Group 0622 Walkthrough

Unit Test Coverage

Sliding package: controller classes, with their respective coverage percentages in parentheses, are SlidingBoard (100%), SlidingBoardManager(100%), SlidingBoardSolvable (94%), SlidingGameFile (100%), SlidingTile(100%).

Twenty package: TwentyBoard (97%), TwentyBoardManager (43%), TwentyGameFile (100%) and TwentyTile (96%).

Checkers package: CheckersBoard (100%), CheckersBoardManager (99%), CheckersGameFile (100%), CheckersTile (100%)

CoreClasses package: Account (41%), Game (100%), GameScore (100%), Leaderboard (8%), Board (75%), GameFile(86%)

Remaining classes were excluded because they are model or view classes, which update views, gestures, or transition between activity classes.

Most important classes

The essential features of the game center application are managing users, playing games and scoring completed game. Consequently, the most important classes are as BoardManager, AccountManager and LeaderBoard.

Board Manager - this super class is responsible for the operations of boards during gameplay including registering movements of tiles, determining if a game is complete, calculating scores and undoing moves. It also provides inheritance structure for implementation of game specific board managers.

Account manager - this class tracks the users in the GameCentre app, which includes storing account data and setting the active account to store data about games played.

Leaderboard - this class stores high scores and updates scores for completed games by game and by user.

Design patterns

We used the iterator, strategy ,and observer design patterns in our code.

The iterator pattern was used to traverse tiles in the board class. The board class contains a two-dimensional array of tiles. This pattern decouples iteration algorithms from container classes.

The strategy design pattern was used to decouple saving and loading serialized files in activity classes and board manager, sliding board manager, twenty board manager and checkers board manager classes. By delegating save and load methods to the Savable class, we were able to more extensively unit test our board manager classes.

The observer pattern was used in the game activity class to watch the board manager classes undo, touchmove and game complete methods and update the gridview accordingly. This design pattern maintains the dependence between these classes without making assumptions about the behaviour of board manager objects.

Scoreboard

The scoreboard was implemented in the LeaderBoard class, which contains two hashmaps composed of array lists of GameScores, which records the scores of completed games. The global hashmap is static and records scores across users for each game, whereas the personal hashmap records per-user scores for each game. LeaderBoard displays the top 20 scores. The leaderboards are serialized and the hashmaps are stored in .ser files. Scores are updated in leaderboard when games are completed in the movement controller classes. Finally, leaderboards are displayed in the personal and global leaderboard activities, which read the data in the hashmaps from the .ser files into listviews.