CSC207 Project Presentation

Group 0616

Unit Test Coverage

The coverage of all classes need to be tested are above 80%,

with several 100%s

Most Important Classes

For the games:

- board managers
  - controls the behaviour of the boards
- movement controllers
  - controls the behaviour of the gridviews
  - composites a board manager so that it supervises the board manager (and indirectly the board) corresponding to the user's interaction (tapping, swiping, etc) to the gridviews

For the game centre:

- profile fragment
  - displays the user's profile information, according to data in Firebase Database.
- games fragment
  - displays the icon for the games, allowing user to choose the desired game to start.
- scoreboard fragment
  - display the user's score information, according to data in Firebase Database.

What problems do each of the design patterns solve?

- Iterator
  - A way to iterate over the elements (i.e. tiles) of our container (i.e. the board) in a “for each” loop
- MVC (Model-View-Controller)
  - We refactored our code from Phase 1 to correspond with MVC pattern, so that model classes contain the data, view classes present the data and controller classes modify the data
  - Convenient for testing: only controller(logic) classes need to be tested
- Observer
  - A way to pass data from models (observable objects) to views (observer objects) because a view depends on a model. A view updates itself when notified of its models changes

How did you design your scoreboard?

1. We made scoreboard as an option in bottom navigation bar of the game centre.

2. As soon as the Scoreboard Fragment is loaded, we fetch the score information from the Internet, by querying through Firebase Database.