VIETNAM NATIONAL UNIVERSITY – HCM

INTERNATIONAL UNIVERSITY

SEMESTER 2 (2023-2024)

DATA STRUCTURE AND ALGORITHM

SUDOKU GAME (Girls Edition)

Author:
Hoàng Ngọc Quỳnh Anh - ITCSIU22256

Instructor:
Dr. Vi Chi Thanh

Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject

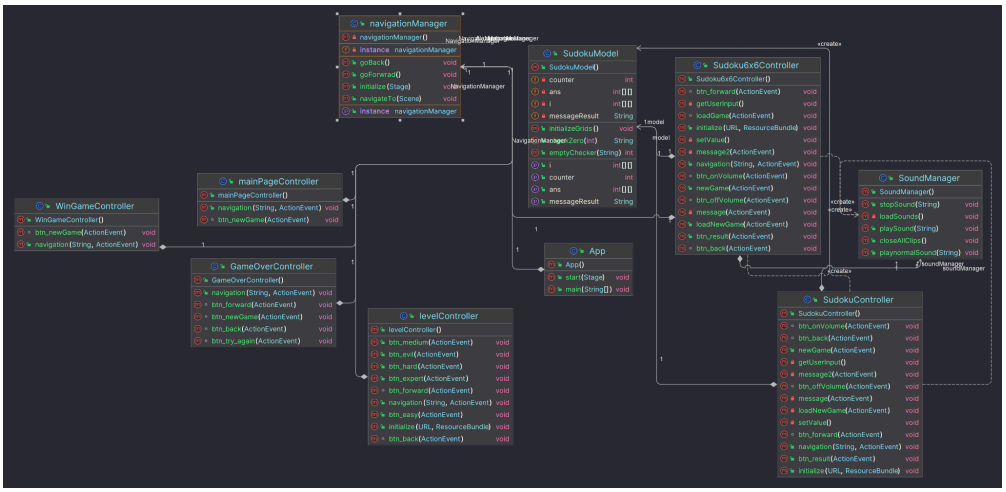
I/ INTRODUCTION

Our professor required us to develop a game using Data Structure and Algorithm in our Data Structure and Algorithm course, therefore I decided to design a sudoku game.So that we can manage and explain to my professor that the code I built uses some type of data structure, such as a 2D-Array dimension, HashMap, etc.

The only coding language I used for my project was Java, and I built the game using the Intellij IDE.

In this report, we will discuss this game in further detail, including how it may be constructed, what sort of game it is, and how people can join. We also discuss how we coded the game using data structures and algorithms.

II/ DESIGN CHART



This is the design chart of my project game

III/ DATA STRUCTURE AND ALGORITHM IN THE PROJECT

1. **Two Dimensional Array**

   The two Dimensional array is the core of this project since it is used for generating sudoku boards.

   This is the justification for creating the sudoku board. Initialize the array of j from 1 to 9 and p = 9, which is the array's element count, before generating. Based on j and p, the first for loop will provide a random number between **1 and 9** for the index between **[0][0] and [2][2]**. Following its generation, that number will be swapped out for the number at the end of the array, and to ensure that it doesn't repeat, p will be minimized by 1.

Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject

```
// part-1
for(int a=0; a<=2; a++){
    for(int b=0; b<=2; b++){
        int nn = random.nextInt(p1);
        i[a][b] = j1[nn];
        int sp = j1[nn];
        j1[nn] = j1[p1-1];
        j1[p1-1] = sp;
        p1--;

    }
}
```

In third loop and the second loop, the program will generate number sudoku for the index from **[0][3] to [2][5]** and from **[0][6] to [2][8]** by the logic of assigns the values from the previous row to the current row, and with the last row it assigns the values from the first row.

```
//  part-2
for(int a=0; a<=2; a++){
    int d = 0;
    for(int b=3; b<=5; b++){
        if(a==2){
            i[a][b] = i[0][d];
            d++;
        }else {
            i[a][b] = i[a + 1][d];
            d++;
        }
    }
}
```

In the fourth, fifth, sixth, seventh, eighth, ninth loop the program will generate the number sudoku for the 2D array at loop fourth ,fifth, seventh and eighth using the justification it assign the value of the array from from the middle three columns of the previous row to the current row, for the last row, where assigns the values from the middle three columns of the first row. However, in the loop sixth, ninth, it assigns the values from the last three columns of the previous row to the current row, with the exception of the last row, where it assigns the values from the last three columns of the first row.

```
//  part-4
for(int a=3, d = 1, s=0; a<=5; a++){
    int c = 3;
    for(int b=0; b<=2; b++){
        if(a==5){
            d = 0;
            i[c][s] = i[b][d];
            c++;
        }else {
            i[c][s] = i[b][d];
            c++;
        }
    }d++;
    s++;
}
```


Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject

Then, it will swap the first column with the second column, the third column with the fourth column and swap the sixth column with the seventh column. For the row it will also swap the first row with the second row, the third row with the fourth row and swap the sixth row with the seventh row.

Finally, the program will store the final grid into the **ans** array and based on the level users choose to play it will randomly set values for the grid of the array.

```
boolean port1 = false;
for(int a=0; a<9; a++){
    for(int b=0; b<9; b++){
        ans[a][b] = i[a][b];
```

After user finishes input all, the program will get user input and compare it with the answer grid that store in ans array, and if the answer is correct the program will pop up the win game window and if not it will appear the game over window, however, the user can click the button play again to be back with their old submission and they can changes it and submit again.

2. HashMap

This project uses a hashmap, one of the data structures, to construct a sound system. The SoundManager class contains the hashmap's logic.

The hashmap technique provides O(1) time complexity for sound search and lookup. Each sound clip in the game is linked to a particular event or level, and the hashmap stores these associations. The names of the events, levels and clips serve as the hashmap's key and value, respectively.

The rationale of the code is that the sound file path will be converted into clip objects by the **loadSounds()** function, and then the clip objects will be added to the hashMap along with the key event or level.

```
ClassLoader classLoader = this.getClass().getClassLoader();
File soundFile = new File(classLoader.getResource( name: "Sound/lofi-synth-pattern-29946.wav").getFile());
AudioInputStream audioInputStream1 = AudioSystem.getAudioInputStream(soundFile);
Clip clip1 = AudioSystem.getClip();
clip1.open(audioInputStream1);

soundMap.put("easy", clip1);
```

The project has created the **playSound()** function, which uses soundMap to obtain the Clip object from the soundMap in order to play ***sound.get(soundKey)***, where the sound key is the event or level, pauses the Clip and rewinds it to the beginning before

Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject

initiating playback if the Clip is underway. While the **stopSound()** function just stops the sound from playing, it follows the same logic as the **loadSounds()** method.

```
public void playSound(String soundKey) {
    Clip clip = soundMap.get(soundKey);
    if (clip != null) {
        clip.setFramePosition(0); // Rewind to the beginning
        clip.loop(Clip.LOOP_CONTINUOUSLY);
        clip.start();
    }
}
```

```
public void stopSound(String soundKey) {
    Clip clip = soundMap.get(soundKey);
    if (clip != null && clip.isRunning()) {
        clip.stop();
    }
}
```

3. Stack

In my project, the stack has implemented a backward and forward button. The reason for choosing the stack for implementing this kind of function in the game is because the stack is a data structure that is LIFO (last in, first out), so that is why it is so really suitable and good for tracking and implementing the backward and forward buttons.

```
private static final Stack<Scene> forwardStack = new Stack<>();
4 usages
private static final Stack<Scene> backStack = new Stack<>();
```

```
public static void goBack() {
    // when we go back the forward stack will push the current screen to the forward stack
    // and then it show the screen that currently in the backstack
    if (!backStack.isEmpty()) {
        forwardStack.push(primaryStage.getScene());
        primaryStage.setScene(backStack.pop());
    }
}
7 usages  quynhanhhoang572004
public static void goForwrad(){
    // when we go foward the back stack will push the current screen to the backstack
    // and then it show the screen that currently in the forwardstack
    if(!forwardStack.isEmpty()){
        backStack.push(primaryStage.getScene());
        primaryStage.setScene(forwardStack.pop());
    }
}
}
```

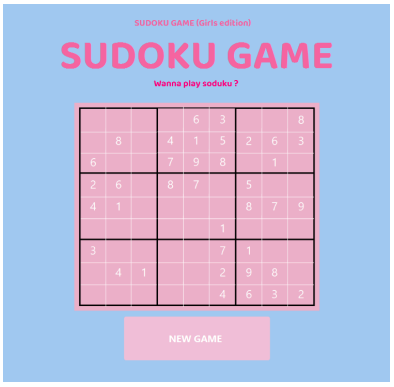
In the project, the class NavigationManager was made for handling backward and forward using the stack. I have created two stacks: the first one is ***Stack<Scene> forwardStack = new Stack<>()***, and the other one is ***Stack<Scene> backStack = new Stack<>()***, to handle back and forward events. The method **goForward()** is implemented for handling the go forward scene. When the user clicks on the event that has implemented the **goForward()** method and the forwardStack is not null, the backStack will push the current scene into the backStack and setScene, which is currently at the peak of the **forwardStack()**. Vice versa, if the **goBack()** method is similar to the **goForward()** method, when the user clicks on the event that has implemented the **goBack()** method and the forwardStack is not null, the backStack

Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject

will push the current scene into the backStack and setScene, which is currently at the peak of the **backStack()**.

IV/ RULE AND GAME PLAY

Initially, the homepage will appear when you click into the game.



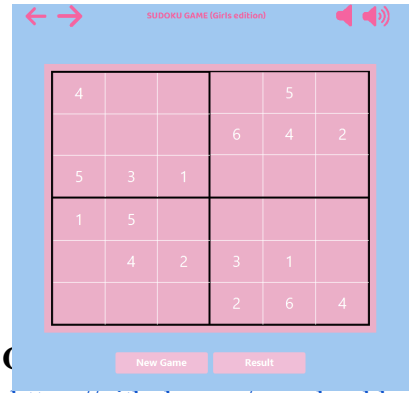
Next, the user will go through the level page when they click the NewGame button.

This game include 5 level which is " Easy", " Medium", " Hard", " Expert ", " Evil "

The 6x6 sudoku game's easy and medium levels require players to fill in each row and column from 1 to 6. This level is intended for novices and features a smaller grid that is simpler to understand and solve.

On the other hand, level hard, expert, evil is a 9x9 sudoku game designed for players who want to push their mental limits and find it somewhat more difficult to complete the puzzles. Players must fill each row and column from 1 to 9.

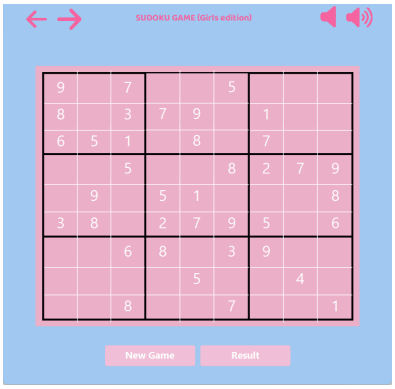
This is the the first-look of 6x6 and 9x9 sudoku level when you first click to them
6x6 sudoku:



C

https://github.com/quynhanhhoang572004/DSA_sudokuProject

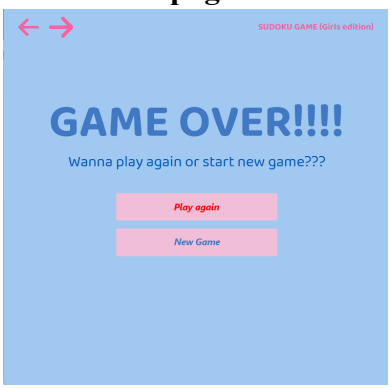
9x9 sudoku



Every level has a higher difficulty level, therefore the 6x6 sudoku level at easy is easier than the 6x6 sudoku level at medium. Similarly, the toughest level in 9x9 sudoku is "Evil." Simple rules apply: to play the game, players must fill in every row and column from 1 to 6 for easy and medium levels and from 1 to 9 for hard, expert, and evil levels. No number must be repeated inside a row, column, or square.

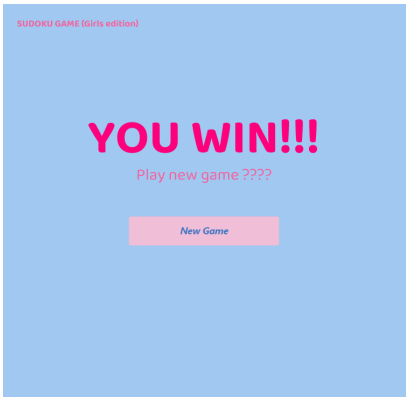
When users have completed filling in the whole blank grid, they may examine their results by clicking the result. The "play again" option enables users to continue the game in the event that the "Game over" screen occurs if their response is incorrect. By doing this, the user will be able to resume where they left off in the game and edit their response before sending it in again.The user will, however, see the "You win the game" page and be able to choose the "new game" option to start a new game if their response is right.

Game Over page:

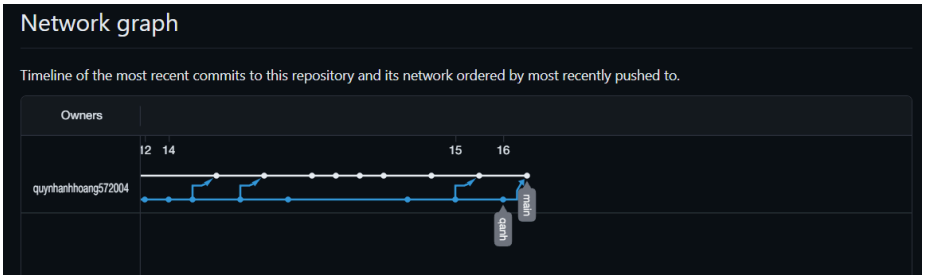


Win game page:

Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject



V/ GIT EXPLANATION



I use git to save my projects because it's quite helpful in allowing me to inspect the history of the code and, in the event that there are any errors, I can use git to check out the commit and see what the code was. Initially, the code was developed and tested on the "qanh" branch in the git repository. It was then run and tested to ensure functioning, and if it passed, it was merged into the main branch.

VI/ EXTRA FEATURE

1. **Apply design pattern**

In this project, it uses one design pattern, which is the **singleton design pattern**, to manage the change from one page to the next and also to manage the move backward and forward between pages. This design pattern applies to this class to ensure that there is only one instance created and the global point of access to it.

The reason why is to apply a singleton design pattern to this function, because with this singleton navigationManager, the state of the navigation is maintained in a single global instance. This ensures that the navigation state is consistent across the entire application. Since there is only one instance, it can effectively optimize the resources required for the whole application to make the application run faster and avoid run-out of memory problems.

```
public static navigationManager getInstance() {
    if (instance == null) {
        instance = new navigationManager();
    }
    return instance;
}
```


Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject

2. **Using JavaFx for GUI**

   A Java-based framework called JavaFx is used to create graphical user interfaces (UIs). It offers tools and UI controls for this purpose. Scene Builder is a drag-and-drop visual layout tool that produces GUI designs using FXML code. The sudoku (girls edition) application was made easier and looks pretty beautiful by integrating JavaFx with Scene Builder.

VII/ REFERENCES

(No date) Getting started with javafx. Available at: https://openjfx.io/openjfx-docs/ (Accessed: 18 June 2024).

Saahil-S-Mehta (no date) Saahil-S-Mehta/Sudoku_JavaFX: Sudoku game made using javafx, GitHub. Available at: https://github.com/Saahil-S-Mehta/Sudoku_JavaFX (Accessed: 18 June 2024).

Setup intellij idea for JavaFX & Scenebuilder and create your first javafx application (2023) YouTube. Available at: https://www.youtube.com/watch?v=IZCwawKILsk&t=22s (Accessed: 18 June 2024).

Singleton method design pattern (2024) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/singleton-design-pattern/ (Accessed: 18 June 2024).

The-Squad (no date) The-Squad/Sudoku-desktop-game: A Sudoku generating and solving algorithm built with javafx, GitHub. Available at: https://github.com/the-squad/sudoku-desktop-game (Accessed: 18 June 2024).

Github respiratory:
https://github.com/quynhanhhoang572004/DSA_sudokuProject