Presentation

Our group's unit test coverage is 19% lines covered. We have several classes with 100% code coverage(e.g. Card, GameManager, Generator and Tile) We also have some classes with 0% code coverage because they can't be tested. For example, the StartingActivity class, we can't test it because this class contains mostly the view methods, like button listener, maketoast text and load, save file, there is only few logic in it. Besides, the SwipeAdapter also can't be tested because adapters are links between other functional classes, but itself has nearly no logic.

Class GameView for game 2048 and class Generator for game minesweeper and class BoardManager for game slidingtiles play the most significant role in the entire program, since the main functionality of the program is those three games and they are construct by classes listed above. Basically, the class GameView provides main algorithms for running the game2048, such as where the cards goes when player swipe the screen with different directions, as well as whether the game ends or not. For game minesweeper, the Generator class generates random combination of grids and is able to check the neighborhood bomb number for each non-bomb cell. The main idea behind game minesweeper is that players assume the location of the bombs according to neighborhood bomb number displayed to them. As for class Boardmanager for game slidingtiles, methods in this class ensure the game will run correctly, such as which two tiles should be swapped when player click valid move and in what situation the click is invalid. Therefore, we conclude these three classes are most important in our program.

We use several design patterns.

1.The iterator design pattern in Board and UserManager class. We use the Board iterator to create tiles, the UserManager iterator to go through a list of users.

2. We used Singleton design pattern in our Manager class to ensure that there can only be one of that instance.

3. We used MVC design pattern in our scoreboard implementation. The controllers are ScoreboardController, ScoreboardControllerLocal class. The view is FragmentPage, FragmentPageLocal class. The model is User class.

Yuxin Yang part: The auto save and the preventing unsolvable sliding tiles and auto save. For the auto save we follow the instructions of the feedback, we give the player the right to choose which they want to play, either resuming the previous autosave game or start a new game. In the algorithm on how to make solvable sliding tiles, i divides it into 3 parts to show that why only this three circumstances can have solvable sliding tiles. I divide cases into odd and even part, I will show from the beginning to the last, in each step, the game will keep the circumstance of solvable sliding tiles.

Jinda Huang part:

The undo part. We have undo functionality for game2048 and sliding tiles. For sliding tiles, I create a list to store the positions of the blank tile, I add the position of blank tile to the list after each single tap. When I want to undo, I take out the last item from the list, which is

the position of the blank tile in the previous step, then swap it with the position of the current blank tile. To undo again, just keep taking out the last item from the list and swap it with the current blank tile. To restrict the undo steps, I restrict the size of the list.

For game2048, I store the grid (4 * 4 = 16 cards) of each state in a 2D int array(2 represent 2, 0 represent blank card), then store this array in an arraylist. Take out the last item in the arraylist when I want to undo. Then copy the number of each position in this 2D array to the current grid. Besides, also store the score of each grid in a scorelist, set the score to the previous one when undo.

Yujie W's part:

Firstly, I will discuss how does our complexity works for minesweeper and sliding tiles. We only implement complexity in those two games since it is very wired to implement different game modes in game 2048. Specifically, the way for us to implement the mode change between hard, medium and easy mode is quite simple, which requires no major change, since classes were well designed. As for unit tests, I will discuss during the presentation to explain why we have some 0% line coverage and how we get some 100% line coverages.

Lantao Cui part:

I was responsible for the UI design and part of the sign up and log in activity. As for the UI, I designed and implemented a splash animation after entering the software, I also designed the log in(sign up) layout. In addition to that, I designed a welcome screen for new users once they finished registering the email. At last, I designed the layout of every game's menu, and come up with a bouncing button animation for some buttons. For the log in qactivity, I implements methods that detect whether the current user name is registered, if yes, i will detect whether the password is correct. And if the current username is not registered, I allow the user to sign up for a new user.

Wenshuo Li part:

I was responsible for the scoreboard and users interaction. I used MVC design pattern for the scoreboard. I also designed the User, UserManager, Score class. I also implemented the save and auto save for the game 2048, and other serialization related stuff.