Presentation Outline

Introduction

Game demo

Unit test demo

- Line coverage for model classes (except saver, SelfDefinedTile (related to bitmap)): > 75%
- Line coverage for view classes, adapter, exception classes: 0 %
- We try to test most methods in controller class, but some methods in controllers, are related to observer, or linked to view(such as button, toast message) which cannot be tested, so controller unit tests have low test coverage.

Overview of the code:

Most important classes: Session, Classes related to Scoreboard, Saver.

Introduction to Scoreboard:

- We have a class called score tuple. This is class will be generated when every game is ended. It stores three info which is the current player, the final score, and the current Game type. This class implements comparable. The reason why we need comparable is because we need a way to sort all scores when it get added in an array list.
- For ScoreManger, it follows singleton design pattern. It cannot be instantiated. it has two HashMap, the first one is called Usermap. The key is the user name and the value is an array list of score tuples the second one is called Topmap. The key is the game type, the value is an array list of ScoreTuples.
- For the Scoremanager, we have two method, add_score and check_top.
  Add_score takes a score and the current user name. It adds the score to the user map by the current user name. After we added the score, the list sorts And it uses saver to save the user map after the score is added check_top takes a score and game type. It adds the score to the top map by current game type.
- After we added the score, the list sorts And it uses saver to save the top map after the score is added.
- The User scoreboard activity generates 10 textviews and put the right ScoreTuple string from Usermap and put it on the right textview.
- The top10 scoreboard activity generates 10 textviews and put the right ScoreTuple string from Topmap and put it on the right textview.

Design pattern:

Model View Controller;
Observer/Observable;
SOLID principle;
Singleton;
Multi-thread;
Generic method;