











CSC207 Group 0612 Walkthrough

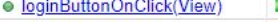
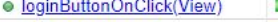


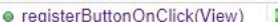
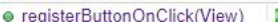
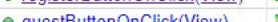
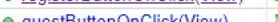
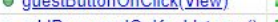
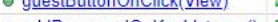
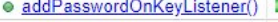
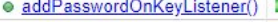


- What is your unit test coverage?

Element	Class, %	Method, %	Line, %
BoardManager	100% (1/1)	100% (17/17)	100% (98/98)
Board	100% (2/2)	100% (16/16)	100% (52/52)
Account	100% (1/1)	100% (15/15)	100% (34/34)
Tile	100% (1/1)	100% (8/8)	100% (16/16)
ScoringSystem	100% (1/1)	100% (1/1)	100% (7/7)
AccountManager	100% (3/3)	100% (13/13)	94% (69/73)

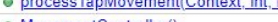
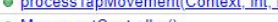




CustomAdapter

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getView(int, View, ViewGroup)		100%		100%	0	2	0	6	0	1
CustomAdapter(ArrayList, int, int)		100%		n/a	0	1	0	6	0	1
getItem(int)		100%		n/a	0	1	0	1	0	1
getCount()		100%		n/a	0	1	0	1	0	1
getItemId(int)		100%		n/a	0	1	0	1	0	1
Total	0 of 52	100%	0 of 2	100%	0	6	0	15	0	5

















LaunchCentre

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
loginButtonOnClick(View)		100%		100%	0	3	0	15	0	1
onCreate(Bundle)		100%		n/a	0	1	0	11	0	1
registerButtonOnClick(View)		100%		50%	1	2	0	6	0	1
guestButtonOnClick(View)		100%		n/a	0	1	0	5	0	1
addPasswordOnKeyListener()		100%		n/a	0	1	0	3	0	1
LaunchCentre()		100%		n/a	0	1	0	2	0	1
getAccountManager()		100%		n/a	0	1	0	1	0	1
Total	0 of 177	100%	1 of 6	83%	1	10	0	43	0	7

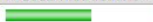







MovementController

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
processTapMovement(Context, int, boolean)		100%		100%	0	3	0	6	0	1
MovementController()		100%		n/a	0	1	0	3	0	1
setBoardManager(BoardManager)		100%		n/a	0	1	0	2	0	1
Total	0 of 39	100%	0 of 4	100%	0	5	0	11	0	3

PreferenceManager

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
PreferenceManager(Activity)		100%		50%	1	2	0	9	0	1
setWidgetPreferences()		100%		50%	1	2	0	5	0	1
wipeLoginData()		100%		n/a	0	1	0	5	0	1
storeLoginData(String, String, boolean)		100%		n/a	0	1	0	4	0	1
storeString(String, String)		100%		n/a	0	1	0	3	0	1
storeBool(String, boolean)		100%		n/a	0	1	0	3	0	1
retrieveString(String, String)		100%		n/a	0	1	0	1	0	1
retrieveBool(String, boolean)		100%		n/a	0	1	0	1	0	1
Total	0 of 131	100%	2 of 4	50%	2	10	0	31	0	8

SlidingTilesScoreManager

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
buildDisplayUserScoresList()		100%		75%	1	3	0	8	0	1
buildDisplayGameScoresList()		100%		75%	1	3	0	7	0	1
SlidingTilesScoreManager(String, Context, Integer)		100%		100%	0	2	0	5	0	1
buildGameScoresList()		89%		87%	1	5	2	15	0	1
Total	8 of 183	95%	3 of 18	83%	3	13	2	35	0	4

- What are the most important classes in your program?
 - Account
 - AccountManager
 - These classes hold the values to all account credentials, account scores, and account saves, most features for the implemented games such as scoreboards and saves require an account (can be a guest, which is set up as a default guest account), without these two classes, the functionality of our app beco
- What design patterns did you use? What problems do each of them solve?
 - Observable pattern in obdodger - notifies activity that the game is over when the player collides, leading to switch to scoreboard
 - Strategy in scoreboard? Different game score managers are created depending on the last game played
 - Dependency Injection in the Scoreboard: takes input of the account, current score, and game to be able to initialize the proper scoremanager, This allows the use of one activity class for game over scoreboards
 - Dependency Injection in the Scoremanager: takes in the score and the username to find out if there is a list of scores to display for the user, and uses the saved accounts to make the sorted lists of scores for display
- How did you design your scoreboard? Where are high scores stored? How do they get displayed?
 - Scoreboard comes up after the game is over, and can also be accessed from the game selection screen
 - Game over

- Each game must pass the account and the score, as well as the last game played
- The scoreboard uses abstract class scoremanager, which is implemented individually for each game since they have different ways of storing the scores
- The scores are stored in the accounts, so each score manager loads the accounts and generates the list of game wide scores for each game and sorts them, as well as load the user's own scores in an independent list of user scores (also sorted)
- There is a button to change the display of the scoreboard. If the display is game-wide scores, then it will switch to the user's own scores, and vice versa
- There is an option to play again or return to game selection on the scoreboards
- Game selection
 - Users can see game-wide scoreboards from the game selection screen for each game by swiping left and right on the screen