TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN CƠ SỞ NGÀNH KĨ THUẬT PHẦN MỀM

ĐỀ TÀI: GAME SUDOKU

Sinh viên thực hiện:

Nguyễn Thị Thúy Nga

MSSV: B1605344

Cán bộ hướng dẫn:

ThS. Phan Huy Cường

Học kì 2, niên khóa 2018-2019

MỤC ĐÁNH GIÁ

Đánh giá của giảng viên hướng dẫn

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

I. TỔNG QUAN

1. Giới thiệu sơ lượt về Sudoku

Sudoku có lịch sử từ hàng ngàn năm. Năm 990, một danh sách những "Ô số kỳ ảo" đã xuất hiện và tỏ ra không khác mấy so với bản Sudoku xuất hiện trong từ điển bách khoa ikhwan al-salfa của các học giả người Ả Rập. "**Abraham Ben ibn Ezra**" một nhà triết học kiêm chiêm tinh học người **Hispanic** gốc Do thái bắc đầu quảng bá khối vuông "buduh" ở châu Âu. Ông đi khắp Tây Ban Nha, Ý và các nước khác ở châu Âu để giới thiệu với công chúng về " những ô số kỳ ảo".Ý tưởng tạo nên những ranh giới cho các khối vuông (biến nó thành trò chơi) đã được "**Ahmed al-Buni**" ghi lại vào năm 1255, mặc dù phương pháp này được tin là có xuất xứ từ Ba tư. Sudoku có thêm một bước tiến hóa mới vào năm 1776 khi một nhà toán học kiêm vật lý học người thụy Sỹ tên "**Leonhard Euler**" bắt đầu nghiên cứu và phát triển các luật chơi mà ngày nay ta gọi là luật chơi Sudoku.Sudoku lần đầu tiên được xuất bản vào cuối thập niên 1970 trong một tạp chí ở New York. Tờ tạp chí này đã được giới thiệu về các ô số kỳ ảo và khuôn nó lại trong một lưới 9x9, tạo thành từ các khối 3x3. Và như thế Sudoku đã ra đời.Năm 1986, trong một chuyến đi Mỹ, một nhà xuất bản Nhật Bản, "**Nikoli"** đã khám phá ra các ô số. Họ đặt tên cho nó là Sudoku(**Su** *là số*, **Doku** *là đơn độc*), và làm cho nó nhanh chóng trở thành một trò chơi phổ biến ở Nhật Bản.Năm 2004, niềm đam mê Sudoku đã đưa Wayne Gould đến với London nhân một chuyến đi thăm ngẫu nhiên báo The Times, Gould đã thuyết phục tổng biên tập của tờ báo này cho đăng Sudoku bên cạch các ô chữ độc giả tức bị cuốn hút. Chỉ trong vài tuần lễ, các tờ báo trên khắp nước anh đã thi nhau đăng Sudoku, từ đó Sudoku bắt đầu lan rộng sang Mỹ, Canada, Úc, Pháp, Nam phi và nhiều quốc gia khác.

2. Mô tả bài toán

Game Sudoku là game trí tuệ hấp dẫn người chơi mà luật chơi lại đơn giản đặc trưng là ở quốc gia Nhật Bản. Đây là game có một mạng lưới các ô bao gồm 81 ô, được xếp thành 9 hàng ngang và dọc và chia thành 9 block. Ban đầu khi chúng ta bấm vào ô bắt đầu thì sẽ có một số ô cho trước, các ô còn lại sẽ trống . Nhiệm vụ của người chơi phải lắp đầy các ô trống sao cho số của các ô trong hàng và cột và trong một block là các số từ 1 đến 9 không được trùng nhau.Trong mỗi game thường phải có ít nhất 4 nút cho người chơi tùy chọn và các nút từ 1-9 để người chơi có thể điền vào, em đã xây dựng game của mình với các nút tùy chọn là: **New, Check, Help on và Exit.**

- **New** là khi người chơi đã chiến thắng game và tiếp tục muốn chơi game hoặc khi người chơi đã chọn nhiều phương án sai và không muốn sữa lại thì bấm vào nút này sẽ cho một game mới. Khi người chơi bấm vào nút này , chương trình đồng thời tạo ra một file mới

tên là "MySudoku.txt". File này chứa đáp án cho trò chơi. Người chơi có thể tham khảo file này khi không tìm ra đáp án.

- **Check** là khi người chơi cảm thấy số điền vào có vẻ không phù hợp với quy luật, để có thể chắc thắng thì người chơi có thể bấm vào nútnày để kiểm tra nếu số điền vào là đúng thì nó sẽ hiển thị ra màu xanh lá còn sai thì số sẽ đổi thành màu đỏ. Nút Check cũng có chức năng là để xóa khi số hiển thị là màu đỏ thì người chơi có thể bấm vào số đó để xóa.
- **Help on** là nút giúp người chơi dễ dàng tìm ra phương án hơn bằng cách sẽ gợi ý cho người chơi các ô thích hợp cho con số đã chọn. Các ô thích hợp sẽ hiện thị màu xanh trên bàn cờ sudoku. Khi người chơi cho bắt đầu game em xây dựng nút Help sẽ được mặc định là đã được check.
- **Exit** là nút kết thúc trò chơi. Để dễ tương tác với người chơi khi người chơi vô tình bấm vào nút Exit thì sẽ có thông báo hỏi rằng người chơi có muốn thoát hay không.

Trước kia game Sudoku thường chơi trên giấy với bút chì, nhưng hiện nay đã có rất nhiều phần mềm chơi game trên máy tính và điện thoại.Khi so sánh game Sudoku với các loại game khác ta thấy, game này đã cũ nhưng vẫn rất được ưa chuộng bởi tính trí tuệ cao của nó. Không yêu cầu đồ họa cao, nhưng người chơi vẫn thích thú, mỗi bàn chơi là một thách thức tư duy logic. Kích thích tính hiếu thắng của người chơi, tính tò mò của các em nhỏ muốn chinh phục các con số.

Hi vọng sau khi em làm xong đề tài này em sẽ nâng cao được khả năng lập trình Java của mình và có thể áp dụng những kiến thức đã tìm hiểu cũng như học tập để có thể viết ra những ứng dụng hay những chương trình có ích cho bản thân em hay đời sống và xã hội.

3. Mục tiêu cần đạt được

- ✓ Hiểu được quy tắc xây dựng một game cơ bản.
- ✓ Áp dụng ngôn ngữ Java đã được học trên lớp để có thể xây dựng nên trò chơi Sudoku hoàn chỉnh.
- ✓ Xây dựng được giao diện thân thiện dễ nhìn và thích thú người chơi.
- ✓ Khắc phục những sự cố lỗi game ngoài ý muốn.
- ✓ Khi người chơi đã lấp đầy 81 ô và theo đúng quy luật của trò chơi thì thông báo là người chơi đã chiến thắng.
- ✓ Có được những tùy chọn cần thiết của 1 game Sudoku.

4. Hướng giải quyết và kế hoạch thực hiện

4.1. Hướng giải quyết

- Tìm hiểu những chức năng của Java(API) cần thiết để xây dựng giao diện cho game.
- Vì game đa số là thể hiện nút nên ta cần tìm hiểu kĩ về các JPanel cũng như các JButton của Java.
- Đọc và tham khảo tài liệu về game Sudoku.

4.2. Kế hoạch thực hiện

Tuần	Công việc thực hiện	Ghi chú
5	Tải game sudoku về chơi và tìm hiểu luật chơi.	
6	Đọc tài liệu tham khảo về game Sudoku.	
6	Thiết kế giao diện cho game.	
8	Tìm hiểu thuật toán cài đặc cho game.	
9	Tìm hiểu thuật toán cài đặc cho game.	
10	Bắt đầu code game.	
11	Viết chương trình.	
12	Viết chương trình.	
13	Viết chương trình.	
14	Viết chương trình.	

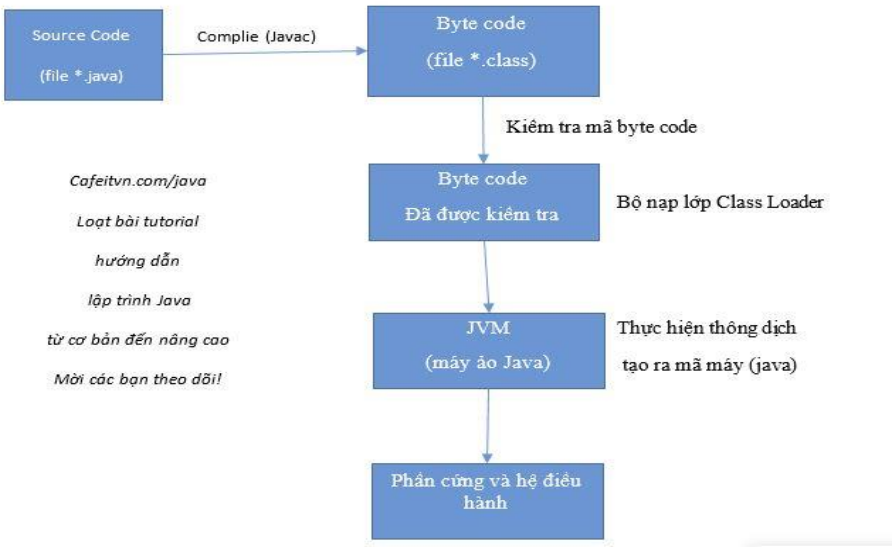
II. CƠ SỞ LÝ THUYẾT

1. Khái niệm lý thuyết sử dụng trong đề tài

Java là ngôn ngữ lập trình hướng đối tượng được phát triển bởi Sun microsystems và nó được phát hành vào năm 1995.James Gosling ban đầu đã phát triển Java trong Sun microsystems (sau này được sáp nhập với Tập đoàn Oracle).Java là một tập hợp các tính năng của C và C ++. Nó đã thu được định dạng từ các tính năng C và OOP từ C ++.Các chương trình Java độc lập với nền tảng, có nghĩa là chúng có thể chạy trên bất kỳ hệ điều hành nào với bất kỳ bộ xử lý nào miễn là trình thông dịch Java có sẵn trên hệ thống đó.Mã Java chạy trên một nền tảng không cần phải biên dịch lại để chạy trên nền tảng khác. Nó được gọi là viết một lần, chạy bất cứ nơi nào (WORA).Máy ảo Java (JVM) thực thi mã Java, nhưng nó đã được viết bằng các ngôn ngữ dành riêng cho nền tảng như C / C ++ / ASM, v.v. JVM không được viết bằng Java và do đó không thể độc lập với nền tảng và trình thông dịch Java là một phần của JVM.Ngôn ngữ Java cho phép xây dựng trình ứng dụng, trong đó nhiều quá trình có thể xảy ra đồng thời. Tính đa nhiệm cho phép các nhà lập trình có thể biên soạn phần mềm đáp ứng tốt hơn, tương tác tốt hơn và thực hiện theo thời gian thực.

- **Java platform** gồm có 3 thành phần chính:
    1. ***Java Virtual Machine*** (*Java VM*): Máy ảo Java.

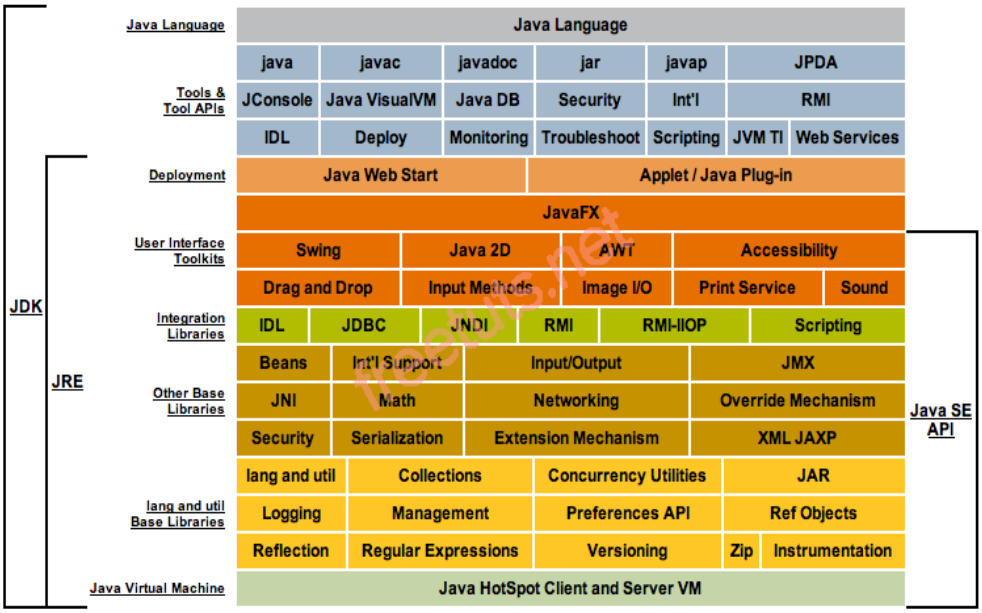
-Là 1 máy ảo trình thông dịch của Java. Nó cung cấp môi trường để code java có thể được thực thi, chương trình Java khi biên dịch sẽ tạo ra các file *.class chứa byte code , Các file *.class này sẽ được JVM thực hiện chuyển byte code thành mã máy tương ứng với từng hệ điều hành và phần cứng khác nhau thực thi. Ta có thể thấy rõ qua sơ đồ dưới đây:



Hình1. Sơ đồ máy ảo java.

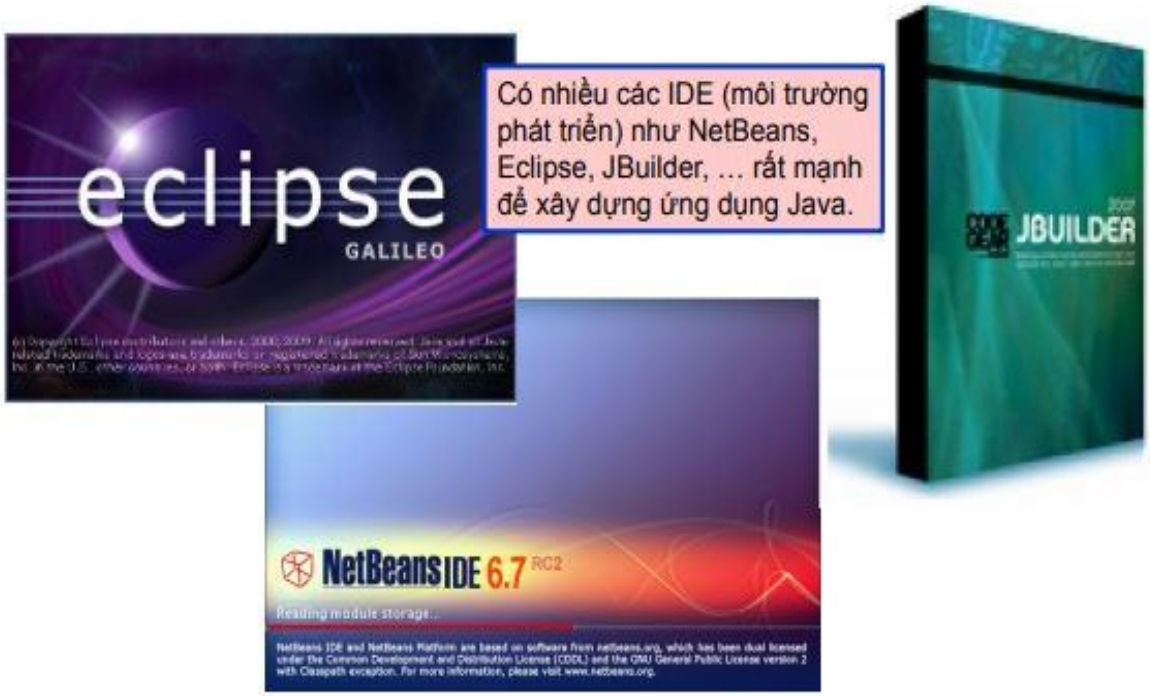
2. ***Java Application Programming Interface*** (*Java API*).
-Là một bộ sưu tập các gói, lớp và giao diện đã được viết sẵn với các phương thức, trường và nhà xây dựng tương ứng. Tương tự như giao diện người dùng, tạo thuận lợi cho sự tương tác giữa con người và máy tính, một API hoạt động như một giao diện chương trình phần mềm tạo điều kiện tương tác.
-API Java, bao gồm trong JDK, mô tả chức năng của mỗi thành phần của nó. Trong lập trình Java, nhiều thành phần này được tạo ra và thường được sử dụng. Do đó, lập trình viên có thể áp dụng mã đã viết trước thông qua API Java. Sau khi tham khảo các lớp và gói API có sẵn, lập trình viên dễ dàng gọi các lớp mã hóa và gói cần thiết để triển khai.
-API là một thư viện các lớp, gói và giao diện Java có sẵn. Ba loại API như sau:
    + Chính thức Java lõi API, được kèm với tải về JDK
    + API Java chính thức tùy chọn, có thể tải xuống nếu cần
    + Các API không chính thức, là các API của bên thứ ba có thể được tải xuống từ web nguồn.

3. ***Java Development Kit*** là một trong ba gói công nghệ cốt lõi được sử dụng trong lập trình Java, cùng với JVM (Máy ảo Java - Java Virtual Machine) và JRE (Java Runtime Environment - Môi trường Java Runtime). Việc phân biệt giữa ba công nghệ này, cũng như hiểu được cách chúng kết nối với nhau là rất quan trọng. (JDK) gồm trình biên dịch, thông dịch, trợ giúp, soạn tài liệu… và các thư viện chuẩn (Bộ công cụ phát triển Java SE) bao gồm một công cụ JRE (Môi trường chạy thi hành Java) hoàn chỉnh cộng với các công cụ để phát triển, gỡ lỗi và giám sát các ứng dụng Java. JDK được yêu cầu để xây dựng và chạy các ứng dụng và tiểu dụng Java.



Hình 2. Sơ đồ Java

4. ***Eclipse IDE*** là một môi trường phát triển tích hợp (IDE) cho Java và các ngôn ngữ lập trình khác như C, C ++, PHP, và Ruby … Môi trường phát triển được cung cấp bởi Eclipse bao gồm các công cụ phát triển Java Eclipse (JDT) cho Java, Eclipse CDT cho C/C ++, và Eclipse PDT cho PHP, và một số thứ khác. Hoặc ta có thể thay **Eclipse** bằng **NetBean** ngoài những ứng dụng cho Java như trên **NetBean** còn là công cụ lý tưởng cho việc phát triển phần mềm bằng các ngôn ngữ PHP, C/C++, Groovy vàGrails, Ruby and Rails, Ajax và JavaScript. Phiên bản 6.5 còn tăng cường hỗ trợ cho web framework (Hibernate, Spring, JSF, JPA), trình ứng dụng máy chủ GlassFish và cơ sở dữ liệu.



Hình 3. Công cụ soạn thảo

Thuật toán Backtracking(thuật toán quay lui): là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy. Người đầu tiên đề ra thuật ngữ này (backtrack) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.Dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng từng phần tử. Mỗi phần tử lại được chọn bằng cách thử tất cả các khả năng.

- Các bước trong việc liệt kê cấu hình dạng X[1...n]:
  - Xét tất cả các giá trị X[1] có thể nhận, thử X[1] nhận các giá trị đó. Với mỗi giá trị của X[1] ta sẽ:
  - Xét tất cả giá trị X[2] có thể nhận, lại thử X[2] cho các giá trị đó. Với mỗi giá trị X[2] lại xét khả năng giá trị của X[3]...tiếp tục như vậy cho tới bước:
  - Xét tất cả giá trị X[n] có thể nhận, thử cho X[n] nhận lần lượt giá trị đó.
  - Thông báo cấu hình tìm được.
  - Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu(Depth-Search).

2. Kết quả sử dụng lý thuyết trong đề tài

- ✓ Sử dụng ngôn ngữ Java đã được học trên lớp cũng như tìm hiểu thêm từ internet để áp dụng và thiết kế trong đề tài.
- ✓ Cài đặc và cấu hình Jdk để trình biên dịch có thể dịch ngôn ngữ trong Java.
- ✓ Sử dụng bộ công cụ NetBean để viết đề tài.
- ✓ Áp dụng quay lui vét cạn để giải bài toán sudoku.
- ✓ Ý tưởng: Mỗi bước tìm tập các giá trị khả dĩ để điền vào ô trống, và sau đó đệ quy để điền ô tiếp theo. Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy.

III. GIẢI QUYẾT VẤN ĐỀ

1. Thuật toán tạo bàn cờ

- Một sudoku 9x9 là được giải nếu ta kiểm tra các hàng, các cột, các block chứa 3x3 ô điều được điền đầy đủ các số từ 1 đến 9 và không trùng nhau.
- Đầu vào của thuật toán là mảng 9x9 chứa game Sudoku.
- Đầu ra là đúng nếu mảng đó tuân theo luật của Sudoku, sai nếu không.
- Ý tưởng:
    - + Đầu tiên ta sẽ tạo một mãng số nguyên từ 1-9 và dùng hàm trộn các số nguyên đó lại với nhau.
    - + Sau đó ta lấy một số từ 9 số đã trộn điền vào game và kiểm tra hàng, cột và block tại vị trí đó xem có thỏa mãn không nếu không thì ta sẽ gán tại vị trí đó bằng 0 và chạy lại với một số ngẫu nhiên khác cho đến khi các ô đã được điền số thì ta return được mảng game đã được điền đủ các số.

2. Thuật toán tạo game Sudoku cho người chơi

- Một sudoku 9x9 là được giải nếu ta kiểm tra các hàng, các cột, các block chứa 3x3 ô điều được điền đầy các số từ 1 đến 9.
- Đầu vào là một mảng 9x9 đã được điền đầy đủ 81 ô trong bàn cờ.
- Đầu ra là một mảng 9x9 đã được ẩn các số để người chơi điền vào.
- Ý tưởng:
    - + Đầu tiên ta trộn các vị trí các số đã được điền vào trong mảng game 9x9.
    - + Sau đó ta lấy một vị trí trong 81 vị trí đã trộn ngẫu nhiên đó và xóa số đã được điền vào trước đó.

+ Ta sử dụng thuật toán vét cạn là lần lược thay các số từ 1-9 vào vào ô đã xóa xem nếu ô đó có thể điền được "hai" số khác nhau trong các số từ 1-9 thì ta trả giá trị đã xóa về cho vị trí đó. Nếu ô đã xóa chỉ có đúng "một" số được điền vào từ 1-9 thì ta sẽ ẩn ô đó và không trả giá trị về cho ô đã bị xóa giá trị.

3. Xây dựng cấu trúc dữ liệu

❖ taoPhuongAn

- Trước khi tạo ra 1 bàn cờ , ta phải tạo ra một giải pháp. Công việc này được thực hiện bởi phương thức bên dưới với từng bước như sau:
    1. Kiểm tra giải pháp đã tìm thấy:
        a. Tìm thấy công việc => dừng.
        b. Chưa tìm thấy => tiếp tục
    2. X là hoành độ hiện tại của ô số, là số dư của vị trí chia cho tổng số ô trong hàng.
    3. Y là tung độ hiện tại của ô số, là số dư của vị trí chia cho tổng số ô trong cột.
    4. Một ArrayList có các số từ 1 đến 9 và trộn lên ngẫu nhiên. Việc trộn các số là quan trọng vì sẽ cho ra các bàn cờ khác nhau.
    5. Các số đã được trộn trong ArrayList sẽ được xử lí theo các bước sau:
        a. Phương thức getSoThichHop(int[][], int, int, List<Integer>) trả về 1 số thích hợp. Nếu không tìm thấy (giá trị số trả về là -1) trả về null.
        b. Tìm thấy số để điền vào vị trí hiện tại.
        c. Gọi đệ quy phương thức này với vị trí tăng lên. Giá trị trả về sẽ được lưu vào một biến.
        d. Nếu giá trị của biến là not null, trả về mảng 2 chiều được điền đầy đủ các vị trí. Ngược lại điền 0 vào ô hiện tại.
    6. Null sẽ được trả về khi không tìm thấy phương án nào.

```java
private int[][] taoPhuongAn(int[][] game, int index) {
    if (index > 80) {
        return game;
    }

    int x = index % 9;
    int y = index / 9;

    List<Integer> dsso = new ArrayList<Integer>();
    for (int i = 1; i <= 9; i++) {
        dsso.add(i);
    }
    Collections.shuffle(dsso);

    while (dsso.size() > 0) {
        int so = getSoThichHop(game, x, y, dsso);
        if (so == -1) {
            return null;
        }

        game[y][x] = so;
        int[][] tmpGame = taoPhuongAn(game, index + 1);
        if (tmpGame != null) {
            return tmpGame;
        }
        game[y][x] = 0;
    }

    return null;
}
```

- Phương thức **getSoThichHop(int[][], int, int, List<Integer>):** Phương thức này được sử dụng để tìm ra số thích hợp. Số này lấy được từ danh sách và được kiểm tra theo hàng, theo cột, theo block để đảm bảo đúng quy luật trò chơi. Nếu tìm được, số này sẽ được trả về, nếu danh sách rỗng mà vẫn chưa tìm thấy thì trả về -1.

```java
private int getSoThichHop(int[][] game, int x, int y, List<Integer> dsSo) {
    while (dsSo.size() > 0) {
        int so = dsSo.remove(0);
        if (ktX(game, y, so) && ktY(game, x, so) && ktBlock(game, x, y, so)) {
            return so;
        }
    }
    return -1;
}
```

- Phương thức **ktX()**, **ktY()**, **ktBlock()** kiểm tra một số đầu vào có thích hợp với cột, hàng và block.

```
private boolean ktX(int[][] game, int y, int so) {
    for (int x = 0; x < 9; x++) {
        if (game[y][x] == so) {
            return false;
        }
    }
    return true;
}

private boolean ktY(int[][] game, int x, int so) {
    for (int y = 0; y < 9; y++) {
        if (game[y][x] == so) {
            return false;
        }
    }
    return true;
}

private boolean ktBlock(int[][] game, int x, int y, int so) {
    int x1 = x < 3 ? 0 : x < 6 ? 3 : 6;
    int y1 = y < 3 ? 0 : y < 6 ? 3 : 6;
    for (int yy = y1; yy < y1 + 3; yy++) {
        for (int xx = x1; xx < x1 + 3; xx++) {
            if (game[yy][xx] == so) {
                return false;
            }
        }
    }
    return true;
}
```

- Phương thức **ktX(int [] game, int y, int so)**: đi qua từng ô trong một hàng, kiểm tra xem giá trị tại ô ở hàng x cột y xem có bằng với số đầu vào không nếu bằng thì trả về False ngược lại trả về true.
  + Flase thì số không thể điền vào bất kì ô nào trong hàng.
  + True thì số có thể điền vào hàng.
- Phương thức **ktY(int [] game, int x, int so)**: đi qua từng ô trong một cột, kiểm tra xem giá trị tại ô ở hàng x cột y xem có bằng với số đầu vào không nếu bằng thì trả về False ngược lại trả về true.
  + Flase thì số không thể điền vào bất kì ô nào trong cột.
  + True thì số có thể điền vào cột.
- Phương thức **ktBlock(int [] game, int x,int y, int so)**
  + biến x1 và y1 là tìm block trong bàn cờ.
  + hai dòng for xx và yy là dùng để đi qua các ô trong block đã xác định.
  + kiểm tra số đầu vào bằng với giá trị tại ô đó thì trả về false ngược lại true.

❖ **taoGame**

- Tạo ra một bàn cờ cho người chơi là lấy ra một vị trí từ 81 vị trí được trộn ngẫu nhiên và chắc rằng nó vẫn hợp lệ. Hợp lệ là với một bàn cờ chỉ có một cách giải. Để tạo ra bàn cờ được thực hiện bởi 2 phương thức bên dưới với các bước như sau:
  1. Ta có một Arraylist đã được điền đủ 81 vị trí trong bàn cờ
  2. Trộn ngẫu nhiên danh sách đó lên.
  3. Sau đó danh sách sẽ được đưa vào phương thức taoGame()
- Với các vị trí sẵn có trong danh sách, ta thực hiện như sau:
  1. Lấy vị trí đầu tiên ra khỏi danh sách và lưu vào một biến.
  2. X và Y được tính toán từ vị trí vừa lấy ra.
  3. Giá trị tại vị trí x, y vừa tính được lưu vào biến tạm.
  4. Gán giá trị tại vị trí x, y vừa tìm được bằng 0.
  5. Các bước này được lặp lại. Tiếp tục lấy vị trí trong danh sách cho đến khi bàn cờ không hợp lệ, giá trị sẽ được trả về vị trí cũ. Ngược lại tiếp tục cho đến khi trả về 1 bàn cờ có thể chơi.

```
private int[][] taoGame(int[][] game, List<Integer> dsViTri) {
    while (dsViTri.size() > 0) {
        int vt = dsViTri.remove(0);
        int x = vt % 9;
        int y = vt / 9;
        int temp = game[y][x];
        game[y][x] = 0;
        if (!hopLe(game)) // không hợp lệ (có 2 giải pháp cho 1 ô số )
        {
            game[y][x] = temp; // ghi lại số vừa lấy vào vị trí cũ
        }
    }
    return game;
}
```

- Một bàn cờ hợp lệ gồm mỗi hàng, mỗi cột, mỗi block 9x9 chứa các giá trị từ 1 đến 9 và chỉ có 1 cách giải duy nhất cho mỗi bàn cờ. Để thực hiện được ta tiến hành điền các giá trị đầu tiên vào các ô còn trống. Sau đó tìm thêm giải pháp bằng cách điền các giá trị khác vào ô còn trống. Nếu như tìm được giải pháp thứ hai thì việc kiểm tra kết thúc và phương thức trả về false, ngược lại thì bàn cờ vừa tạo ra là hợp lí và phương thức trả về true.

1. Kiểm tra nếu tìm thấy giải pháp
   + Tìm thấy => tăng số phương án lên 1 và trả về true nếu bằng 1, ngược lại trả về false.
   + Chưa tìm thấy => tiếp tục thực hiện
2. Tính tọa độ x,y từ vị trí đã được truyền vào
3. Kiểm tra ô hiện tại nếu = 0
   + True:
     - Tạo danh sách chứa các số từ 1 đến 9
     - Trong khi danh sách chưa rỗng. Thực hiện:
     - Tìm số phù hợp. Nếu giá trị là -1 , dừng thực hiện và trả về true
     - Đặt số vừa tìm thấy vào ô hiện tại
     - Gọi đệ quy hàm kiểm tra sự hợp lệ của bàn cờ và trả về giá trị true khi chỉ có 1 giải pháp, false khi có nhiều hơn 1 giải pháp, dừng kiểm tra và trả về game như cũ
     - Trả về ô hiện tại giá trị là 0
   + False:
     - Gọi đệ quy hàm kiểm tra sự hợp lệ của bàn cờ và trả về kết quả:
       - True: tiếp tục
       - False: return false

```
private boolean hopLe(int[][] game, int index, int[] sophuongan) {
    if (index > 80) {
        return ++sophuongan[0] == 1;
    }
    int x = index % 9;
    int y = index / 9;
    if (game[y][x] == 0) { // tim o co gia tri bang 0
        List<Integer> dsso = new ArrayList<Integer>();
        for (int i = 1; i <= 9; i++) {
            dsso.add(i);
        }
        while (dsso.size() > 0) {
            int so = getSoThichHop(game, x, y, dsso);
            if (so == -1) {
                break;
            }
            game[y][x] = so;

            if (!hopLe(game, index + 1, sophuongan)) {
                game[y][x] = 0;
                return false;
            }
            game[y][x] = 0;
        }
    } else if (!hopLe(game, index + 1, sophuongan)) // khi index chay het ma dung se return true
    {
        return false;
    }
    return true;
}
```

❖ **kiemtraGame**

- Kiểm tra số đầu vào do người dùng nhập vào. So sánh với giá trị tại ô vừa nhập với giải pháp và lưu kết quả vào một array. Số nhập vào luôn được đặt là 0 (không số nào được chọn). Tất cả các class observers được thông báo nếu Model có sự thay đổi.

```
public void kiemtraGame() {
    sochon = 0;
    for (int y = 0; y < 9; y++) {
        for (int x = 0; x < 9; x++) {
            kiemtra[y][x] = game[y][x] == phuongan[y][x];
        }
    }
    setChanged();
    notifyObservers(Update.CHECK);
}
```

❖ **ghiFile(int[][] sudoku)**

```
public void ghiFile(int[][] sudoku) {
    try {
        File f = new File("MySudoku.txt");
        if (!f.exists()) {
            f.createNewFile();
        }
        else {
            f.delete();
            f.createNewFile();
        }
        FileWriter fw = new FileWriter(f, true);
        fw.write("---------- MY Sudoku ------------\n\n");
        for (int x = 0; x < 9; x++) {
            for (int y = 0; y < 9; y++) {
                String s = String.valueOf(sudoku[x][y]) + " ";
                fw.write(s);
            }
            fw.write("\n");
        }
        fw.close();
    } catch (Exception ex) {
        ex.printStackTrace();
    }

}
```

Sau khi chạy chương trình, nếu file đã tồn tại ta sẽ xóa file đó và tạo lại một file khác. Mục đích của việc xóa đi file cũ sẽ giúp người chơi dễ dàng nhìn thấy do file chỉ lưu đáp án cho bàn cờ hiện tại. Nếu file chưa tồn tại sẽ tạo ra một file mới. File được tạo sẽ như sau

```
---------- MY Sudoku ------------

3 7 6 1 9 8 2 5 4
2 5 4 6 3 7 8 9 1
1 9 8 5 4 2 6 7 3
6 8 9 3 2 4 7 1 5
5 2 1 9 7 6 4 3 8
7 4 3 8 1 5 9 6 2
4 6 7 2 5 1 3 8 9
9 1 2 7 8 3 5 4 6
8 3 5 4 6 9 1 2 7
```

❖ **MySudoku**

- Chứa phương thức main, phương thức này để hiển thị giao diện chính của phần mềm. Lớp này xây dựng giao diện người dùng bằng cách tạo ra 1 JFrame trong đó chứa lớp SudokuPannel, ButtonPannel, ButtonPanel_Option. Lớp này cũng tạo ra một Lớp Game và thêm 3 lớp trên để lắng nghe sự thay đổi.

❖ **SudokuPanel và SubSudokuPanel**

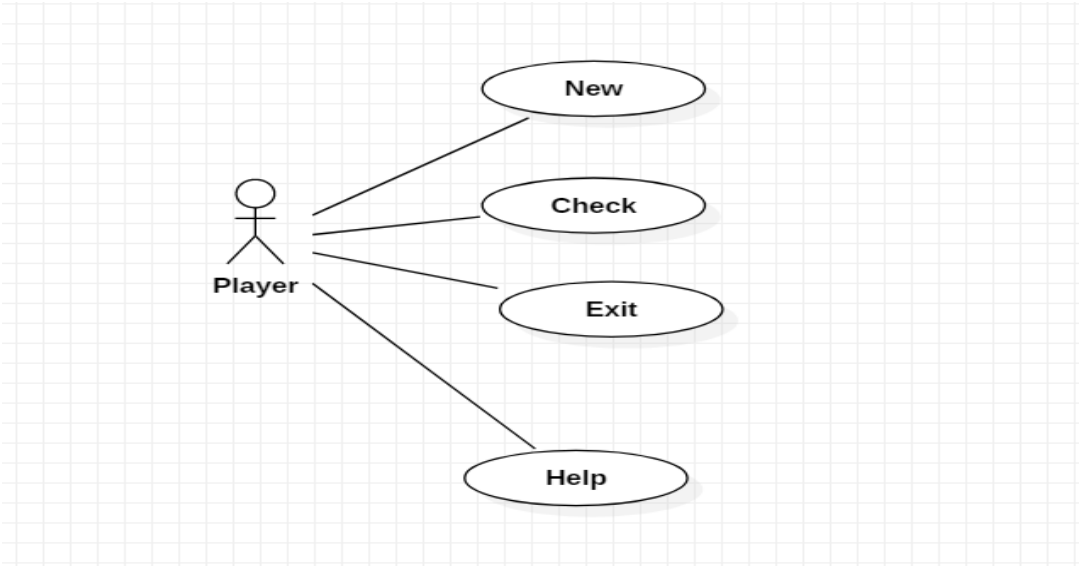
- Lớp SudokuPanel chứa 9 ô lớn, mỗi ô lại chứa 9 ô nhỏ Tất cả 9 ô lớn và 9 nhỏ đều là GridLayout 3x3. Mỗi 1 ô nhỏ này được thêm vào lớp SudokuController để quản lí đầu vào thông qua chuột.

❖ **ButtonPanel_Option chứa 3 Nút: New , Check , Exit.**

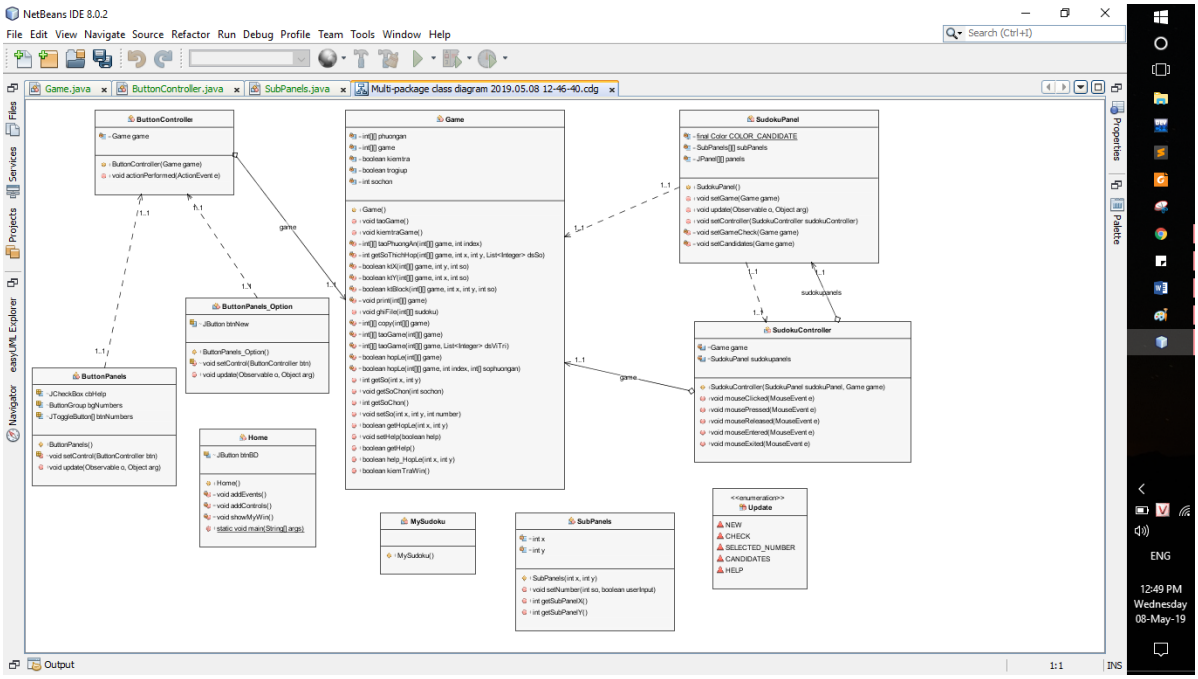
❖ **ButtonPanel** chứa radio button Help on và 9 button, mỗi button là 1 con số từ 1 đến 9 theo thứ tự tăng.

4. Sơ đồ chức năng

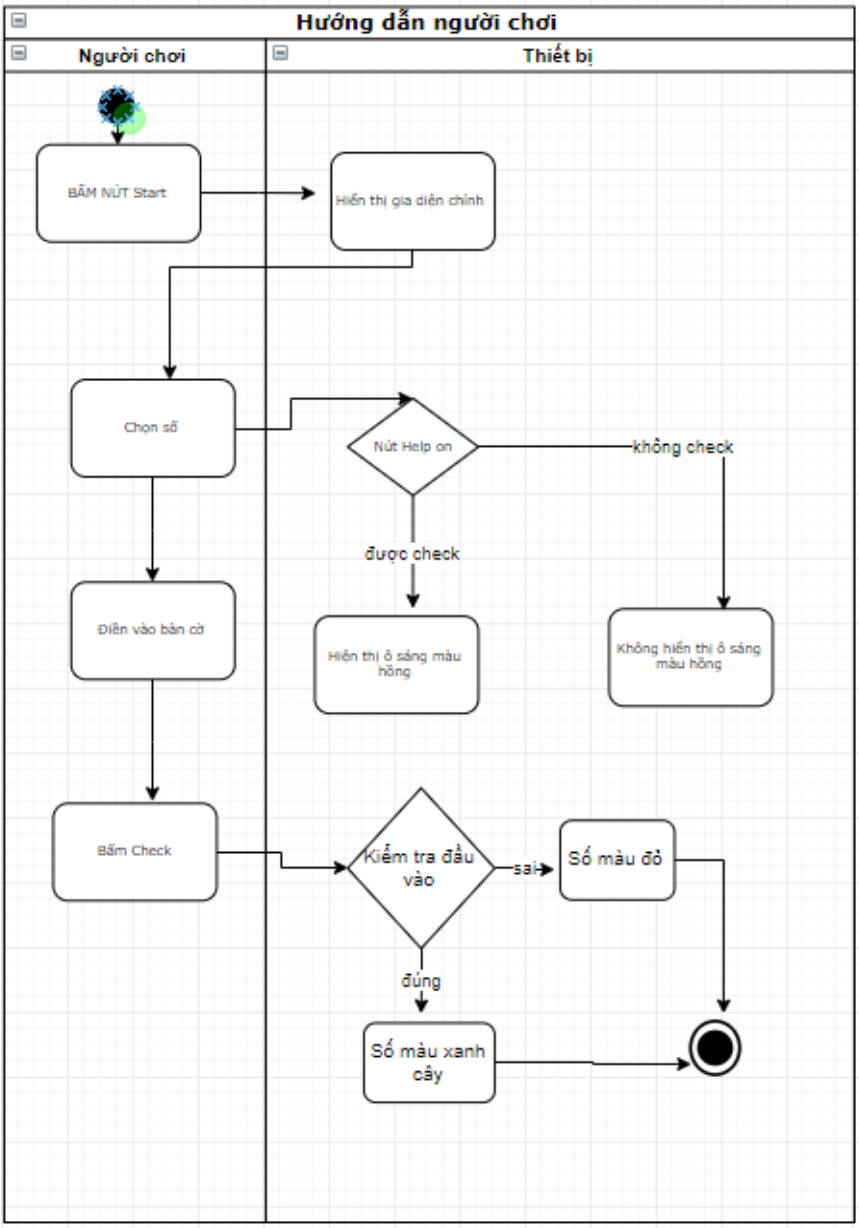
➢ Sơ đồ usecase



➢ Sơ đồ lớp



➢ Sơ đồ làn


Hướng dẫn người chơi
Người chơi
Thiết bị
BẤM NÚT Start
Hiển thị giao diện chính
Chọn số
Nút Help on
không check
được check
Điền vào bàn cờ
Hiện thị ô sáng màu hồng
Không hiển thị ô sáng màu hồng
Bấm Check
Kiểm tra đầu vào
sai
Số màu đỏ
đúng
Số màu xanh cây

IV. KẾT LUẬN VÀ ĐÁNH GIÁ

1. Kết quả đạt được

- ✓ Hiểu được cách thức xây dựng cũng như cấu trúc trong ngôn ngữ Java.
- ✓ Biết được thêm những hàm xây dựng trong Java và áp dụng để giải bài toán nhanh chóng hơn.
- ✓ Xây dựng được một chương trình game cơ bản, giao diện thân thiện với người dùng, bố cục mạch lạc và rõ ràng.
- ✓ Cải thiện khả năng suy nghĩ logic hơn nhờ vào chơi game.

2. Hạn chế và nguyên nhân

- Game được xây dựng ở mức cơ bản, không có tùy chọn mức độ.
- Không có thời gian theo dõi người khi hoàn thành.
- Giao diện chiến thắng không bắt mắt chỉ hiển thị dòng thông báo chiến thắng.
- Vì lần đầu tiếp xúc bằng ngôn ngữ mới để xây dựng một chương trình game hoàn chỉnh nên không tránh được những sai sót.

3. Hướng phát triển

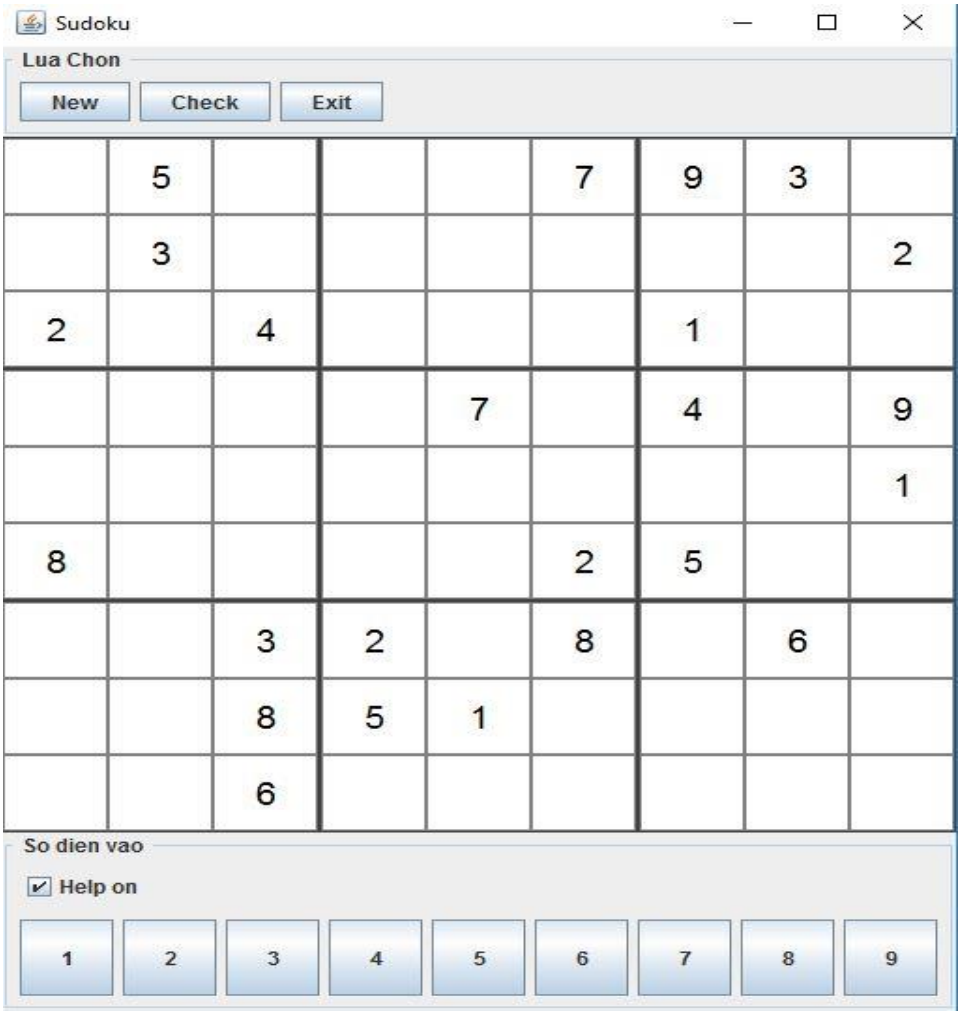
- ➢ Game cần có tùy chọn mức độ cho người chơi có thể nâng cao trình độ cho người chơi
- ➢ Cần phải thêm thời gian theo dõi người chơi nằm ở góc bên phải hiển thị thời gian hoàn thành game của người chơi.
- ➢ Giao diện khi chiến thắng là hình biểu tượng hoặc hình ảnh vô địch khiến cho người chơi có thích thú và chơi ở mức khó hơn.
- ➢ Qua những góp ý như trên thì game có thể đóng gói và đưa vào sử dụng rộng rãi.

V. PHỤ LỤC

1. Hướng dẫn sử dụng

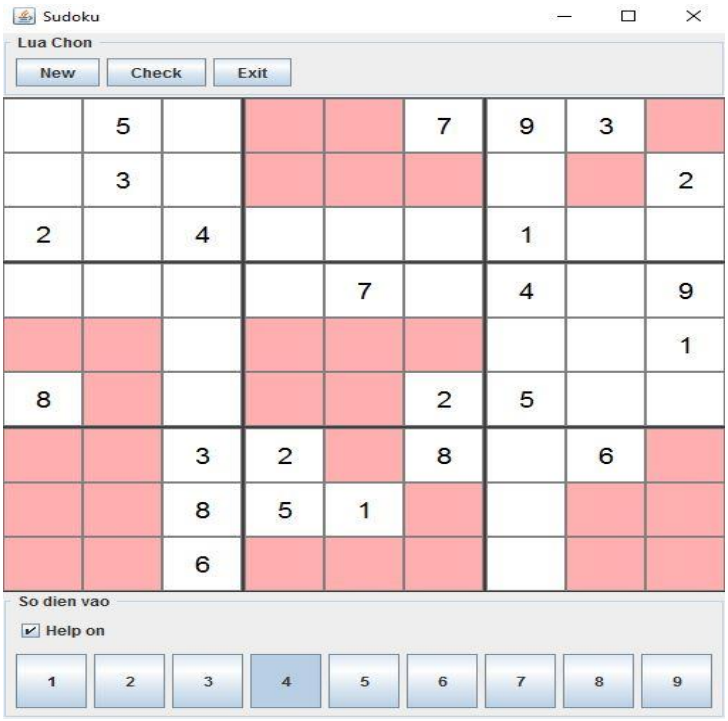
- Khi khởi động ứng dụng đầu tiên người chơi sẽ thấy hình như bên dưới và bấm nút START để bắt đầu trò chơi. Do hướng đến sự đơn giản nên giao diện chính chỉ có 1 nút Start.

- Giao diện game được hiển thị lên và bắt đầu chơi game. Người chơi sẽ thấy giao diện như bên dưới hình. Phần trên là các tùy chọn như New, Check, Exit. Phần giữa là bàn cờ để người chơi nhập số. Phần dưới là các số để người chơi nhập.

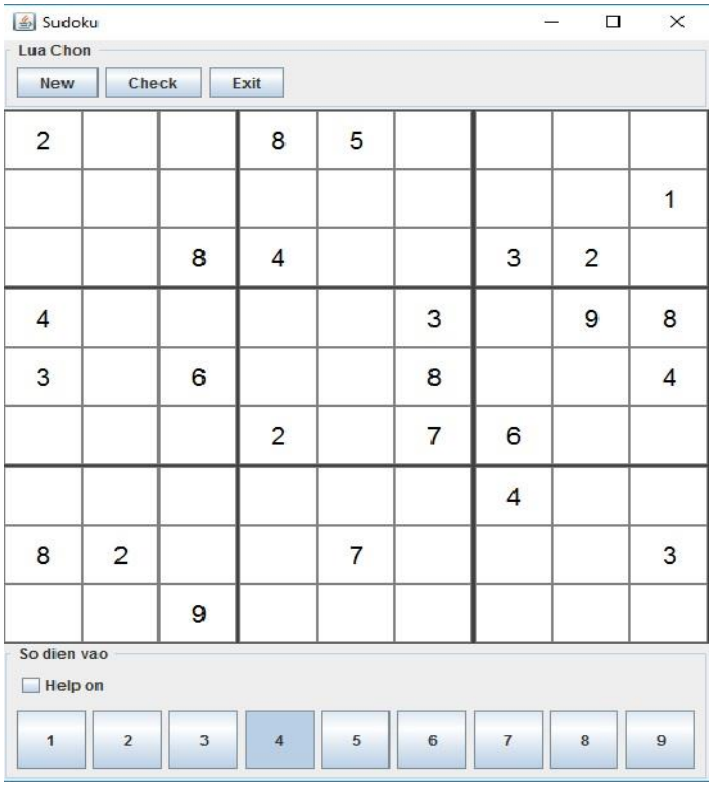


- Khi người chơi muốn điền số vào ô

+ Có trợ giúp

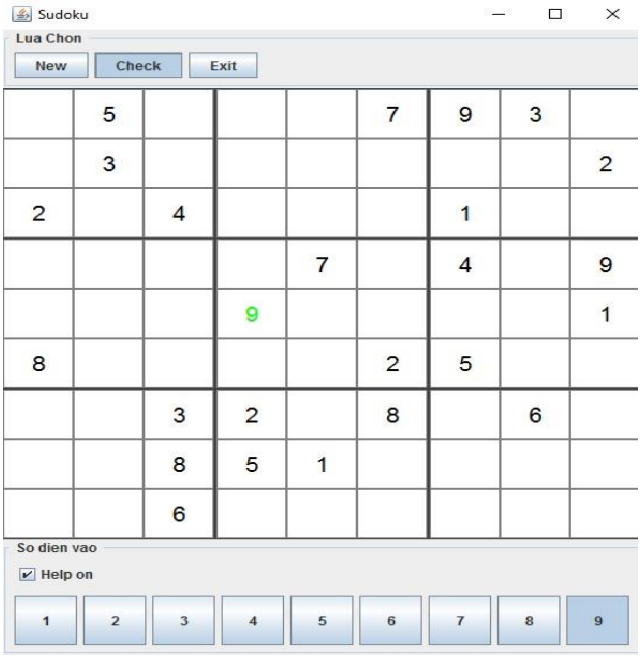


+ Không có trợ giúp

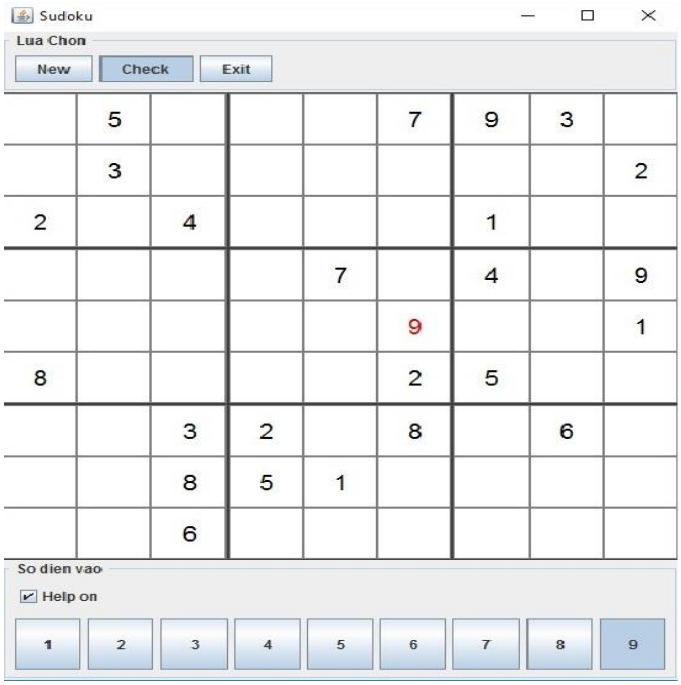


- Check số điền vào

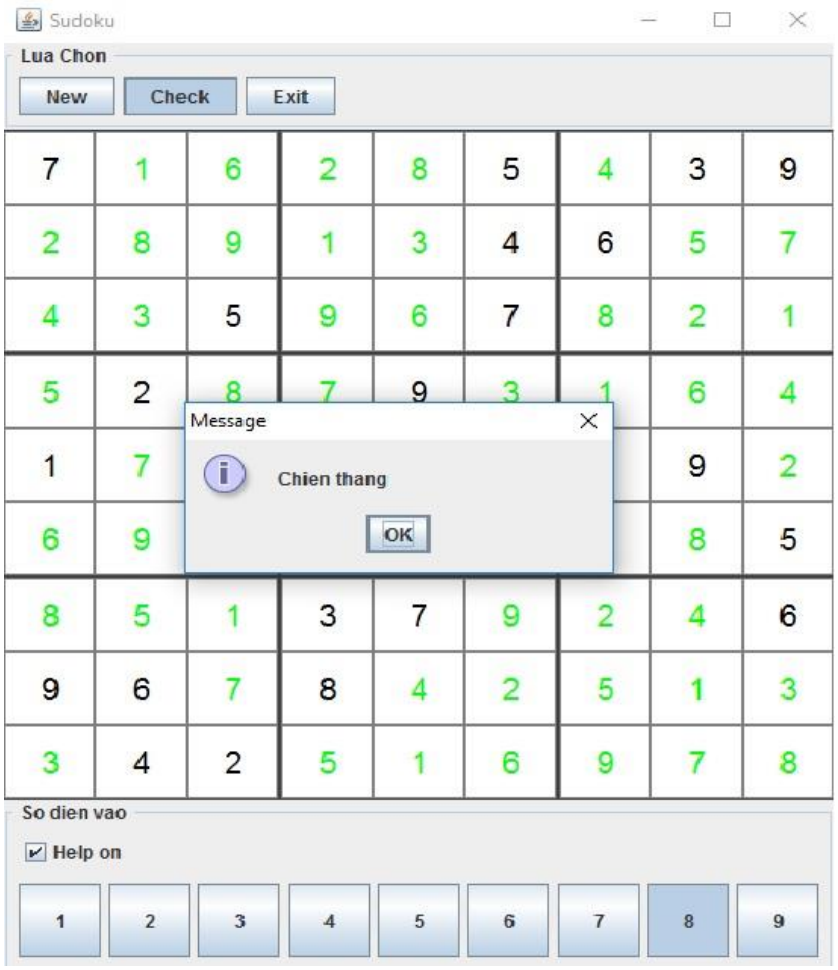
  + Đúng



+Sai

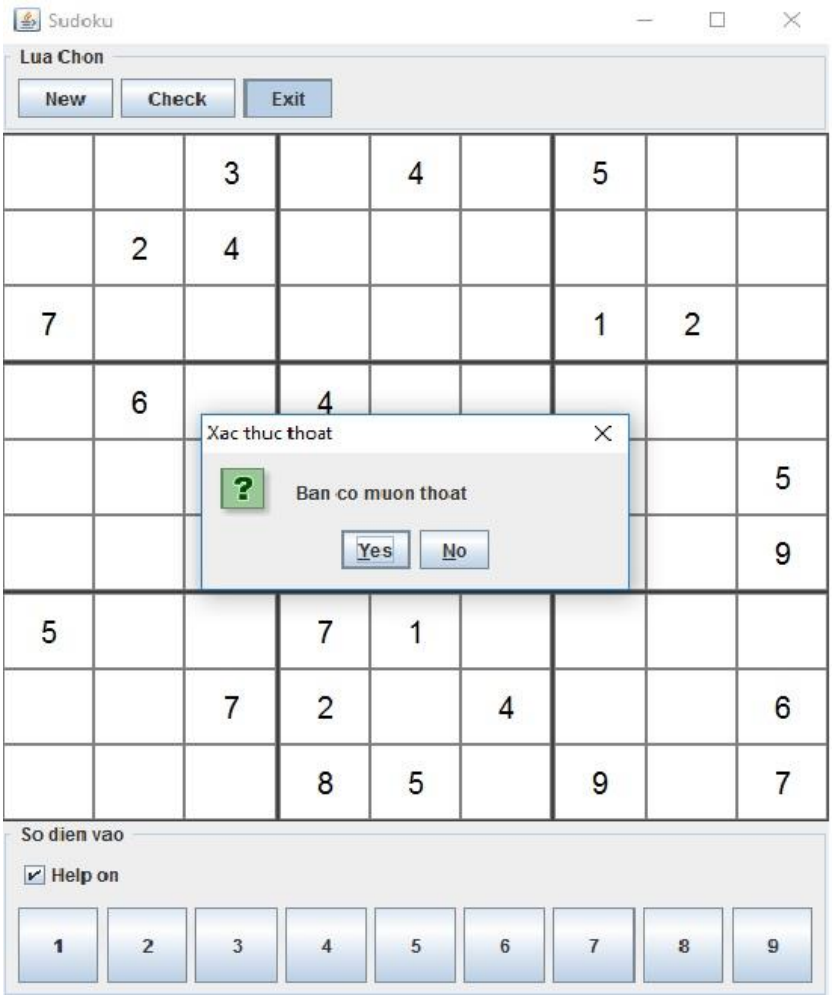


- Khi bấm vào check mà số được chọn sai ta có thể bấm vào số được chọn để hủy và điền số khác cứ như thế khi điền đủ hết ô ta sẽ có thông báo chiến thắng.



- Nếu người chơi muốn chơi tiếp thì bấm vào **Ok** và chọn **New** để chơi tiếp.

- Thoát game


Sudoku
Lua Chon
New
Check
Exit
Xac thuc thoat
Ban co muon thoat
Yes
No
So dien vao
Help on
1
2
3
4
5
6
7
8
9

2. Tài liệu tham khảo

Niên luận có sự tham khảo của các nguồn tài liệu.

[1] **Nguyễn Văn Linh**. Phân tích thiết kế thuật toán. Nhà xuất bản Đại học Cần Thơ, Cần Thơ, 2010.

[2] **TS.Trần Công Án, Ths Nguyễn Công Huy**. Giáo trình lập trình hướng đối tượng. Nhà xuất bản Đại học Cần Thơ, Cần Thơ, 2016.

[3]**Dasgupta and Sanjoy.** Algorithm. Boston: McGraw hill, 2008.

[4] **Trần Tiến Dũng**. Giáo trình lý thuyết và bài tập Java. Nhà xuất bản giáo dục, 1999.