignStage7:



* initGame(): Được override từ lớp CampaignStage6. Hàm này dùng để khởi tạo nội dung bảng trò chơi và trạng thái của game. Làm tất cả ô vuông đều trống đồng thời cho 3 ký hiệu O ở vị trí giữa bảng, current state sẽ là PLAYING và người chơi ký hiệu X sẽ đánh lượt đầu tiên.



* updateGame(): Được override từ lớp CampaignStage6. Không thay đổi gì nhiều, chỉ thay đổi phần text thông báo.
* GetAnhSetHighScore:



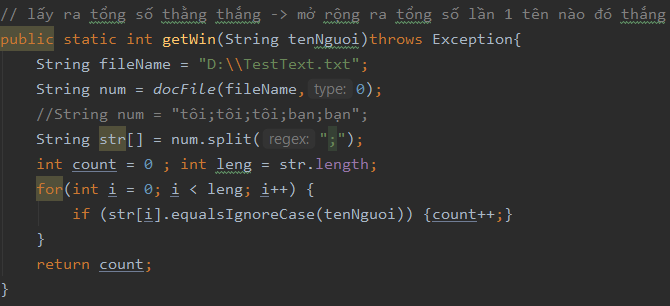
* FILE\_NAME: chỉ đường dẫn đến file txt.



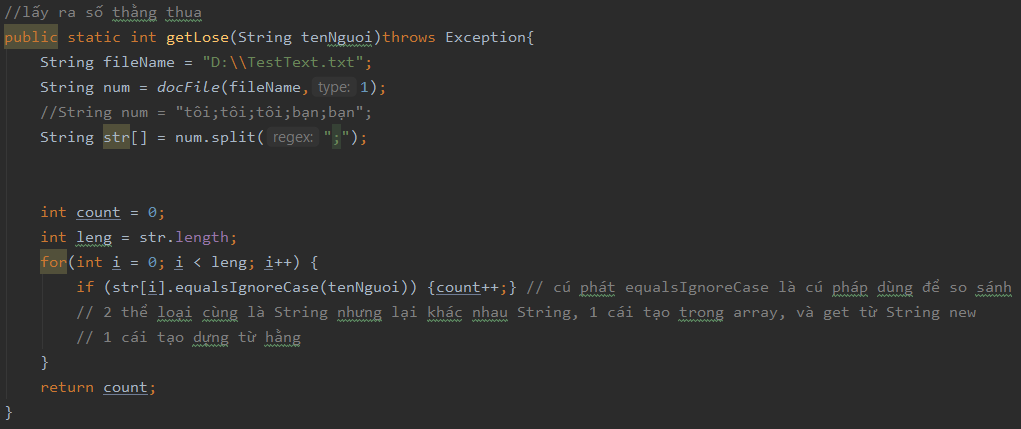
* docFile(): Hàm dùng để đọc file txt trong đường dẫn. Nếu không có tìm thấy file txt đó thì sẽ tạo file txt trong đường dẫn đó. Thông tin trả về sẽ là một chuỗi người chơi theo định dạng int type.



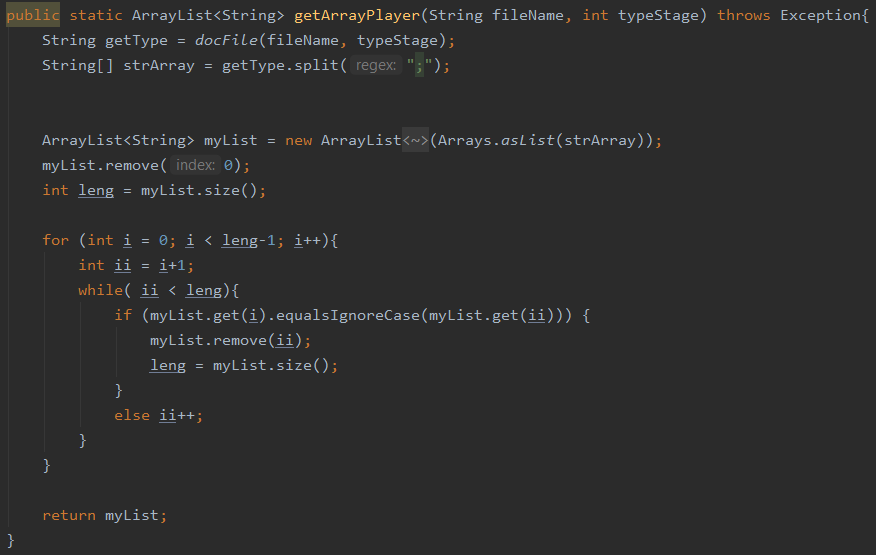
* ghiFile(): Hàm dùng để ghi dữ liệu vào file txt (nếu không thấy file trong đường dẫn thì sẽ tạo file luôn). File txt có 2 dòng là thắng và thua, khi có người thắng thì sẽ ghi tên người đó vào dòng thắng và thua thì sẽ ghi vào dòng thua, giữa mỗi lần ghi được năng cách nhau bởi dấu chấm phẩy (nếu trường hợp là máy thì chỉ ghi lại người chơi 1).



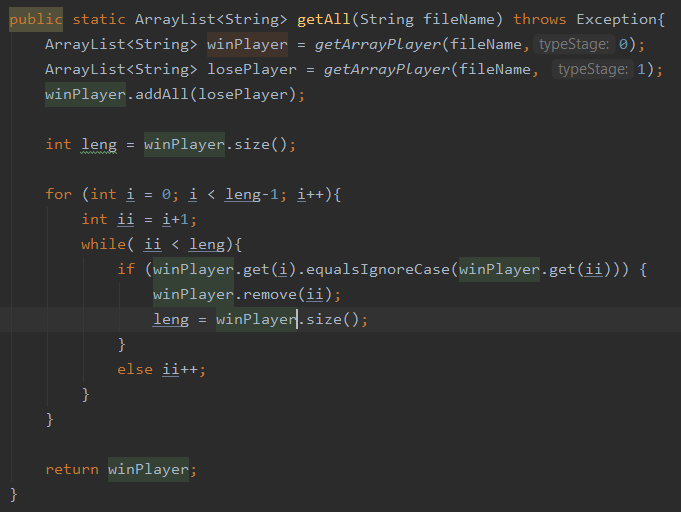
* getWin(): Dò số lần thắng theo tên người chơi rồi trả về tổng số lần thắng của người đó.



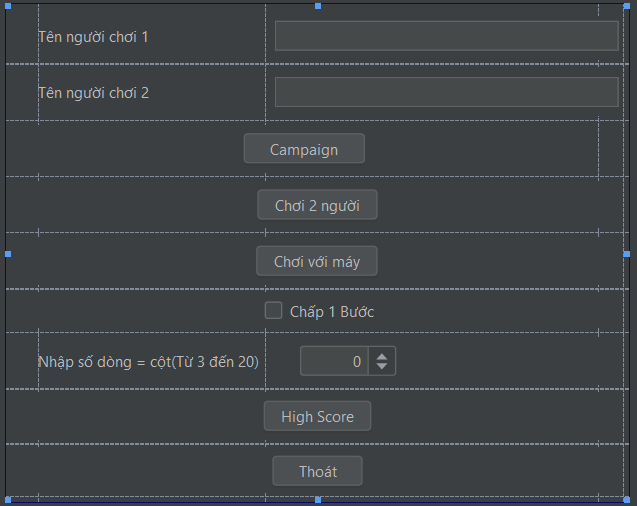
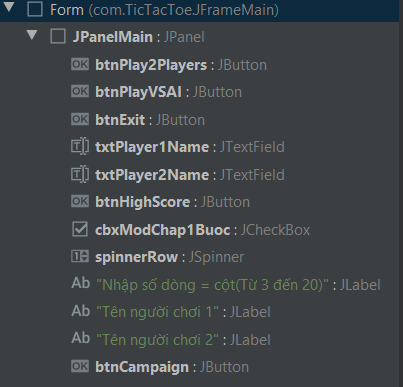
* getLose(): Dò số lần thua theo tên người chơi rồi trả về tổng số lần thua của người đó.



* getArrayPlayer(): Đọc dữ liệu trong file txt rồi làm thành một mảng chứa tên người thắng hoặc người thua (trong file tìm thấy tên người chơi trùng nhau thì xóa đi, giữ lại 1 tên duy nhất).



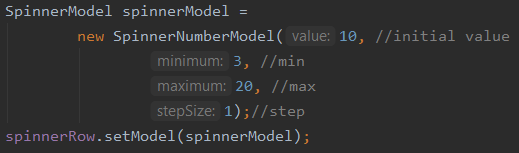
* getAll(): Tạo ra 2 mảng người thắng và người thua nhờ getArrayPlayer() cho người thắng và getArrayPlayer() cho người thua. Mảng trả về sẽ là một mảng hoàn chỉnh chứa tên của người chơi.
* JFrameMain:



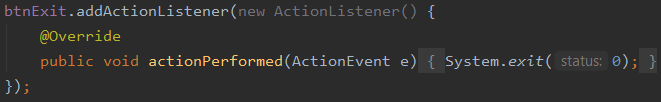
JFrameMain.form



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

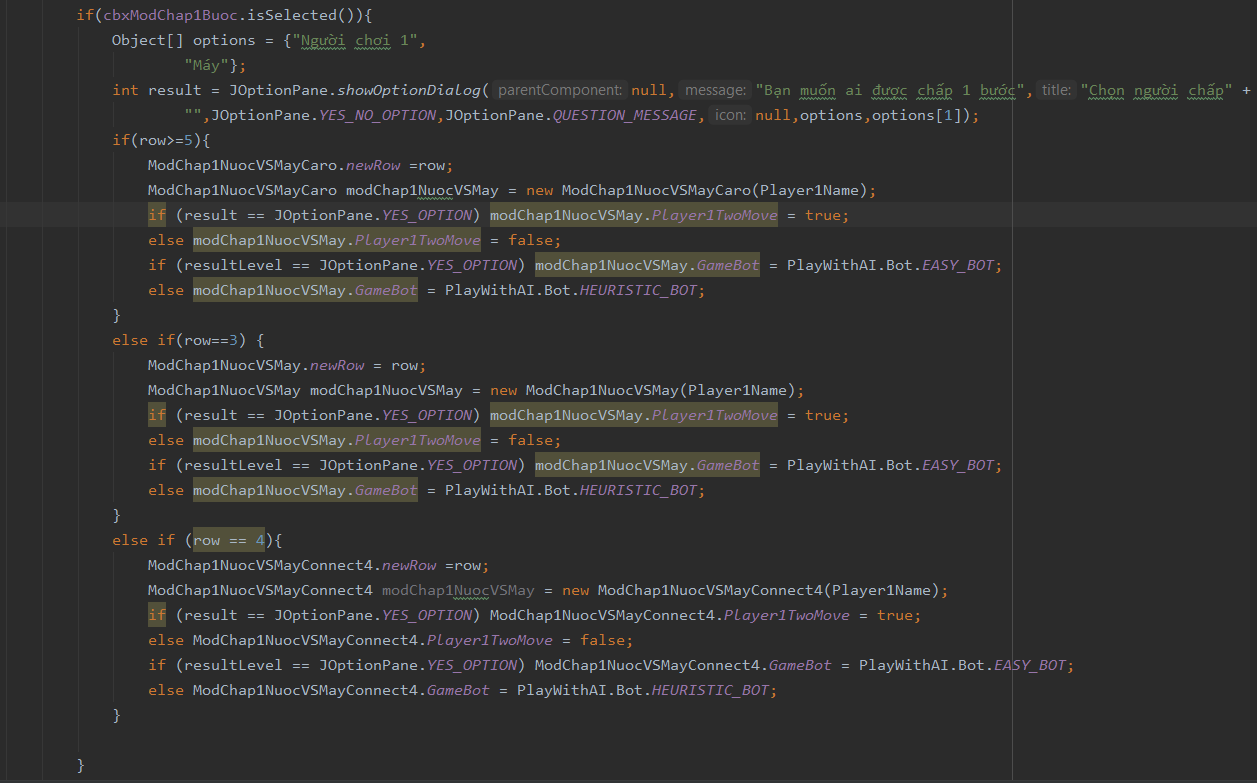


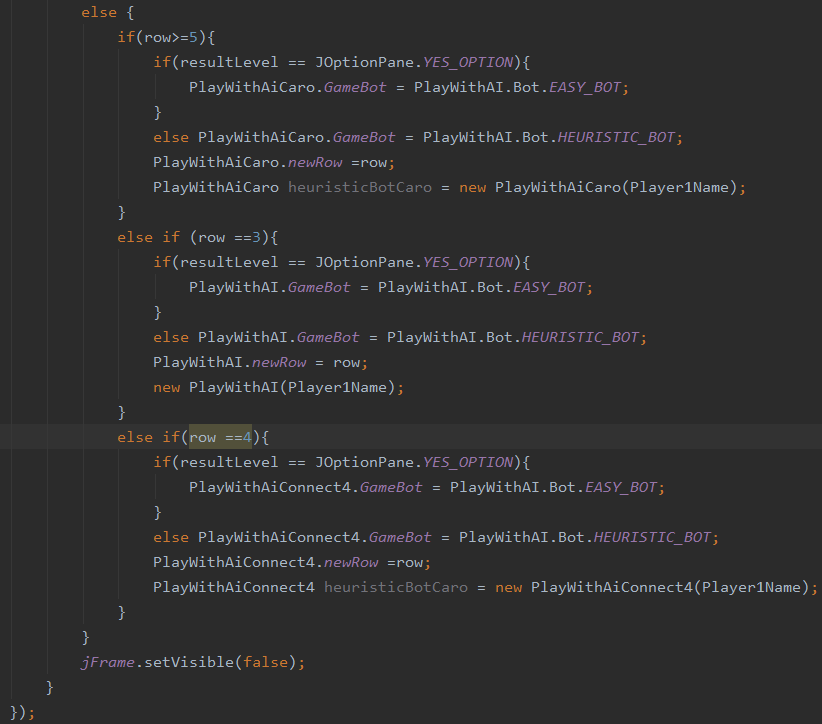
* Cài đặt model cho spinner nhập dòng, cột.



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

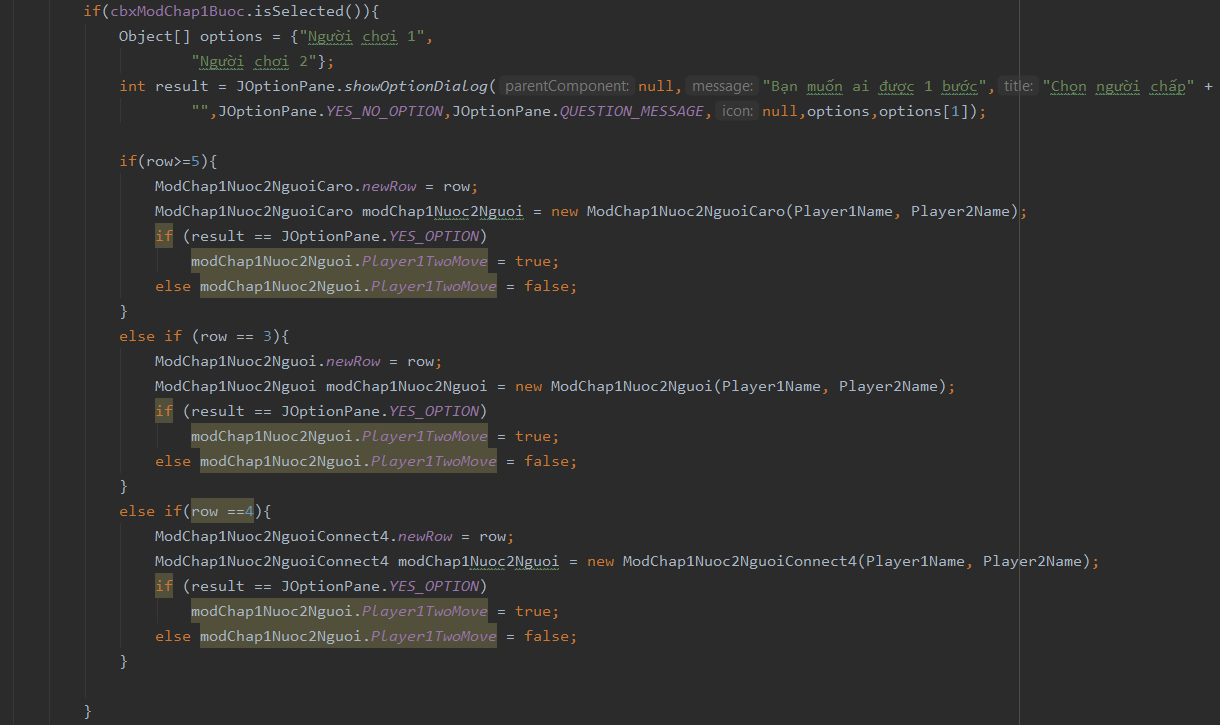


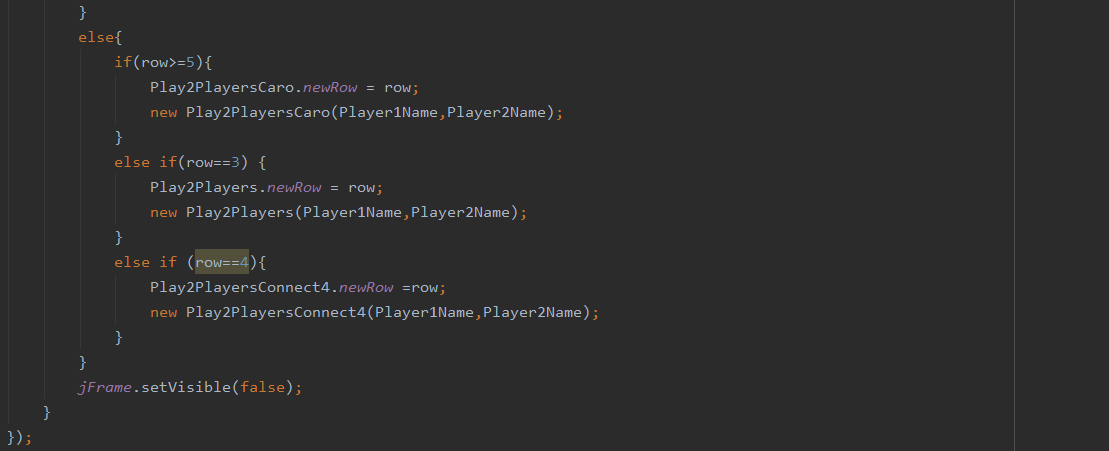




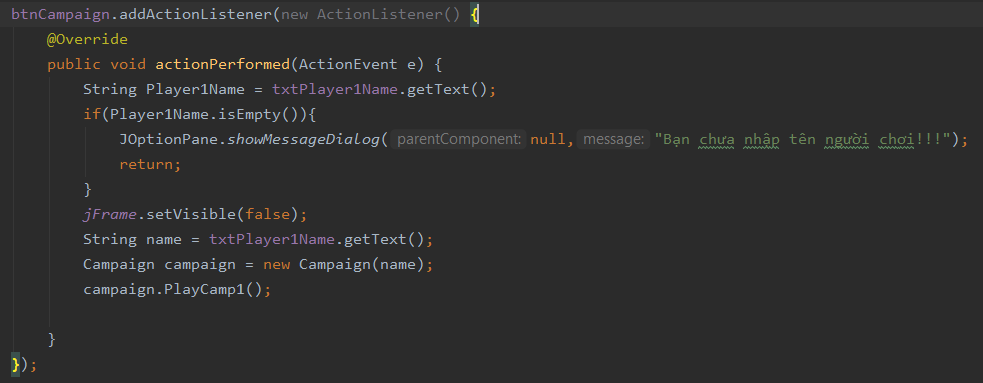
* btnPlayVSAI: Nút này dùng để chuyển giao diện chính thành vô hình, tạo bảng tùy vào số mình nhập ở SpinnerModel và chơi với máy nhưng yêu cầu phải nhập tên người chơi 1 ở txtPLayer1Name. Nếu bậc chế độ ở cbxModChap1Buoc chấp trước 1 bước thì sẽ hiển thị bảng chọn xem ai được chấp trước 1 bước rồi mới tạo bảng chơi.



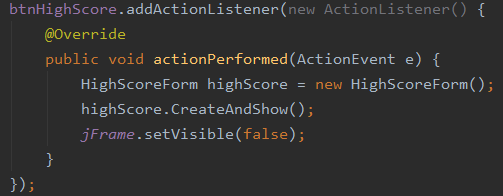




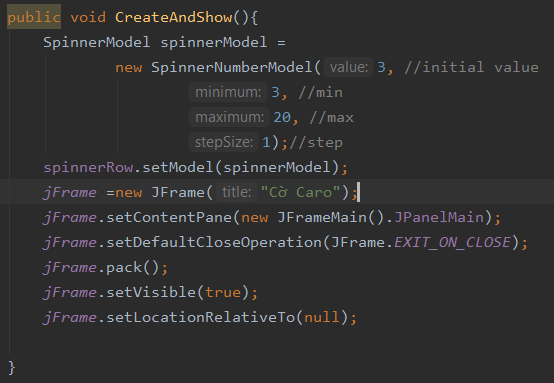
* btnPLay2PLayers: Nút này dùng để chuyển giao chính thành vô hình, tạo bảng tùy vào số mình nhập ở SpinnerModel và chơi 2 người nhưng yêu cầu phải nhập tên 2 người chơi ở txtPLayer1Name và txtPlayer2Name. Nếu bậc chế độ ở cbxModChap1Buoc chấp trước 1 bước thì sẽ hiển thị bảng chọn xem ai được chấp trước 1 bước rồi mới tạo bảng chơi.



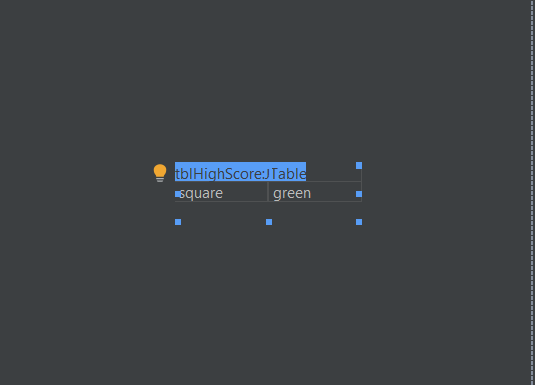
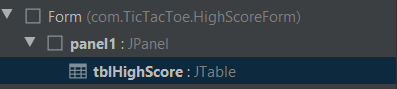
* btnCampaign: Nút này thực hiện việc chơi Campaign. Yêu cầu nhập tên người chơi mới có thể thực hiện.



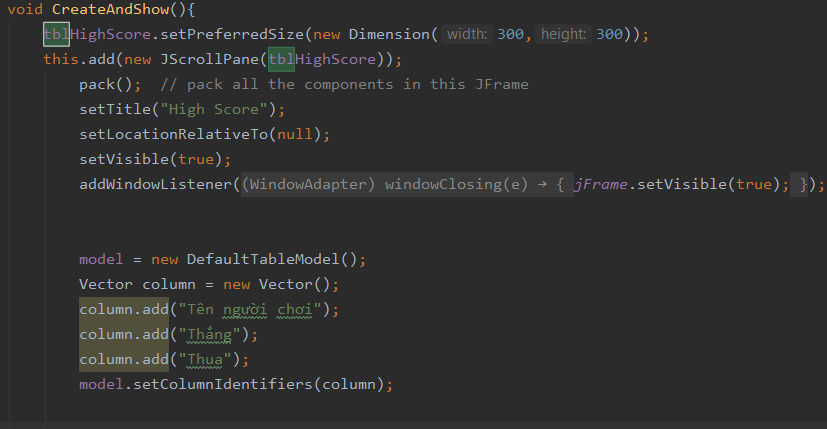
* btnHighScore: Nút này thực hiện việc hiển thị giao diện lịch sử qua CreateAndShow() của lớp HighScoreForm, chuyển giao chính thành vô hình.



* CreateAndShow(): Thực hiện việc hiển thị giao diện với tiêu đề “Cờ Caro”.
* HighScoreForm: lớp kế thừa lớp JFrame



HighScoreForm.form

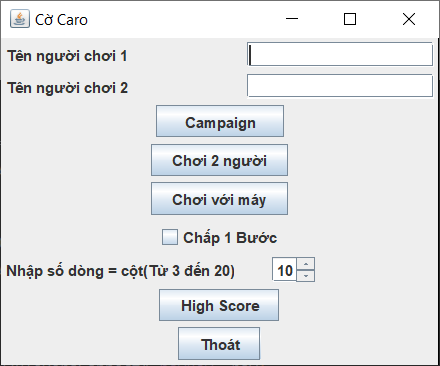




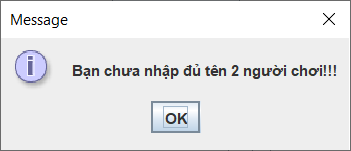
* CreateAndShow(): Tạo độ cao và rộng cho bảng danh sách high score trong giao diện. Lấy dữ liệu thông qua lớp GetAndSetHighScore và hiển thị giao diện với tiêu đề “High Score”.

1. **Giao diện ứng dụng:**

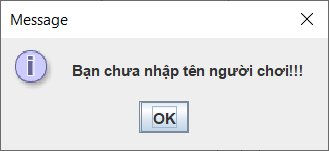
* Giao diện khởi đầu:



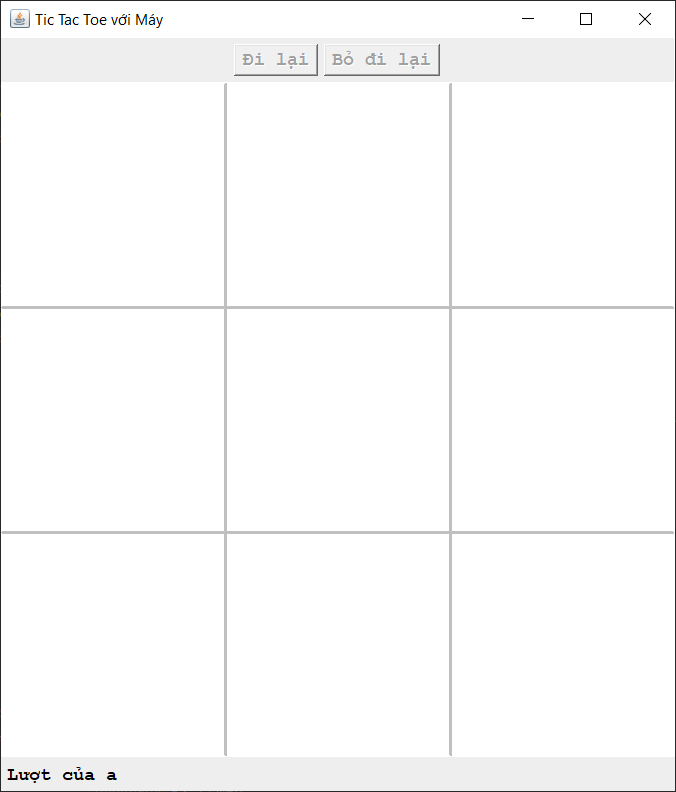
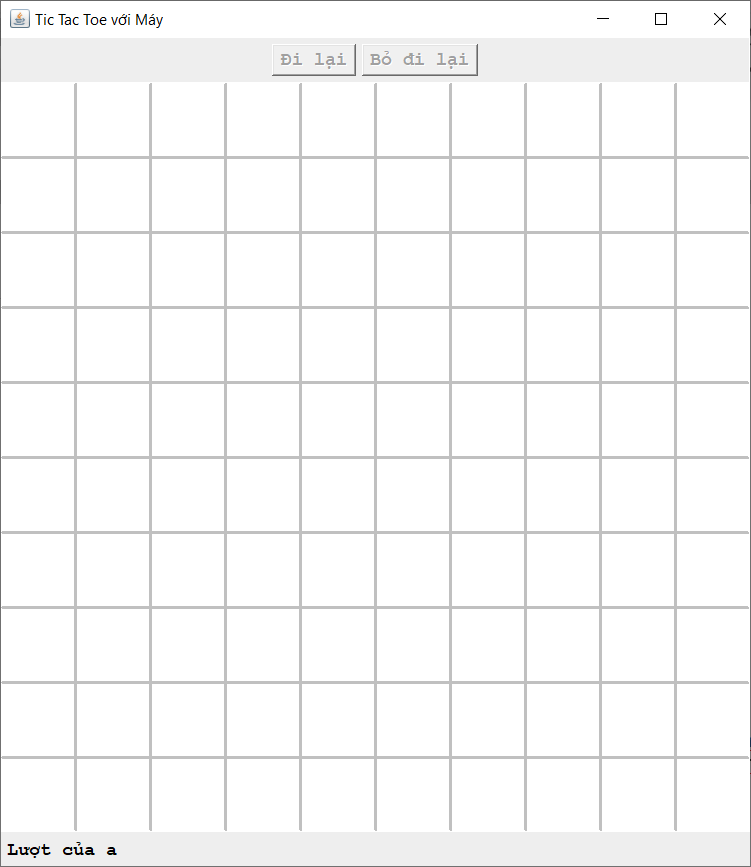
* Thông báo nhập tên:
* Nếu chọn chơi 2 người:



* Nếu chọn chơi với máy hoặc Campaign:

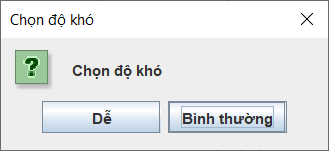


* Giao diện chơi Game:

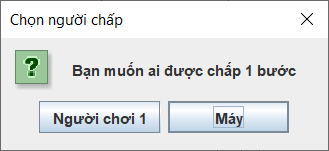
 

Giao diện 3 dòng 3 cột Giao diện 10 dòng 10 cột

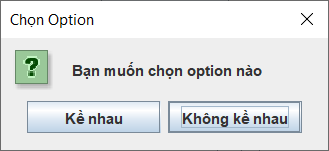
* Giao diện chọn độ khó Bot:



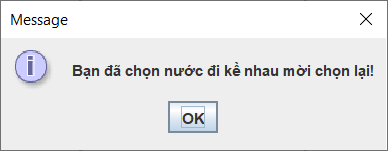
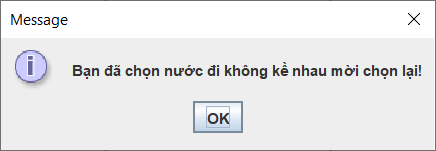
* Giao diện thông báo chế độ chấp trước 1 bước:
* Chọn ai chấp trước:



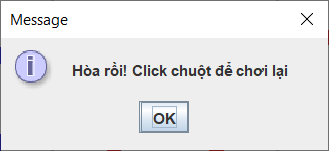
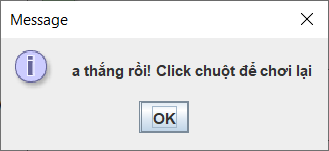
* Chế độ liền kề hay không liền kề:



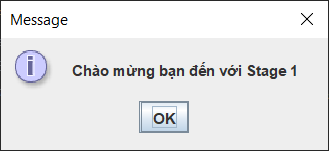
* Chơi đúng quy luật liền kề hay không liền kề:



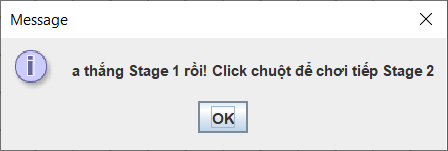
* Giao diện thông báo thắng hoặc hòa:



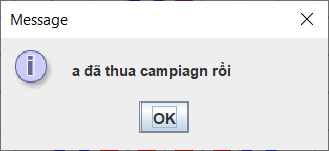
* Giao diện thông báo Campaign:
* Vào 1 Stage:



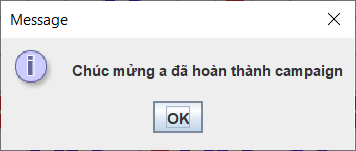
* Hoàn thành 1 Stage:



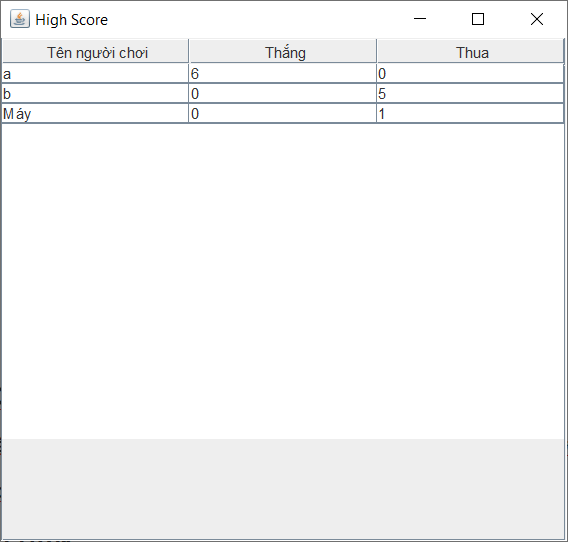
* Thua Campaign:



* Hoàn thành Campagn:



* Giao diện xem High Score theo tên người chơi:



1. **HƯỚNG PHÁT TRIỂN:**

Việc lập trình ra được trò chơi giúp nhóm tạo tiền đề cho các trò chơi có độ khó lập trình sau này. Củng cố thêm nền tảng kiến thức cơ bản để có thể tạo Bot chơi mạnh hơn như Bot dùng thuật giải Minimax.

1. **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.

1. **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>

**HẾT**