PRESENTATION WALKTHROUGH - group_0619

1. Design Patterns and what problems each of them solves:
    a. Observer
        i. Observer: GameActivities
        ii. Observable: Boards
        iii. Problems solved:
            1. Separates Model and View by making view observe model.
            2. Reduces coupling
            3. Makes testing easier
    b. Iterator
        i. For the Boards to iterate through tiles/boxes
            1. Cleaner code (ie iterating vs nested for loops)
    c. MVC
        i. Model: Boards + Tiles/Boxes
        ii. View: Game Activities
        iii. Controller: Gesture + Movement Controller + Game
        iv. Problems solved:
            1. Removes coupling between model and view
            2. Single responsibility
            3. Easier for testing
    d. General software design
        i. Refactoring: Removing static variables and using bundles
        ii. Problems solved:
            1. Avoid changing the static variable when switching activities
            2. Less room for error when saving a score to the right save file
2. Implementation of scoreboard:
    a. Design: arraylist of Score objects
    b. Storing scores: in each GameActivity and loadFromFile/saveToFile methods
    c. Managing scores: sortAscending and sortDescending
    d. Retrieving scores: getScores and getScores for a particular player
    e. Displaying scores: ScoreboardActivity
3. Unit tests
    a. 100% example: TFBoard
        i. Tested all of the methods in different scenarios. (Merging identical boxes; corner boxes)
        ii. Covered the situation where the BoxGenerator creates a new box on top of the pre existing one (RNG test in TFBoard)
    b. 0% explanation: TFActivity + TFSettings
        i. Serves as the front-end. There are some logic to them as in save_to_file for we decided to save files when change activities.
4. Notable Code/Classes
    a. Sliding Tiles Design:

    i. Unlimited undo function: Arraylist of opposite moves
    ii. Generating an always solvable board
b. Minesweeper Design:
    i. When the player taps a blank tile, revealing the right tiles was done by adding each blank tile into a stack
    ii. Board generation algorithm
c. 2048 Game Design:
    i. Detecting swipes: the ability to swipe boxes to the 4 sides of the board was implemented through 4 methods (one for each direction)
    ii. Undo function: Each time a move was made, information was pushed to these stacks. Every time an undo was made, the info was popped of the stack and used.
d. Login + Firebase + addons(Encryption)
    i. Applied Firebase Backend system in storing files.
    ii. Log in through cross-checking hashed passwords + username.