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.

Yuxin Yang additional part: The auto save and the preventing unsolvable sliding tiles. 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.

Jinda Huang part: Mostly the undo part. We have undo functionality for game2048 and sliding tiles. For sliding tiles, I store every position of blank tile of the board in a list, and take it out when I want to undo. For game2048, I store the grid (4 * 4 = 16 cards) in a 2D int array in each state, then store it in an arraylist, take it out when I want to undo.