Presentation outline

- **Introduction**

1. Group number: group_0667.
2. Team member: Wei CUI, Xinyi Ji, Bohan Jiang, Zhuozi Zou, Kewei Qiu.
3. Each team member will say a sentence or two about which part of the game center code he or she worked on, matching with the meeting file.

- **Game Center**

1. Sudoku

   Demonstrate how to play sudoku such like you need to first click on the tile that you want to change then click on the button you want the tile to be, otherwise you will get a "has not click tile" warning. The user cannot change the tile that was given by the game itself, or you will get a "tile unchangeable" warning. The user cannot fill in number which is already an item of the tile's row, column or the 3×3 square, otherwise you will also get an "already has this number" warning.

   Show that the sudoku game can be unlimited undo.

   Load the already saved game and demonstrate that the sudoku game can be saved automatically.

   Show the scoreboard for current user and the scoreboard for the sudoku game itself.

2. Hanoi

   Choose a difficulty of this game and demonstrate how to play it (move disks on the screen).

Show the scoreboard for current user and the scoreboard for the Hanoi game itself.

The reason why our Hanoi cannot be saved is because the board manager for this game is draw class, and this class is actually a subclass of the View class which does not have a default constructor. Therefore, since the parent class does not have a default constructor, it cannot be serializable, and its subclass cannot be serializable neither. That is draw class cannot be saved like the board manager of the other two games. Hence the game Hanoi does not have a save function.

- **Code walkthrough**

Present our design and show the TA around our code.

1. The most important classes in our program:

   The User class, Score class which will save all the documents of this whole game.

   The UserManager class which will be the connection between the saved file and our game implementation.

   The SudokuBoardManager, SlidingTilesBoardManager, Draw classes which will implement all the logical function of these 3 games

   The HanoiGameActivity, SlidingTilesGameActivity, SudokuGameActivity which handles the collection between the controller and the model and the viewer of this game.

2. The design pattern we used and the problem they solved:

- Singleton Design Pattern, we use this design pattern in UserManager class, so there will always be only one unique instance in this class and that will be much easier for us to reach the signIn, signUp method during our signIn, signUp activity and write the connection between the saved file and our game implementation since we need to save to file every time there is a movement due to the implementation of autosave. Therefore the static field and static constructor in UserManager are actually a design pattern!!!
- Iterator Design Pattern, we use this design pattern in SlidingTilesBoard to solve the problem of puzzleSoved method in SlidingTilesBoardManager, so we can test whether the user has won this game in the order which we like it to be.
- Model_Viewer_Controller (MVC) Design Pattern, we actually use this design pattern during the whole implementation, mainly in the game activity classes, so the user will input some modification through Controller which will modify the model and update the whole view which will tell the user what to do next.
- Factory Design Pattern, we use this design pattern in BoardManager, StartingActivity and HelpActivity classes. The StartingActivity and HelpActivity classes interact with SlidingTilesBoardManager and SudokuBoardManager, and we obscure the creation process for these two board managers in these two classes. In StartingActivity and HelpActivity

classes we only know the game type which we are playing and we can implement the function using the subclass of BoardManager which is chosen depending on the game type. Without this design pattern, we will have to write two StartingActivity and two HelpActivity classes for sliding tiles and sudoku which is such a waste.

3. How we design our scoreboard, how do they displayed and where the high scores are stored.

   This program provides 2 kinds of score boards: one based on game and another based on user. Score board based on games will give the best 3 scores it recorded for each game. Username will be shown first, followed by one score, line by line. Score board based on users will only show the best 3 scores for the current user, also classified by games.

   For game Sliding Tiles, Sudoku and Hanoi, lower score means better performance since we get our score depending on the number of steps the user goes or the total time the user spent to win the game.

   The highest scores of these two types of scoreboards are stored as instance variables (each type is a 2-dimentional array of int) in a score instance which is an instance variable of a user instance which is an element of userList. (The userList itself is a instance variable (a map) of the unique UserManager instance.) And we will save this userList to the file save_file_temp.ser. (The way of storage of the instances of User is shown in the image bellow for better understanding)

- **Unit Tests**

  We will spend one minute running our unit tests and show the coverage to the teaching assistants.

  The reason why we don’t have the unit test for Hanoi is because the game Hanoi is implemented by drawing things on the screen so it is not testable. However, we can see whether our implementation is right or not by just watching the screen.

  Show our best unit test to the TA, SudokuPuzzleSolvedTest.

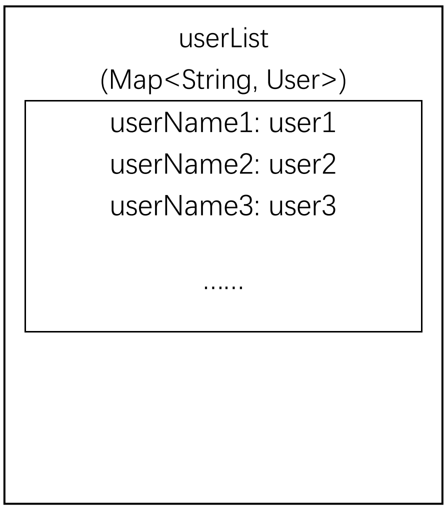
- **Appendix:**

  1.Since we change the names of some files of meeting document after the due date, we have to keep the copy of the old ones in afraid of changing the whole code. Therefore you should ignore these files: Firstmeeting.md, Secondmeeting.md, Thirdmeeting.md, Fourthmeeting.md, Fifthmeeting.md. And the real meeting documents are actually: meeting1.md, meeting2.md, meeting3.md, meeting4.md, meeting5.md, meeting6.md, meeting7.md, meeting8.md, meeting9.md. We are sorry for this happening.

  2.The image of the way of storage of the instances of User:

save_file_temp.ser

(file)


userList
(Map<String, User>)
userName1: user1
userName2: user2
userName3: user3
......