**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN II**

-----🙞🙜🕮🙞🙜-----



**NIÊN KHÓA: 2012-2017**

***Đề tài:***

**LẬP TRÌNH GAME CARO MẠNG LAN**

**BẰNG NGÔN NGỮ JAVA**

**NHÓM 7**

**Sinh viên: Lê Văn Ninh**

**Lớp : D12CQCN01-N**

**Mã SV: N12DCCN032**

**Giáo viên HD: TS. Nguyễn Văn Mùi**

Mục lục:

[1. Giới thiệu chung: 3](#_Toc388397868)

[2. Thuật toán: 3](#_Toc388397869)

[2.1. Thuật toán tìm đường đi của máy tính: 3](#_Toc388397870)

[2.2. Thuật toán kiểm tra chiến thắng: 14](#_Toc388397871)

[2.3. Thuật toán xây dựng kết nối mạng LAN: 18](#_Toc388397872)

[2.3.1. Ý tưởng: 18](#_Toc388397873)

[2.3.2. Xây dựng đối tượng Server: 18](#_Toc388397874)

[2.3.3. Xây dựng đối tượng Client: 19](#_Toc388397875)

[2.3.4. Mô tả hoạt động của Server và Client: 20](#_Toc388397876)

[2.3.5. Xây dựng code cho class Server.java và Client.java 22](#_Toc388397877)

[3. Đồ họa: 37](#_Toc388397878)

[3.1. Xây dựng đối tượng JPanel có khả năng chén ảnh: 37](#_Toc388397879)

[3.2. Xây dựng giao diện menu chính của game: 38](#_Toc388397880)

[3.3. Xây dựng giao diện chơi game: 42](#_Toc388397881)

[3.4. Xây dựng giao diện kết nối mạng LAN: 49](#_Toc388397882)

[3.5. Xây dựng giao điện chơi game qua mạng LAN: 53](#_Toc388397883)

[3.6. Xây dựng đối tượng kết nối các cảnh game lại: 54](#_Toc388397884)

[4. Âm thanh: 56](#_Toc388397885)

[4.1. Xây dựng đối tượng phát âm thanh: 56](#_Toc388397886)

[4.2. Xây dựng nút tắt/ mở âm thanh trong game: 57](#_Toc388397887)

[4.3. Xây dựng đối tượng chơi nhạc nền: 59](#_Toc388397888)

[4.4. Xây dựng đối tượng tắt mở nhạc nền: 63](#_Toc388397889)

[5. Kết luận: 65](#_Toc388397890)

# Giới thiệu chung:

Cờ caro thuộc thể loại game đối kháng giữa 2 người chơi. Hai người chơi sẽ đi theo lượt, mỗi lượt đi một quân cờ vào ô còn trống. Người chơi chơi có 5 quân cờ liên tiếp theo hàng ngang, hàng dọc hoặc hàng chéo sẽ chiến thắng.

Trong đề tài này, game caro được lập trình bao gồm 3 chế độ chơi game khác nhau :

* 1 Player : Người chơi đầu cờ với máy tính.
* 2 Player: Hai người chơi đấu cờ với nhau trên cùng một máy tính.
* LAN: Hai người chơi kết nối mạng LAN đánh cờ với nhau.

# Thuật toán:

## Thuật toán tìm đường đi của máy tính:

Sử dụng phương pháp tính điểm.

Có hai ma trận điểm sau:

**public** **int**[ ] DScore = **new** **int**[5] { 0, 1, 9, 81, 729 };*//Mảng điểm chặn*

**public** **int**[ ] AScore = **new** **int**[5] { 0, 2, 18, 162, 1458 };*//Mảng điểm tấn công*

Tiến hành quét một bộ 5 ô theo 4 hướng là dọc, ngang, chéo lên, chéo xuống, đếm số quân cờ mỗi bên và tiến hành cộng điểm cho các ô trống khi trong bộ 5 ô đó chỉ có toàn quân ta, hoặc toàn quân địch.

Nếu có n quân ta thì các ô trống được cộng Ascore[n] điểm.

Nếu có n quân địch thì các ô trống được cộng Dscore[n] điểm.

Sau đó, ta chọn ra ô cao điểm nhất để đi.

Ví dụ: ta có bàn cờ 8x8 như sau:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  | X |  |  |  |  |  |
| 2 |  | **O** | **O** | **O** |  |  |  |  |
| 3 |  |  |  |  | X | O | X |  |
| 4 |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |

Ta cần tính nước đi tiếp theo cho máy tính chính là tính nước đi cho quân O.

Quân ta : quân O

Quân địch: quân X

Khối các ô ngang từ [1,0] đến [1,4] có 1 quân địch nên mỗi ô còn trống được công DScore[1] = 1 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 1 | **X** | 1 | 1 |  |  |  |
| *2* |  | O | O | O |  |  |  |  |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [1,1] đến [1,5] có 1 quân địch nên mỗi ô còn trống được cộng thêm DScore[1] = 1 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 2 | 2 | 1 |  |  |
| *2* |  | O | O | O |  |  |  |  |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [1,2] đến [1,6] có 1 quân địch nên mỗi ô còn trống được cộng thêm DScore[1] = 1 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 3 | 3 | 2 | 1 |  |
| *2* |  | O | O | O |  |  |  |  |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [2,0] đến [2,4] có 3 quân ta nên mỗi ô còn trống được cộng thêm AScore[3] = 162 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 3 | 3 | 2 | 1 |  |
| *2* | 162 | O | O | O | 162 |  |  |  |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [2,1] đến [2,5] có 3 quân ta nên mỗi ô còn trống được cộng thêm AScore[3] = 162 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 3 | 3 | 2 | 1 |  |
| *2* | 162 | O | O | O | 324 | 162 |  |  |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [2,2] đến [2,6] có 2 quân ta nên mỗi ô còn trống được cộng thêm AScore[2] = 18 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 3 | 3 | 2 | 1 |  |
| *2* | 162 | O | O | O | 342 | 180 | 18 |  |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [2,3] đến [2,7] có 1 quân ta nên mỗi ô còn trống được cộng thêm AScore[2] = 2 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 3 | 3 | 2 | 1 |  |
| *2* | 162 | O | O | O | 344 | 182 | 20 | 2 |
| *3* |  |  |  |  | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Khối các ô ngang từ [3,0] đến [3,5] có 1 quân địch nên mỗi ô còn trống được cộng thêm AScore[2] = 1 điểm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| *0* |  |  |  |  |  |  |  |  |
| *1* | 1 | 2 | X | 3 | 3 | 2 | 1 |  |
| *2* | 162 | O | O | O | 344 | 182 | 20 | 2 |
| *3* | 1 | 1 | 1 | 1 | X | O | X |  |
| *4* |  |  |  |  |  |  |  |  |
| *5* |  |  |  |  |  |  |  |  |
| *6* |  |  |  |  |  |  |  |  |
| *7* |  |  |  |  |  |  |  |  |

Ta thấy ô [2,4] có điểm cao nhất nên nước đi tiếp theo là ô [2,4]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  | X |  |  |  |  |  |
| 2 |  | **O** | **O** | **O** | **O** |  |  |  |
| 3 |  |  |  |  | X | O | X |  |
| 4 |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |

|  |
| --- |
| Computer.java |
| package caro;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** Computer {  **private** **int** height;  **private** **int** width;  **public** **int** optimalX;  **public** **int** optimalY;  **private** EvalBoard myEvalBoard;  **public** Computer(**int** height, **int** width) {  **this**.height = height;  **this**.width = width;    myEvalBoard = **new** EvalBoard(height, width);  }  **public** **void** calculateEvalBoard(**int** player, **int**[][] status) {  **int**[] DScore = **new** **int**[]{0, 1, 9, 81, 729};  **int**[] AScore = **new** **int**[]{0, 2, 18, 162, 1458};  **int** row, col, ePC, eHuman;  myEvalBoard.ResetBoard();  *//Đánh giá theo hàng ngang*  **for** (row = 0; row < height; row++) {  **for** (col = 0; col < width - 4; col++) {  ePC = 0;  eHuman = 0;  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row][col + i] == 1) {  eHuman++;  }  **if** (status[row][ col + i] == 2) {  ePC++;  }  }  **if** (eHuman \* ePC == 0 && eHuman != ePC) *// một trong 2 bằng 0 và không đồng thời bằng 0*  {  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row][col + i] == 0) *// Nếu ô chưa đánh*  {  **if** (eHuman == 0) {  **if** (player == 1) {  myEvalBoard.EBoard[row][ col + i] += DScore[ePC];  } **else** {  myEvalBoard.EBoard[row][ col + i] += AScore[ePC];  }  }  **else** **if** (ePC == 0) {  **if** (player == 2) {  myEvalBoard.EBoard[row][col + i] += DScore[eHuman];  } **else** {  myEvalBoard.EBoard[row][col + i] += AScore[eHuman];  }  }  **if** (eHuman == 4 || ePC == 4) {  myEvalBoard.EBoard[row][col + i] \*= 2;  }  }  }  }  }  }  *//Đánh giá theo cột*  **for** (col = 0; col < width; col++) {  **for** (row = 0; row < height - 4; row++) {  ePC = 0;  eHuman = 0;  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row + i][col] == 1) {  eHuman++;  }  **if** (status[row + i][col] == 2) {  ePC++;  }  }  **if** (eHuman \* ePC == 0 && eHuman != ePC) {  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row + i][col] == 0) *// cộng điểm cho các ô chưa đánh*  {  **if** (eHuman == 0) {  **if** (player == 1) {  myEvalBoard.EBoard[row + i][col] += DScore[ePC];  } **else** {  myEvalBoard.EBoard[row + i][col] += AScore[ePC];  }  }  **if** (ePC == 0) {  **if** (player == 2) {  myEvalBoard.EBoard[row + i][ col] += DScore[eHuman];  } **else** {  myEvalBoard.EBoard[row + i][col] += AScore[eHuman];  }  }  **if** (eHuman == 4 || ePC == 4) {  myEvalBoard.EBoard[row + i][col] \*= 2;  }  }  }  }  }  }  *//Đánh giá theo đường chéo chính*  **for** (col = 0; col < width - 4; col++) {  **for** (row = 0; row < height - 4; row++) {  ePC = 0;  eHuman = 0;  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row + i][col + i] == 1) {  eHuman++;  }  **if** (status[row + i][col + i] == 2) {  ePC++;  }  }  **if** (eHuman \* ePC == 0 && eHuman != ePC) {  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row + i][col + i] == 0) *// Neu o chua duoc danh*  {  **if** (eHuman == 0) {  **if** (player == 1) {  myEvalBoard.EBoard[row + i][col+i]+= DScore[ePC];  } **else** {  myEvalBoard.EBoard[row +i][col+i]+= AScore[ePC];  }  }  **if** (ePC == 0) {  **if** (player == 2) {  myEvalBoard.EBoard[row+i][col+i]+= DScore[eHuman];  } **else** {  myEvalBoard.EBoard[row+i][col+i]+= AScore[eHuman];  }  }  **if** (eHuman == 4 || ePC == 4) {  myEvalBoard.EBoard[row + i][col + i] \*= 2;  }  }  }  }  }  }  *//Đánh giá theo đường chéo phụ*  **for** (row = 4; row < height; row++) {  **for** (col = 0; col < width - 4; col++) {  ePC = 0;  eHuman = 0;  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row - i][col + i] == 1) {  eHuman++;  }  **if** (status[row - i][col + i] == 2) {  ePC++;  }  }  **if** (eHuman \* ePC == 0 && eHuman != ePC) {  **for** (**int** i = 0; i < 5; i++) {  **if** (status[row - i][col + i] == 0) *// Neu o chua duoc danh*  {  **if** (eHuman == 0) {  **if** (player == 1) {  myEvalBoard.EBoard[row - i][col+i]+= DScore[ePC];  } **else** {  myEvalBoard.EBoard[row -i][col +i]+= AScore[ePC];  }  }  **if** (ePC == 0) {  **if** (player == 2) {  myEvalBoard.EBoard[row-i][col+i]+= DScore[eHuman];  } **else** {  myEvalBoard.EBoard[row-i][col+i]+= AScore[eHuman];  }  }  **if** (eHuman == 4 || ePC == 4) {  myEvalBoard.EBoard[row - i][col + i] \*= 2;  }  }  }  }  }  }  }  *//Ham tim nuoc di cho may*  **public** **void** FindMove(**int**[][] status) {  calculateEvalBoard(2, status);  Point temp = myEvalBoard.MaxPos();    optimalX = temp.x;  optimalY = temp.y;  }  } |

## Thuật toán kiểm tra chiến thắng:

Ta xây dựng class **Check.java** để kiểm tra nước đi có chiến thắng không.

boolean **isDraw**(int[][] status): kiểm tra game có hòa hay không. Game hòa khi không còn ô trống trên bàn cờ mà chưa có người chiến thắng.

boolean **rowCheck**(int row, int col, int[][] status, int player) : Kiểm tra theo hàng ngang nước đi vị trí dòng thứ row, cột thứ col của người chơi thứ player có chiến thắng hay không? Bảng status là một mảng 2 chiều. chứa các giá trị 0,1,2 dùng để xác định trạng thái ô là chưa đánh, ô của quân 1 hoặc ổ của quân 2.

Để kiểm tra theo hàng ngang, ta chỉ xét trên hàng row. Nếu có 5 ô liên tiếp có cùng giá trị player thì người chơi chiến thắng.

boolean **columnCheck**(int row, int col, int[][] status, int player) : Kiểm tra nước đi chiến thắng của hàng chéo bên trái chứa ô có địa chỉ [row,col].

Tương tự như kiểm tra hàng ngang, ta chỉ việc kiểm tra cột thư col xem có 5 ô liên tiếp có cùng giá trị player hay không.

boolean **leftCheck**(int row, int col, int[][] status, int player): Kiểm tra nước đi chiến thắng của hàng chéo bên trái chứa ô có địa chỉ [row,col].

boolean **rightCheck**(int row, int col, int[][] status, int player): Kiểm tra nước đi chiến thắng của hàng chéo bên phải chứa ô có địa chỉ [row,col].

boolean **checkIt**(int row, int col, int[][] status, int player): Thực hiện kiểm tra nước đi theo hàng ngang, dọc, đường chéo chính và đường chéo phụ của player. Có nghĩa là thực hiện 4 phương thức rowCheck, columnCheck, leftCheck và rightCheck.

row

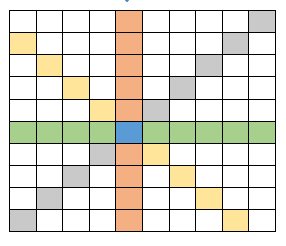
col

🗖 rowCheck

🗖 columCheck

🗖 leftCheck

🗖 rightCheck



|  |
| --- |
| **Check.java** |
| package caro;  **public** **class** Check {  **private** **int** height;  **private** **int** width;  **public** Check(**int** height, **int** width) {  **this**.height = height;  **this**.width = width;  }  **public** boolean isDraw(**int**[][] status) {  **for** (**int** row = 0; row < height; row++) {  **for** (**int** col = 0; col < width; width++) {  **if** (status[row][col] == 0) {  **return** false;  }  }  }  **return** true;  }  **public** boolean rowCheck(**int** row, **int** col, **int**[][] status, **int** player) {  **int** count = 0;  **for** (**int** i = 0; i < width; i++) {  **if** (status[row][i] == player) {  count++;  **if** (count >= 5) {  **return** true;  }  } **else** {  count = 0;  }  }  **return** false;  }  **public** boolean columnCheck(**int** row, **int** col, **int**[][] status, **int** player) {  **int** count = 0;  **for** (**int** i = 0; i < height; i++) {  **if** (status[i][col] == player) {  count++;  **if** (count >= 5) {  **return** true;  }  } **else** {  count = 0;  }  }  **return** false;  }  **public** boolean leftCheck(**int** row, **int** col, **int**[][] status, **int** player) {  **int** count = 0;  **try** {  **while** (col > 0 && row > 0) {  col--;  row--;  }  **while** (col < width && row < height) {  **if** (status[row][col] == player) {  count++;  **if** (count >= 5) {  **return** true;  }  } **else** {  count = 0;  }  col++;  row++;  }  } **catch** (Exception e) {  }  **return** false;  }  **public** boolean rightCheck(**int** row, **int** col, **int**[][] status, **int** player) {  **int** count = 0;  **try** {  **while** (col > 0 && row < 15) {  col--;  row++;  }  **while** (col <= 31 && row >= 0) {  **if** (status[row][col] == player) {  count++;  **if** (count >= 5) {  **return** true;  }  } **else** {  count = 0;  }  col++;  row--;  }  } **catch** (Exception e) {  }  **return** false;  }  **public** boolean checkIt(**int** row, **int** col, **int**[][] status, **int** player) {  **if** (rowCheck(row, col, status, player)) {  **return** true;  }  **if** (columnCheck(row, col, status, player)) {  **return** true;  }  **if** (leftCheck(row, col, status, player)) {  **return** true;  }  **if** (rightCheck(row, col, status, player)) {  **return** true;  }  **return** false;  }  } |

## Thuật toán xây dựng kết nối mạng LAN:

### Ý tưởng:

Sử dụng stream socket để kết nội mạng LAN.

### Xây dựng đối tượng Server:

Ta tạo class **Server.java** là tượng đóng vai trò Server kết nối và xử lí :

* Nhận thông tin nước đi từ Client, xét xem nước đi có hợp lệ.
* Kiểm tra xem kết thúc game chưa.

Ta cần tạo một Thread để chạy ServerSocket song song với chương trình.

Định nghĩa một class **Player** trong class Server.java. Class Player kế thừa Thread, có khả năng hoạt động song song với Server, có thuộc tính Client (đề cập nên dưới) để tiếp nhận thông tin từ Server.

### Xây dựng đối tượng Client:

Xây dựng class **Client.java** có chức năng:

* Gửi thông tin về vị trí đến server
* Hiện thị tin nhắn từ server trên một TextArea.
* Hiện thị bàn cờ.
* Hiện thị kết quả trận đấu khi kết thúc game.

Thread: Sever

Vị trí

Vị trí hợp lệ?

Kết thúc?

Kết quả

Thông báo

Hiển thị

Kết thúc Server

Thoát

Thread: Player X

Thread: Player O

Client

Hình 1 Thread Server giao tiếp với Thread Player X

Thread: Sever

Vị trí

Vị trí hợp lệ?

Kết thúc?

Kết quả

Thông báo

Hiển thị

Kết thúc Server

Thoát

Thread: Player X

Thread: Player O

Client

Hình 2 Thread Server giao tiếp với Thread Player O

### Mô tả hoạt động của Server và Client:

#### Server kết nối với Client X:

Thread: Sever

WELCOME X

MESSAGE Wait another connects

Khởi tạo gane

Hiện thông báo

Thread: Player X

Thread: Player O

Client

#### Server kết nối với Client O:

Kết nối này tương tự như kết nối Client X.

Thread: Sever

WELCOME O

MESSAGE Wait another connects

Khởi tạo gane

Hiện thông báo

Thread: Player X

Thread: Player O

Client

#### Tất cả người chơi kết nối:

play

Thread: Sever

MESSAGE First turn

wait

Client

MESSAGE All player connected

Client

Thread : Player X

Thread : Player O

#### Client gửi thông tin nước đi đến Server:

Client gửi thông tin nước đi đến Server. Server xem nước đi đó có hợp lệ không? Nước đi hợp lệ là nước đi chưa bị chiếm và Client phải đang ở lượt đi của mình.

Nếu đúng là ở lượt chơi của mình mà đánh vào ô đã bị chiếm, Server sẽ bỏ qua và tiếp tục chờ vị trí khác đến từ Client hiện hành.

Nếu nước đi hợp lệ, Server gửi thông báo đến Client và hiện thị vị trí trên bàn cờ. Client gửi thông báo vị trí đến “Oponent Client” và chuyển lượt đi cho “Oponent Client”.

Ở đây ta lấy ví dụ đang ở lượt đi của Client X.

Hợp lệ ?

no

Thread: Sever

Client X

Client O

MOVE location

VALID\_MOVE

Hiện thị

Hiện thị

Win or Lose or Tie ?

yes

Kết thúc?

yes

no

Chuyển lượt

### Xây dựng code cho class Server.java và Client.java

|  |
| --- |
| Server.java |
| package caro;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.ServerSocket;  import java.net.Socket;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** Server {  */\**  *\*Chạy chương trình . Cho Client kết nối đến Server*  *\*/*  **public** ServerSocket listener;    **public** Thread serverThread;      **public** Runnable serverAccept = **new** Runnable() {  @Override  **public** **void** run() {  **try** {  System.out.println(**"Tic Tac Toe Server is Running"**);  **while** (true) {  Game game = **new** Game();  Game.Player playerX = game.**new** Player(listener.accept(), **'X'**);  Game.Player playerO = game.**new** Player(listener.accept(), **'O'**);    playerX.setOpponent(playerO);  playerO.setOpponent(playerX);  game.currentPlayer = playerX;  playerX.start();  playerO.start();    }  } **catch** (Exception ex) {  System.out.println(**"Bỏ qua lỗi : tồn tại server "**);      } finally {  **try** {  listener.close();  } **catch** (Exception e) {  }  }  }  };  **public** Server() {    **try** {  listener = **new** ServerSocket(8901);    } **catch** (Exception ex) {  }  serverThread = **new** Thread(serverAccept);  serverThread.start();  }    */\*\**  *\* A two-player game.*  *\*/*  **public** **class** Game {  */\**  *\* Một bảng gồm 16 x 16 ô . Mỗi ô là một player.*  *\**  *\*/*  **private** Player[][] board = **new** Player[16][16];  **private** **int**[][] status = **new** **int**[16][16];  **public** Game() {  **for** (**int** row = 0; row < 16; row++) {  **for** (**int** col = 0; col < 16; col++) {  board[row][col] = null;  status[row][col] = 0;  }  }  }    Player currentPlayer; *// người chơi hiện tại đang có quyền đi*  */\**  *\**  *\* Kiểm tra người chơi hiện tại có chiến thắng hay không?*  *\*/*  **public** boolean hasWinner(**int** row, **int** col) {  Check myCheck = **new** Check(16, 16);  **int** prePlayer = (currentPlayer.mark == **'X'** ? 2 : 1);  System.out.println(**"Player "** + prePlayer + **"Win?"** + myCheck.checkIt(row, col, status, prePlayer));  **if** (myCheck.checkIt(row, col, status, prePlayer)) {    **return** true;    } **else** {  **return** false;  }  }  */\**  *\* Kiểm tra bảng có còn ô trống nào không ?*  *\*/*  **public** boolean boardFilledUp() {  **for** (**int** row = 0; row < 16; row++) {  **for** (**int** col = 0; col < 16; col++) {  **if** (board[row][col] == null) {  **return** false;  }  }  }  **return** true;  }  */\**  *Phương thức được gọi bởi phayer Thread khi một người chơi cố gắng thực*  *hiện một nước đi.*  *Phương thức này kiểm tra xem nước đi có hợp lệ không. Nước đi hợp lệ*  *phải là ô còn trống và người chơi đến lượt đi của mình. Nếu hợp lệ, ô cờ*  *sẽ cặp nhật lại và chuyển lượt chơi ( currentPlayer = opponent)*  *\*/*  **public** synchronized boolean legalMove(**int** location, Player player) {  System.out.println(**"location "** + location);  **int** row = location / 16;  **int** col = location % 16;  **if** (player == currentPlayer && board[row][col] == null ) {  board[row][col] = currentPlayer;  status[row][col] = (currentPlayer.mark == **'X'** ? 1 : 2);  currentPlayer = currentPlayer.opponent;  currentPlayer.otherPlayerMoved(location);  **return** true;  }  **return** false;  }    */\**  *Mỗi người chơi sẽ là một Thread. Mỗi người chơi sẽ có kí tự đánh dấu*  *là 'X' hoặc 'O'. Để giao tiếp được, mỗi Player có socket với input*  *và output stream.*  *\*/*  **class** Player extends Thread {  **char** mark;  Player opponent;  Socket socket;  BufferedReader input;  PrintWriter output;  */\**  *Construct sẽ khởi tạo socket và kí tự đánh dấu.*  *Khởi tạo các luồng input và output qua mạng LAN.*  *Gửi tin nhắn "Welcome" đế người chơi.*  *\*/*  **public** Player(Socket socket, **char** mark) {  **this**.socket = socket;  **this**.mark = mark;  **try** {  input = **new** BufferedReader(  **new** InputStreamReader(socket.getInputStream()));  output = **new** PrintWriter(socket.getOutputStream(),true);  System.out.println(**"sent: "** + **"WELCOME "** + mark);  output.println(**"WELCOME "** + mark);  output.println(**"MESSAGE Wait another connects"**);      } **catch** (IOException e) {  System.out.println(**"Player died: "** + e);  }  }  */\**  *\* Phương thức thiết lập đối thủ cho một Player*  *\*/*  **public** **void** setOpponent(Player opponent) {  **this**.opponent = opponent;  }  */\**  *\* Xử lí tin nhắn : otherPlayerMoved .*  *\*/*  **public** **void** otherPlayerMoved(**int** location) {  System.out.println(**"sent: OPPONENT\_MOVED "** + location);  output.println(**"OPPONENT\_MOVED "** + location);  System.out.println(  hasWinner(location / 16, location % 16) ? **"DEFEAT"** : boardFilledUp() ? **"TIE"** : **""**);  output.println(  hasWinner(location / 16, location % 16) ? **"DEFEAT"** : boardFilledUp() ? **"TIE"** : **""**);  }  */\**  *phương thức run của thread.*  *\*/*  **public** **void** run() {  **try** {  *// The thread bắt đầu khi cả hai người chơi đã kết nối*  output.println(**"MESSAGE All is connected"**);    Client.isStartGame = true ;    *// Bảo người chơi thứ nhất Đi lượt đầu tiên*  **if** (mark == **'X'**) {  output.println(**"MESSAGE First turn"**);      }    *// Serve: Nhận các yêu cầu(commands) từ client và xử lí chúng*  **while** (true) {  String command = input.readLine();  **if** (command.startsWith(**"MOVE"**) ) {  **int** location = Integer.parseInt(command.substring(5));  **if** (legalMove(location, **this**)) {  output.println(**"VALID\_MOVE"**);  output.println(hasWinner(location / 16, location % 16) ? **"VICTORY"**  : boardFilledUp() ? **"TIE"**  : **""**);  } **else** {  output.println(**"MESSAGE This is not your turn"**);  }  } **else** **if** (command.startsWith(**"QUIT"**)) {    **return**;  }  }  } **catch** (IOException e) {  System.out.println(**"Player died: "** + e);  } finally {  **try** {  socket.close();  } **catch** (IOException e) {  }  }  }  }  }  } |
|  |
| Client.java |
| Client.java |
| package caro;  import java.awt.event.MouseAdapter;  import java.awt.event.MouseEvent;  import java.io.BufferedReader;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.Socket;  import javax.swing.Icon;  import javax.swing.ImageIcon;  import javax.swing.JLabel;  import javax.swing.JOptionPane;  */\**  *\**  *\* @author VanNinh*  *\**  *\* Client gửi các tin nhắn đến server:*  *MOVE*  *QUIT*  *\* Client tiếp nhận thông tin từ Server:*  *MESSAGE*  *VICTORY / DEFEAT /TIE*  *\*/*  **public** **class** Client {  */\**  *\* Các thuộc tính đồ họa của game*  *\*/*  **private** JLabel messageLabel = **new** JLabel(**""**);  **private** String URLIcon;  **private** String URLOpenentICon;  **public** ImagePanel background = **new** ImagePanel(**"picture/main.png"**, 0, 0, 800, 600);  **public** ImagePanel waitingPicture = **new** ImagePanel(**"picture/waiting2.png"**, 500, 350, 300, 225);  **public** ImagePanel attackPicture = **new** ImagePanel(**"picture/attack1.png"**, 500, 350, 300, 225);  **public** ImagePanel waitingConnectPicture = **new** ImagePanel(**"picture/WaitingConnect.png"**, 500, 350, 300, 225);  **private** ImagePanel[][] board = **new** ImagePanel[16][16];  **private** ImagePanel currentSquare;  **public** **static** BackButton myBackButton;  **public** SoundPlayer mySoundPlayer = **new** SoundPlayer();  */\**  *Các thuộc tính kết nối server :*    *\*/*  **private** **static** **int** PORT = 8901;  **private** Socket socket;  **private** BufferedReader input;  **private** PrintWriter output;  */\**  *\* Các thuộc tính xét trạng thái game*  *\**  *\*/*  **public** **static** boolean isStartGame;  **char** mark;  **public** Client(String serverAddress) {  **try** {  isStartGame = false;  *// Setup networking*  socket = **new** Socket(serverAddress, PORT);  input = **new** BufferedReader(**new** InputStreamReader(  socket.getInputStream()));  output = **new** PrintWriter(socket.getOutputStream(), true);  } **catch** (Exception ex) {  }  *// Layout GUI*  ImagePanel messagePanel = **new** ImagePanel(**"picture/message.png"**, 20, 500, 480, 40);  messageLabel.setBounds(20, 4, 300, 30);  messagePanel.add(messageLabel);  background.add(messagePanel);  myBackButton = **new** BackButton(**"towLanPlayerPanel"**);  myBackButton.setEnabled(false);  background.add(myBackButton);  background.repaint();  *// thêm vào waitting picture*  background.add(waitingConnectPicture);  ImagePanel boardPanel = **new** ImagePanel(**"picture/table.png"**, 20, 20, 480, 480);  boardPanel.setLayout(null);  **for** (**int** row = 0; row < 16; row++) {  **for** (**int** col = 0; col < 16; col++) {  final **int** k = row;  final **int** l = col;  board[row][col] = **new** ImagePanel(**"picture/khung.png"**, col \* 30, row \* 30, 30, 30);  board[row][col].repaint();  background.repaint();  board[row][col].addMouseListener(**new** MouseAdapter() {  @Override  **public** **void** mouseClicked(MouseEvent e) {  *// play sound*  **if** (GamePanel.canPlaySound) {  mySoundPlayer.playSound(**"sound/kick.mp3"**);  }  currentSquare = board[k][l];  System.out.println(**"Client : isStartGame = "** + isStartGame);  **if** (isStartGame) {  System.out.println(**"MOVE "** + (k \* 16 + l));  output.println(**"MOVE "** + (k \* 16 + l));  }  }  });  boardPanel.add(board[row][col]);  board[row][col].repaint();  background.repaint();  }  }  background.add(boardPanel);  }  */\* phương thức của Client để lắng nghe tin nhắn từ Server.*  *\*Tin nhắn đầu tiên là "WELCOME" và đồng thời nhận được kí tự đánh dấu.*  *\*Sử dụng vòng lặp vô hạn để lắng nghe các tin nhắn "VALID\_MOVE",*  *"OPPONENT\_MOVED", "VICTORY", "DEFEAT", "TIE", "OPPONENT\_QUIT hoặc "MESSAGE".*  *\*Các tin "VICTORY", "DEFEAT" và "TIE" là các tính hiệu kết thúc game. Khi đó,*  *chương trình hỏi người chơi có muốn chơi lại hay không. Nếu không thì Server*  *gửi tin nhắn "QUIT" để thoát game và "OPPONENT\_QUIT" để đối thủ cũng thoát game.*  *\*/*  **public** **void** play() {  String response;  **try** {  response = input.readLine();  **if** (response.startsWith(**"WELCOME"**)) {  mark = response.charAt(8);  URLIcon = (mark == **'X'** ? **"picture/khung1.png"** : **"picture/khung2.png"**);  URLOpenentICon = (mark == **'X'** ? **"picture/khung2.png"** : **"picture/khung1.png"**);  **if** (mark == **'X'**) {  waitingPicture.setPicture(**"picture/waiting2.png"**);  attackPicture.setPicture(**"picture/attack1.png"**);  } **else** **if** (mark == **'O'**) {  waitingPicture.setPicture(**"picture/waiting1.png"**);  attackPicture.setPicture(**"picture/attack2.png"**);  isStartGame = true;  background.remove(waitingConnectPicture);  background.add(waitingPicture);  }  }  **while** (true) {  response = input.readLine();  background.repaint();  Main.myFrame.repaint();  **if** (response.startsWith(**"VALID\_MOVE"**)) {  currentSquare.setPicture(URLIcon);  background.remove(attackPicture);  background.add(waitingPicture);  messageLabel.setText(**"Bạn đã đi, đợi tí"**);  background.repaint();  Main.myFrame.repaint();  } **else** **if** (response.startsWith(**"OPPONENT\_MOVED"**)) {  background.remove(waitingPicture);  background.add(attackPicture);  **int** location = Integer.parseInt(response.substring(15));  **int** row = location / 16;  **int** col = location % 16;  board[row][col].setPicture(URLOpenentICon);  background.repaint();  Main.myFrame.repaint();  myBackButton.setEnabled(true); *// kích hoạt nút thoát ra*  messageLabel.setText(**"Đối thủ đã đi , đến lượt bạn"**);  } **else** **if** (response.startsWith(**"VICTORY"**)) {  *// âm thanh*  **if** (GamePanel.canPlaySound) {  mySoundPlayer.playSound(**"sound/win.mp3"**);  }    *// bảng thông báo*  Icon myIcon = **new** ImageIcon(getClass().getResource(**"picture\\winner"** + (mark == **'X'** ? **'1'** : **'2'**) + **".gif"**));  JOptionPane.showMessageDialog(null, null, **"Chiến thắng!"**, JOptionPane.INFORMATION\_MESSAGE, myIcon);    messageLabel.setText(**"Bạn giành được thắng lợi"**);  Main.myFrame.repaint();  **break**;  } **else** **if** (response.startsWith(**"DEFEAT"**)) {  messageLabel.setText(**"Bạn đã bị đánh bại"**);  *// âm thanh*  **if** (GamePanel.canPlaySound) {  mySoundPlayer.playSound(**"sound/GameOver.mp3"**);  }    *// bảng thông báo*  Icon myIcon = **new** ImageIcon(getClass().getResource(**"picture\\loser"** + (mark == **'X'** ? **'1'** : **'2'**) + **".gif"**));  JOptionPane.showMessageDialog(null, null, **"Thua!"**, JOptionPane.INFORMATION\_MESSAGE, myIcon);  Main.myFrame.repaint();  **break**;  } **else** **if** (response.startsWith(**"TIE"**)) {  messageLabel.setText(**"Hòa nhau"**);  Main.myFrame.repaint();  **break**;  } **else** **if** (response.startsWith(**"MESSAGE"**)) {  **if** (**"MESSAGE First turn"**.equals(response)) {  background.remove(waitingConnectPicture);  background.add(attackPicture);  messageLabel.setText(**"Game đã bắt đầu. Bạn hãy đi một ô"**);  } **else** **if** (**"MESSAGE All is connected"**.equals(response)) {  isStartGame = true;  messageLabel.setText(**"Tất cả người chơi đã kết nối"**);  } **else** **if** (**"MESSAGE This is not your turn"**.equals(response)) {  messageLabel.setText(**"Chưa đến lượt chơi của bạn. Đợi tí"**);  } **else** **if** (**"MESSAGE Wait another connects"**.equals(response)) {  messageLabel.setText(**"Đợi người chơi khác kết nối"**);  } **else** {  messageLabel.setText(response.substring(8));  }  Main.myFrame.repaint();  }  }  output.println(**"QUIT"**);  } **catch** (Exception ex) {  } finally {  **try** {  socket.close();  } **catch** (Exception ex) {  }  }  }  */\**  *\* The methord: Do you want restart game?*  *\*/*  **public** boolean wantsToPlayAgain() {  **int** response = JOptionPane.showConfirmDialog(null,  **"Bạn muốn chơi lại lần nữa không ?"**,  **"Game Carô"**,  JOptionPane.YES\_NO\_OPTION);  isStartGame = true;  **return** response == JOptionPane.YES\_OPTION;  }  */\*\**  *\* Runs the client as an application.*  *\*/*    } |

# Đồ họa:

## Xây dựng đối tượng JPanel có khả năng chén ảnh:

Ta xây dựng class **ImagePanel.java** kế thừa class JPanel của thư viện swing có khả năng thiết lập và thay đổi hình ảnh từ một đường dẫn cho trước. Bao gồm:

Image myImage : đối tượng kiểu Image trong thư viện java.awt.Image có khả vẽ hình ảnh.

void paintComponent(Graphics g): phương thức trong java.awt.Graphics sẽ tiến hành vẽ một đối tượng Image trên JPanel.

void setPicture(String nameFile) : được định nghĩa để thay đổi hình ảnh.

void paintComponent(Graphics g): hàm được Java định nghĩa sãn trong lớp trừa tượng Jcomponent. Phương thức này có chức năng vẽ đối tượng Graphics trên một component, chẳng hạn như JPanel.

Lưu ý: Hình ảnh phải để trong thư mục src trong project (đây là thư mục chứa file \*.java)

|  |
| --- |
| ImagePanel.java |
| package caro;  import java.awt.Graphics;  import java.awt.Image;  import javax.swing.ImageIcon;  import javax.swing.JPanel;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  public class ImagePanel extends JPanel {  private int width;  private int height;  private Image myImage;  public ImagePanel(String nameFile, int x, int y, int width, int height) {  this.width = width;  this.height = height;  String s = getClass().getResource(nameFile).toString();  this.myImage = new ImageIcon(getClass().getResource(nameFile)).getImage();  *// hình ảnh phải bỏ trong thư mục src của project*  setLayout(null);  this.setBounds(x, y, width, height);  repaint();  }  public void setPicture(String nameFile) {  this.myImage = new ImageIcon(getClass().getResource(nameFile)).getImage();  }  @Override  public void paintComponent(Graphics g) {  g.drawImage(myImage, 0, 0, this);  }  } |

## Xây dựng giao diện menu chính của game:

Xây dựng class **StartPanel.java** để hiện thị các chế độ chơi game :

* 1 Player : 1 người chơi đấu cờ với máy .
* 2 Player : 2 người chơi trên cùng một máy.
* LAN GAME : 2 người chơi thông qua 2 máy kết nối mạng LAN .
* Exit: thoát game.

Để xây dựng class này, ta xây dựng một lớp kế thừa JPanel từ thư viện swing, bao gồm các đối tượng sau:

* Hình nền: ta tạo đối tượng **background** là đối tượng kiểu ImagePanel ( đối tượng do người lập trình định nghĩa)để tải ảnh nền.
* Các nút chức năng: **oneButton**, **twoButton**, **LANButton**, **exitButton** là đối tượng kiểu Button của thư viện swing có chức năng truy cập đến các chế độ game khác nhau.

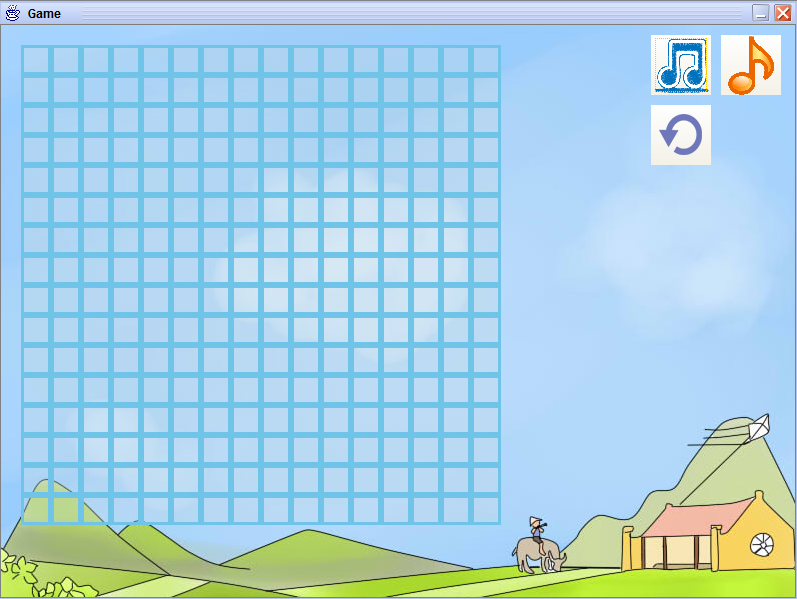


|  |
| --- |
| **StartPanel.java** |
| package caro;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.JButton;  import javax.swing.JPanel;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** StartPanel extends JPanel {  SoundPlayer mySound = **new** SoundPlayer();  **public** StartPanel() {  setLayout(null);  setBounds(0, 0, 800, 600);  */\* hình nền cho Game :*  *đối tượng tự tạo ImagePanel - một panel có chức năng load ảnh \*/*  ImagePanel background = **new** ImagePanel(**"picture/StartGame.png"**, 0, 0, 800, 600);  *//Thêm các button chức năng*  JButton oneButton = **new** JButton(**"1 Player"**);  JButton twoButton = **new** JButton(**"2 Players"**);  JButton LANButton = **new** JButton(**"LAN game"**);  JButton exitButton = **new** JButton(**"Exit"**);  *// định vị trí các button*  oneButton.setBounds(350, 300, 100, 30);  twoButton.setBounds(350, 350, 100, 30);  LANButton.setBounds(350, 400, 100, 30);  exitButton.setBounds(350, 450, 100, 30);  *// thêm hành động cho button "1 player" : người chơi đấu với máy*  oneButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent ae) {  **if** (GamePanel.canPlaySound) {  mySound.playSound(**"sound/coin.mp3"**); *// phát âm thanh*  }  *// chương trình gỡ bỏ chế độ START MENU, thay vào là chế độ GAME 1 người chơi*  Main.myFrame.remove(Main.myStartPanel);  Main.myGamePanel = **new** GamePanel();  Main.myGamePanel.numberPlayer = 1; *// số người chơi là một*  Main.myFrame.add(Main.myGamePanel);  Main.myFrame.repaint();  }  });    *// thêm hành động cho button "2 player" : chế độ 2 người chơi trên cùng một máy*  twoButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent ae) {  **if** (GamePanel.canPlaySound) {  mySound.playSound(**"sound/coin.mp3"**); *// âm thanh*  }  */\* chương trình gỡ bỏ chế độ START MENU :*  *thay vào là chế độ GAME 2 người chơi \*/*  Main.myFrame.remove(Main.myStartPanel);  Main.myGamePanel = **new** GamePanel();  Main.myGamePanel.numberPlayer = 2;  Main.myFrame.add(Main.myGamePanel);  Main.myFrame.repaint();  }  });  */\* thêm hành động cho LAN GAME button :*  *2 người chơi kết nối 2 máy qua mạng LAN \*/*  LANButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent ae) {  **if** (GamePanel.canPlaySound) {  mySound.playSound(**"sound/coin.mp3"**); *// âm thanh*  }  */\* chương trình gỡ bỏ chế độ START MENU,*  *thay vào là menu kết nối mạng LAN*  *\*/*  Main.myFrame.remove(Main.myStartPanel);  NetworkPanel.joinButton.setEnabled(true);  Main.myFrame.add(Main.myNetworkPanel);  Main.myFrame.repaint();  }  });  *// thêm hành động cho EXIT button*  exitButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent e) {  System.exit(0);  }  });  add(oneButton);  add(twoButton);  add(LANButton);  add(exitButton);  **this**.add(background);  } |

## Xây dựng giao diện chơi game:

Ta xây dựng class GamePanel.java có chức năng tương tác với người chơi ở chế “1 player” và chế độ “2 player” .

Ta tạo một class kế thừa từ JPanel của thư viện swing. Bao gồm:

* Hình nền: **backgroundPanel** là một đối tượng kiểu ImagePanel (đối tượng người lập trình tự định nghĩa)
* Bàn cờ : **tablePanel** là một đối tượng kiểu ImagePanel, có chức năng chứa các ô cờ.
* Ô cờ: **mySquare** là một mảng hai chiều chứa các đối tượng ImagePanel, được thiết lập action thay đổi icon là ‘X’ hoặc ‘O’.
* Nút thoát game: **myBackButton** có chức năng thoát khỏi màn hình game để trở về màn hình chính. 

tablePanel

myBackButton

**mySquare**

backgroundPanel

Để xác định chế độ chơi game là “1 player” hay “2 player” ta dùng thêm một biến int **numberPlayer**.

* Nếu numberPlayer =1 : chế độ 1 người chơi. Tiến hạnh khởi tạo thêm một đối tượng kiểu Computer (kiểu do người lập trình định nghĩa) có chức năng sử dụng thuật toán để tính toán nước đi cần đi cho máy.
* Nếu numberPlayer = 2: chế độ 2 người chơi. Ta tiến thành thay đổi luân phiên người chơi hiện hành. Khởi tạo biến int player:
  + Nếu player = 1: Lượt người chơi thứ nhất. Sử dụng icon là ‘X’.

Nếu player = 2: Lượt người chơi thứ nhất. Sử dụng icon là ‘O’.

* + Lượt chơi luân phiên đổi cho đến khi kết thúc game.

|  |
| --- |
| GamePanel.java |
| package caro;  import java.awt.event.MouseAdapter;  import java.awt.event.MouseEvent;  import javax.swing.JPanel;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** GamePanel extends JPanel {  **public** **static** **int** winner; *// player 1 win -> 1 ; player 2 win -> 2 ; computer win->3 ;draw ->0*  **public** **int** height = 16 ; *// chiều cao bàn cờ , mặc định la 16 ô*  **public** **int** width = 16 ; *// chiều rộng bàn cờ , mặc định la 16 ô*  **public** **int** numberPlayer;  **public** **int** player; *// lượt người chơi hiện hành : 1-> player 1 ; 2 -> computer|player 2*  **public** StatusBoard myStatus; *// bảng trạng thái gồm các giá trị : 0 chưa đánh , 1|2 đã đánh*  **public** **int** address; *// địa chỉ ô được click chuột*    **public** SoundPlayer mySound = **new** SoundPlayer(); *// đối tượng chơi âm thanh*  **public** **static** boolean canPlaySound = true; *// biến điều kiện xem có thể chơi âm thanh*      **public** **static** ImagePanel backgroundPanel; *// JPanel chính*  **public** ImagePanel tablePanel; *// JPanel chứa các ô vuông*  **public** BackButton myBackButton; *// JButton quay về màn hình chính*    **public** Computer myComputer; *// đối tượng computer có khả năng tính toán nước đi*  **public** Check myCheck; *// đối tượng kiểm tra xem có ai thắng hay hòa?*      **public** MouseAdapter myAction; *// đối tượng tạo hành động cho các ô*      **public** GamePanel() {  */\* khởi tạo các giá trị ban đầu cần thiết \*/*  winner = -1 ;  setBounds(0, 0, 800, 600);  setLayout(null);  height = 16;  width = 16;  myCheck = **new** Check(height, width);  myStatus = **new** StatusBoard(height, width);  myComputer = **new** Computer(height, width);  *// add button back to menu*  myBackButton = **new** BackButton(**"GamePanel"**);  add(myBackButton);  *// mainPanel*  player = 1; *// nguoi thu 1 đi dau tien*  backgroundPanel = **new** ImagePanel(**"picture/main.png"**, 0, 0, 800, 600);  *// panel chứa bàn cờ*  tablePanel = **new** ImagePanel(**"picture/table.png"**, 20, 20, 480, 480);  *// tạo các ô cờ*  ImagePanel[][] mySquare = **new** ImagePanel[16][16];  *// action of normal game*  normalGame();  **for** (**int** i = 0; i < 16; i++) {  **for** (**int** j = 0; j < 16; j++) {  myStatus.statusBoard[i][j] = 0;  mySquare[i][j] = **new** ImagePanel(**"picture/khung.png"**, i \* 30, j \* 30, 30, 30);  tablePanel.add(mySquare[i][j]);    mySquare[i][j].addMouseListener(myAction);  }  }  repaint();  *//*  add(tablePanel);  add(backgroundPanel);  }  **public** **void** normalGame() {  player = 1;  myAction = **new** MouseAdapter() {  @Override  **public** **void** mouseClicked(MouseEvent e) {  **if** (canPlaySound) {  mySound.playSound(**"sound/kick.mp3"**);  }  **if** (Main.startGame == true) {  Main.myFrame.repaint();  ImagePanel a = (ImagePanel) e.getComponent();  address = tablePanel.getComponentZOrder(a); *// lấy địa chỉ ô ấn vào từ 0 đến (max -1) theo cột dọc*  **int** row = address % 16;  **int** col = address / 16;  **if** (myStatus.statusBoard[row][col] == 0) {  **if** (player == 1) {  a.setPicture(**"picture/khung1.png"**);  myStatus.setStatus(row, col, player);  repaint();  System.out.println(**"index "** + row + **" "** + col + **" "** + myStatus.statusBoard[row][col]);  *// kiem tra*    **if** (myCheck.checkIt(row, col, myStatus.statusBoard, player) == true) {  Main.startGame = false;  winner = 1;  winnerFrame myWinnerFrame = **new** winnerFrame(1);  System.out.println(**" Player 1 win !"**);  } **else** **if** (myCheck.isDraw(myStatus.statusBoard)) {  Main.startGame = false;  winner = 0;  winnerFrame myWinnerFrame = **new** winnerFrame(0);  System.out.println(**"Draw"**);  }  player = 2;  **if** (numberPlayer == 1) { *//chế độ mội người chơi*  *// Computer tính toán và đi*  myComputer.calculateEvalBoard(player, myStatus.statusBoard);  **do** {  Computer myComputer = **new** Computer(height, width);  myComputer.calculateEvalBoard(player, myStatus.statusBoard);  myComputer.FindMove(myStatus.statusBoard);  row = myComputer.optimalX;  col = myComputer.optimalY;  } **while** (myStatus.statusBoard[row][col] != 0);  ImagePanel b = **new** ImagePanel(**"picture/khung2.png"**, col \* 30, row \* 30, 30, 30);  tablePanel.add(b);  repaint();  myStatus.statusBoard[row][col] = 2;  **if** (myCheck.checkIt(row, col, myStatus.statusBoard, player) == true) {  Main.startGame = false;  winner = 3;  winnerFrame myWinnerFrame = **new** winnerFrame(3);  System.out.println(**" Computer win !"**);  } **else** **if** (myCheck.isDraw(myStatus.statusBoard)) {  Main.startGame = false;  winner = 0;  winnerFrame myWinnerFrame = **new** winnerFrame(0);  System.out.println(**"Draw"**);  }  player = 1;  }  } **else** **if** (player == 2) {  a.setPicture(**"picture/khung2.png"**);  myStatus.setStatus(row, col, player);  repaint();  System.out.println(**"index "** + row + **" "** + col + **" "** + myStatus.statusBoard[row][col]);  *// kiem tra nước đi chiến thắng?*    **if** (myCheck.checkIt(row, col, myStatus.statusBoard, player) == true) {  Main.startGame = false;  winner = 2;  winnerFrame myWinnerFrame = **new** winnerFrame(2);  System.out.println(**" Player 2 win !"**);  } **else** **if** (myCheck.isDraw(myStatus.statusBoard)) {  Main.startGame = false;  winner = 0;  winnerFrame myWinnerFrame = **new** winnerFrame(0);  System.out.println(**"Draw"**);  }  player = 1;  }  }  }  }  };  }  } |

## Xây dựng giao diện kết nối mạng LAN:

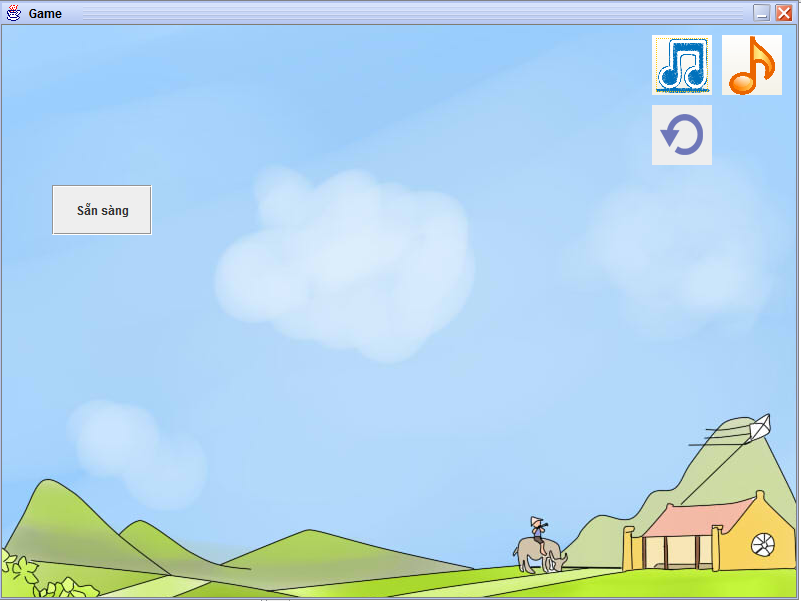
Xây dựng đối tượng **NetworkPanel.java** kế thừa JPanel của thư viện swing với chức năng kết nối các người chơi trong mạng LAN thông qua cách kết nối **socket** với nhau.

Trên JPanel này ta tạo một joinButton kiểu Jbutton của thư viện swing có chức năng khởi tạo một đối tượng kiểu Client (đối tượng do người lập trình định nghĩa).

Để kết nối mạng LAN, ta cần :

* Server kết nối : tạo đối tượng **myServer** kiểu Server(kiểu người lập trình định nghĩa).
* Client kết nối: tạo đối tượng **myClient** kiểu Client (đối tượng do người lập trình định nghĩa).

Trong quá trình kết nối mạng LAN, khi người chơi đầu tiên kết nối thì tiếp theo là quá trình đợi người chơi thứ hai kết nối. Trong quá trình chời đợi này, ta không thể thực hiện các thao tác như thoát, tắt mở nhạc. Do đó quá trình kết nối mạng LAN cần phải diễn ra song song với các quá trình khác nên ta cần tạo một Thread để thực hiện quá trình này.

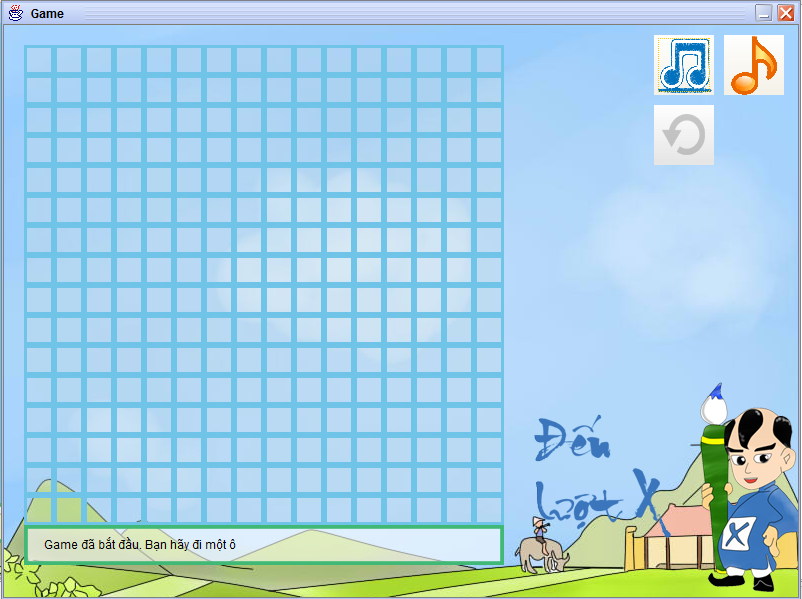


|  |
| --- |
| NetworkPanel.java |
| package caro;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.net.InetAddress;  import javax.swing.JButton;  import javax.swing.JOptionPane;  import javax.swing.JPanel;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** NetworkPanel extends JPanel {  **public** **static** Client myClient;  **public** BackButton myBackButton;  **public** String host;  **public** **int** port;  **public** **static** Server myServer;  **public** **static** JButton joinButton;  **public** ImagePanel waitingPanel;  ImagePanel background = **new** ImagePanel(**"picture/main.png"**, 0, 0, 800, 600);  **public** SoundPlayer mySoundPlayer = **new** SoundPlayer();  **public** **void** addJoinButton() {  joinButton = **new** JButton(**"Sẵn sàng"**);  joinButton.setBounds(50, 150 + 10, 100, 50);  joinButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent ae) {  *// player sound*  **if** (GamePanel.canPlaySound) {  mySoundPlayer.playSound(**"sound/click.mp3"**);  }  *// tao thread đẻ quá trình kết nối song song với game*  **new** Thread(**new** Runnable() {  @Override  **public** **void** run() {  **while** (true) {  String serverAddress;  **try** {  serverAddress = InetAddress.getLocalHost().getHostAddress();  serverAddress = (String) JOptionPane.showInputDialog(null, **"Nhập địa chỉ IP"**, **"Thông tin"**,  JOptionPane.INFORMATION\_MESSAGE, null, null, serverAddress);  System.out.println(**"server address: "** + serverAddress);  myClient = **new** Client(serverAddress);  JOptionPane.showMessageDialog(null, **"Để kết nối, người chơi của bạn cần nhập IP : "** + serverAddress);  Main.myFrame.remove(Main.myNetworkPanel);  Main.twoLanPlayerPanel = myClient.background;  Main.myFrame.add(Main.twoLanPlayerPanel);  Main.myFrame.repaint();  myClient.play();  **if** (myClient.wantsToPlayAgain()) {  Main.myFrame.remove(Main.twoLanPlayerPanel);  } **else** {  Main.myFrame.remove(Main.twoLanPlayerPanel);  Main.myFrame.add(Main.myStartPanel);  **break**;  }  } **catch** (Exception ex) {  }  }  }  }).start();  joinButton.setEnabled(false);  }  });  add(joinButton);  }  **public** NetworkPanel() {  myServer = **new** Server();  **new** Thread(  **new** Runnable() {  @Override  **public** **void** run() {  setBounds(0, 0, 800, 600);  setLayout(null);  *// button to return start menu*  myBackButton = **new** BackButton(**"NetworkPanel"**);  add(myBackButton);  */\*----------button to join to host---------- \*/*  addJoinButton();  */\*----------- Picture of backround----------- \*/*  *// background = new ImagePanel("picture/main.png", 0, 0, 800, 600) ;*  add(background);  }  }  ).start();  }  } |

## Xây dựng giao điện chơi game qua mạng LAN:

Tương tự như giao diện người chơi thông thường, ta thiết kế một JPanel có các đối tượng:

* Hình nền: **backgroundPanel** là một đối tượng kiểu ImagePanel (đối tượng người lập trình tự định nghĩa)
* Bàn cờ : **tablePanel** là một đối tượng kiểu ImagePanel, có chức năng chứa các ô cờ.
* Nút thoát game: **myBackButton** có chức năng thoát khỏi màn hình game để trở về màn hình chính.
* Ô cờ: **mySquare** là một mảng hai chiều chứa các đối tượng ImagePanel, được thiết lập action thay đổi icon là ‘X’ hoặc ‘O’. Ngoài ra, do giao tiếp qua mạng LAN, nến các phần tử này có thêm action gửi tin nhắn chứa thông tin đến vị trí đến Server để xử lí.
* Bảng thông báo từ Server: **messageLabel** là đối tượng kiểu Jlabel của thư viện swing.



## Xây dựng đối tượng kết nối các cảnh game lại:

Xây dựng đối tượng **Main.java** có chức năng quyết định “cảnh game” hiện hành.

Ta tạo đối tượng **myFrame** kiểuJFrame của thư viện swing.

Ta có 3 cảnh game đó là : MenuPanel , GamePanel, NetworkPanel.

* Ứng với cảnh game x cần dùng: myFrame.add(x);
* Ứng với cảnh game x cần loại bỏ: myFrame.remove(x);

|  |
| --- |
| **Main.java** |
| package caro;  */\**  *@author VanNinh*  *Main.java : là đối tượng chạy chính , quyết định "cảnh game"* *hiện hành*    *\*/*  import javax.swing.JFrame;  import javax.swing.UIManager;  **public** **class** Main {  **public** **static** JFrame myFrame;  **public** **static** StartPanel myStartPanel = **new** StartPanel();  **public** **static** GamePanel myGamePanel = **new** GamePanel();  **public** **static** NetworkPanel myNetworkPanel = **new** NetworkPanel();  **public** **static** ImagePanel twoLanPlayerPanel = **new** ImagePanel(**"picture/main.png"**, 0, 0, 800, 400);  **public** **static** boolean startGame;  **public** Main() {  startGame = true; *// đánh dấu game đã bắt đầu*    *// Khởi tạo cửa sổ Game*    myFrame = **new** JFrame(**"Game"**);  myFrame.setResizable(false);  myFrame.setVisible(true);  myFrame.setLayout(null);  myFrame.setBounds(100, 100, 800, 600);  myFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  *// nhạc nền*  MusicPanel myMusicPanel = **new** MusicPanel();  myFrame.add(myMusicPanel);  *// âm thanh*  SoundPanel mySoundPanel = **new** SoundPanel();  *// thêm button nhạc nền và âm thanh vào Frame chính*  myFrame.add(mySoundPanel);  myFrame.repaint();  *// khởi chạy menu start game*  myFrame.add(myStartPanel);  }  @SuppressWarnings(**"empty-statement"**)  **public** **static** **void** main(String[] args) {  **try** { *// sử Jato libary có chức năng thay đổi giao diện game đẹp hơn*  UIManager.setLookAndFeel(**"com.jtattoo.plaf.smart.SmartLookAndFeel"**);  } **catch** (Exception e) {  };  Main myMain = **new** Main();  }  } |

# Âm thanh:

## Xây dựng đối tượng phát âm thanh:

Xây dụng class **SoundPlayer.java** có chức năng phát âm thanh (không có chức năng lặp lại) từ một đường dẫn đến file nhạc cho trước, sử dụng cho các hiệu ứng âm thanh trong game.

Thêm thư viện phát nhạc là JPlayer. JPlayer là một thư viện phát nhạc định dạng mp3 mã người mở, để sử dụng tải về thêm vào trong thư viện.

Khi tiến hành đọc một file nhạc, chương trình sẽ đọc file cho đến khi hết file. Do đó, trong quá trình phát nhạc, ta không thể vừa chơi game được. Vậy, muốn vừa chơi game, vừa phát nhạc thì ta phải cho 2 tiến trình này song song nào bằng cách sử dụng Thread.

|  |
| --- |
| **SoundPlayer.java** |
| package caro;    import java.io.FileInputStream;  import javax.swing.JOptionPane;  import javazoom.jl.player.Player;    */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** SoundPlayer {  **private** FileInputStream FIS ;  **private** Player myPlayer ;      **public** **void** playSound(String path ) {    **try** {    FIS = **new** FileInputStream(path) ;  myPlayer = **new** Player(FIS) ;    **new** Thread (  **new** Runnable() {    @Override  **public** **void** run() {  **try** {  myPlayer.play();    }**catch**(Exception e) {  }  }  }  ).start(); ;    }  **catch**(Exception e ) {  JOptionPane.showInputDialog(e.getMessage()) ;  }    }    } |

## Xây dựng nút tắt/ mở âm thanh trong game:

Xây dựng một đối tượng có tên là SoundPanel.java đế thừa từ JPanel của swing.

Để có thể tắt hoặc mở âm thanh, ta tạo một đối tượng boolean **canPlaySound** :

* + Nếu canPlaySound = true : cho phép phát âm thanh.
  + Nếu canPlaySound = false : không cho phép phát âm thanh.

Ta xây dựng thêm đối tượng soundButton kiểu JButton của thư viện swing để thiết lập trạng thái true hoặc false cho đối tượng canPlaySound.

|  |
| --- |
| **SoundPanel.java** |
| package caro;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.Icon;  import javax.swing.ImageIcon;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JPanel;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** SoundPanel extends JPanel {  JButton soundButton;  **public** SoundPanel() {  setLayout(null);  setBounds(720, 10, 60, 60);  Icon bug1 = **new** ImageIcon(getClass().getResource(**"picture/OnSound.png"**));  soundButton = **new** JButton(bug1);  soundButton.setBorderPainted(false);  soundButton.setBounds(0, 0, 60, 60);  soundButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent ae) {  **if** (GamePanel.canPlaySound) {  GamePanel.canPlaySound = false;  Icon bug1 = **new** ImageIcon(getClass().getResource(**"picture/OffSound.png"**));  soundButton.setIcon(bug1);  } **else** {  GamePanel.canPlaySound = true;  Icon bug1 = **new** ImageIcon(getClass().getResource(**"picture/OnSound.png"**));  soundButton.setIcon(bug1);  }  }  });  add(soundButton);  }  } |

## Xây dựng đối tượng chơi nhạc nền:

Ta xây dựng đối tượng **BackgroundMusic.java** có khả năng phát ra một đoạn nhạc nền Nhạc nền sẽ được tự động lặp lại và có thể tắt hoặc mở tùy ý.

Tương tự như việc phát âm thanh, ta tạo một Thread để có thể vừa chơi game, vừa phát nhạc.

Ta cần thêm vào thư viện JPlayer để phát các đoạn nhạc mp3.

int **total** : Lưu trữ kích thước byte của file nhạc.

int **stop** :lưu trữ số byte còn lại chưa đọc của file.

void **playMusic**(final int pos ): phát đoạn nhạc từ vị trí byte thứ pos. Ta cần làm quá trình phát nhạc và quá trình chơi game diễn ra song song, do đó cần sử dụng Thread.

void **stopMusic**() : Dừng nhạc và tiến hành tính toán số Byte chưa đọc của file.

void **resumMusic**(): tính toán vị trí byte tiếp tục phát nhạc để gọi lại hàm chơi nhạc playMusic. Vị trí này sẽ bằng (total – stop).

total

stop

resume

**void** repeatMusic(): lập lại đoạn nhạc. Để làm, ta chỉ việc tiến hành đọc lại file nhạc.

|  |
| --- |
| **BackgroundMusic.java** |
| package caro;  import java.io.FileInputStream;  import javax.swing.JPanel;  import javazoom.jl.player.Player;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** BackgrondMusic extends JPanel {  **private** **int** total; *// tổng số byte của file nhạc*  **private** **int** stop; *// số byte chưa đọc*  **private** FileInputStream FIS;  **private** Player myPlayer;  **public** boolean firstPress = true;  **public** boolean isStopping = false;  **public** boolean isPlay = true;  **public** **void** stopMusic() {  **try** {  System.out.println(**"stop: "** + stop);  stop = FIS.available();  myPlayer.close();  } **catch** (Exception e) {  }  }  **public** **void** repeatMusic() {  **new** Thread(**new** Runnable() {  @Override  **public** **void** run() {  **while** (isStopping == false) {  **try** {  Thread.sleep(100);  **int** now = FIS.available();  **if** (now == 0) {  playMusic(-1);  }  } **catch** (Exception e) {  }  }  }  }).start();;  }  **public** **void** playMusic(final **int** pos) {  **new** Thread(**new** Runnable() {  @Override  **public** **void** run() {  **try** {  FIS = **new** FileInputStream(**"sound/bkmusic.mp3"**);  System.out.println(**"total "** + total);  total = FIS.available();  **if** (pos > -1) {  FIS.skip(pos);  }  myPlayer = **new** Player(FIS);  myPlayer.play();  } **catch** (Exception e) {  }  }  }  ).start();;  }  **public** **void** resumeMusic() {  **try** {  System.out.println(**"resume"** + (total - stop));  **if** (stop == 0) {  playMusic(-1);  } **else** {  playMusic(total - stop);  }  } **catch** (Exception e) {  }  }  **public** BackgrondMusic() {  playMusic(-1);  isStopping = false;  repeatMusic();  }  } |

## Xây dựng đối tượng tắt mở nhạc nền:

Ta xây dựng đối tượng **MusicPanel.java** kế thừa từ JPanel của thư viện swing**.**

Để phát nhạc nền, ta tạo đối tượng **myMusic** kiểu *BackgrondMusic* (kiểu do người lập trình định nghĩa).

* Tắt nhạc : myMusic.stopMusic() ;
* Phục hồi : myMusic.resumeMusic() ;

Để có thể tắt hoặc phục hồi nhạc nền dễ dàng, ta tạo đối tượng myButton kiểu JButton của thư viện swing.

|  |
| --- |
| MusicPanel.java |
| package caro;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.Icon;  import javax.swing.ImageIcon;  import javax.swing.JButton;  import javax.swing.JPanel;  */\*\**  *\**  *\* @author VanNinh*  *\*/*  **public** **class** MusicPanel extends JPanel {  **public** **static** BackgrondMusic myMusic;  JButton myButton;  **public** MusicPanel() {  myMusic = **new** BackgrondMusic();  setLayout(null);  setBounds(650, 10, 60, 60);  Icon bug1 = **new** ImageIcon(getClass().getResource(**"picture/OnMusic.png"**));  myButton = **new** JButton(bug1);  myButton.setBounds(0, 0, 60, 60);  myButton.setBorderPainted(false);  myButton.addActionListener(**new** ActionListener() {  @Override  **public** **void** actionPerformed(ActionEvent ae) {  **if** (myMusic.isPlay == false) {  Icon bug1 = **new** ImageIcon(getClass().getResource(**"picture/OnMusic.png"**));  myButton.setIcon(bug1);  System.out.println(**"stopping : "** + myMusic.isStopping);  myMusic.isPlay = true;  myMusic.isStopping = false;  **if** (myMusic.firstPress == true) {  myMusic.playMusic(-1);  myMusic.firstPress = false;  myMusic.repeatMusic();  } **else** {  myMusic.resumMusic();  myMusic.repeatMusic();  }  } **else** { *// ( my.isPlay = true )*  Icon bug1 = **new** ImageIcon(getClass().getResource(**"picture/OffMusic.png"**));  myButton.setIcon(bug1);  myMusic.isPlay = false;  myMusic.isStopping = true;  System.out.println(**"stopping : "** + myMusic.isStopping);  myMusic.stopMusic();  }  }  });  add(myButton);  }  } |

# Kết luận:

Ưu điểm :

* Đề tài đã xây dựng tương đối thành công về một game caro, bao gồm : hình ảnh trực quan, có âm thanh.
* Áp dụng tương đối tốt kiến thức xây dựng đồ họa bằng ngôn ngữ Java.
* Áp dụng được kiến thức lập trình mạng trong Java.

Hạn chế:

* Chưa thể kết nối qua internet mà chỉ kết nối qua mạng LAN.