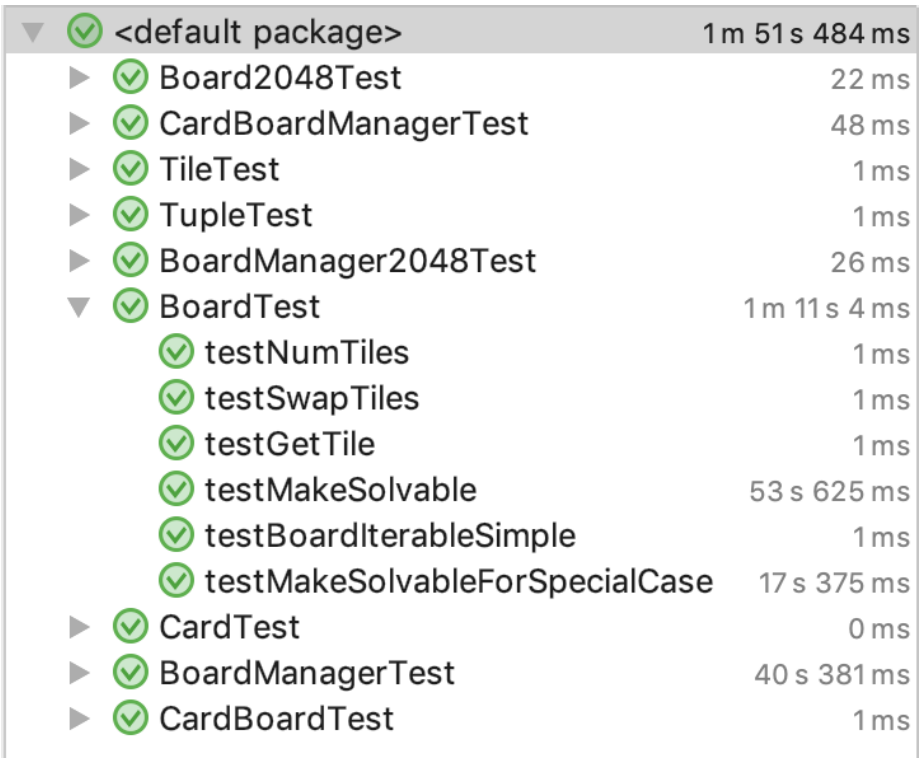
Presentation walkthrough

Zhouyue Lyu, Jiahe Lyu and Thomson Yen

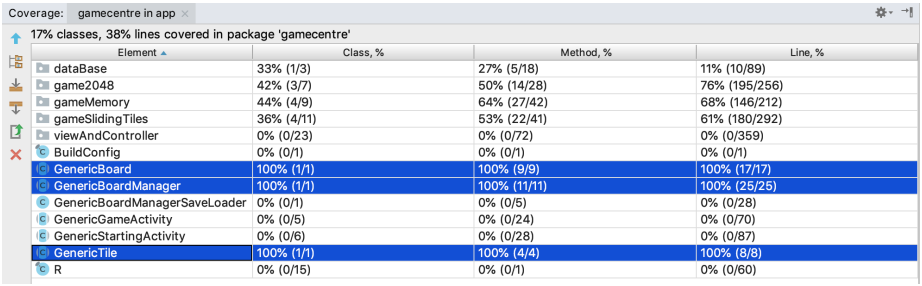
December 4, 2018

1 Unit test coverage

Tests under **Test** folder



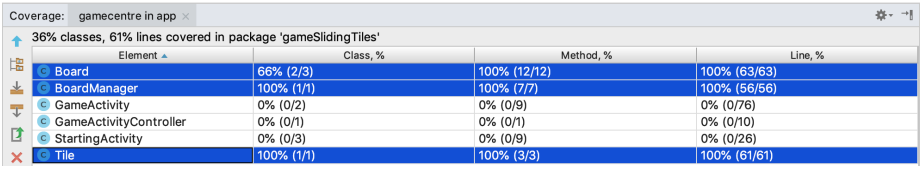
Overall result


Coverage: gamecentre in app

17% classes, 38% lines covered in package 'gamecentre'

Element	Class, %	Method, %	Line, %
dataBase	33% (1/3)	27% (5/18)	11% (10/89)
game2048	42% (3/7)	50% (14/28)	76% (195/256)
gameMemory	44% (4/9)	64% (27/42)	68% (146/212)
gameSlidingTiles	36% (4/11)	53% (22/41)	61% (180/292)
viewAndController	0% (0/23)	0% (0/72)	0% (0/359)
BuildConfig	0% (0/1)	0% (0/1)	0% (0/1)
GenericBoard	100% (1/1)	100% (9/9)	100% (17/17)
GenericBoardManager	100% (1/1)	100% (11/11)	100% (25/25)
GenericBoardManagerSaveLoader	0% (0/1)	0% (0/5)	0% (0/28)
GenericGameActivity	0% (0/5)	0% (0/24)	0% (0/70)
GenericStartingActivity	0% (0/6)	0% (0/28)	0% (0/87)
GenericTile	100% (1/1)	100% (4/4)	100% (8/8)
R	0% (0/15)	0% (0/1)	0% (0/60)

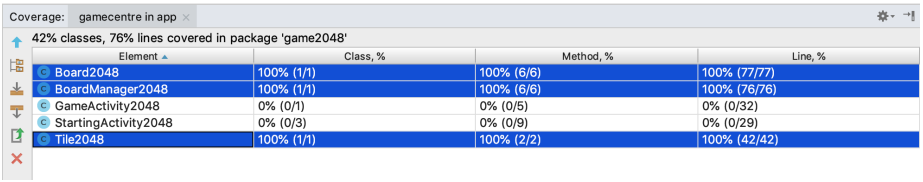
result for **gameSlidingTiles**


Coverage: gamecentre in app

36% classes, 61% lines covered in package 'gameSlidingTiles'

Element	Class, %	Method, %	Line, %
Board	66% (2/3)	100% (12/12)	100% (63/63)
BoardManager	100% (1/1)	100% (7/7)	100% (56/56)
GameActivity	0% (0/2)	0% (0/9)	0% (0/76)
GameActivityController	0% (0/1)	0% (0/1)	0% (0/10)
StartingActivity	0% (0/3)	0% (0/9)	0% (0/26)
Tile	100% (1/1)	100% (3/3)	100% (61/61)

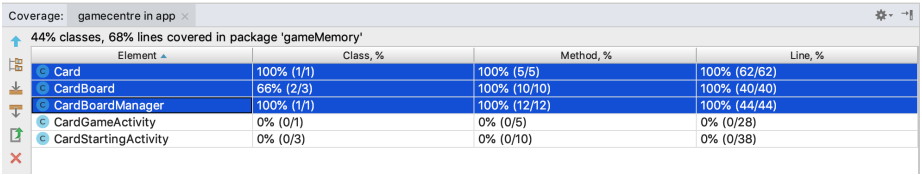
result for **game2048**


Coverage: gamecentre in app

42% classes, 76% lines covered in package 'game2048'

Element	Class, %	Method, %	Line, %
Board2048	100% (1/1)	100% (6/6)	100% (77/77)
BoardManager2048	100% (1/1)	100% (6/6)	100% (76/76)
GameActivity2048	0% (0/1)	0% (0/5)	0% (0/32)
StartingActivity2048	0% (0/3)	0% (0/9)	0% (0/29)
Tile2048	100% (1/1)	100% (2/2)	100% (42/42)

result for **gameMemory**


Coverage: gamecentre in app

44% classes, 68% lines covered in package 'gameMemory'

Element	Class, %	Method, %	Line, %
Card	100% (1/1)	100% (5/5)	100% (62/62)
CardBoard	66% (2/3)	100% (10/10)	100% (40/40)
CardBoardManager	100% (1/1)	100% (12/12)	100% (44/44)
CardGameActivity	0% (0/1)	0% (0/5)	0% (0/28)
CardStartingActivity	0% (0/3)	0% (0/10)	0% (0/38)

The other part of the classes are covered by **instrumented test** (since we find it really hard to mock the context and view), the result is shown below:
Tests under **androidTest** folder

Test Summary

13 tests	0 failures	3.492s duration	100% successful

Packages | **Classes**

Class	Tests	Failures	Duration	Success rate
group_0617.csc207.gamecentre.DatabaseHelperInstrumentedTest	6	0	0.692s	100%
group_0617.csc207.gamecentre.gameSlidingTiles.GameActivityControllerTest	1	0	0.799s	100%
group_0617.csc207.gamecentre.viewAndController.LeaderboardActivityControllerTest	2	0	0.987s	100%
group_0617.csc207.gamecentre.viewAndController.MovementControllerTest	1	0	0.191s	100%
group_0617.csc207.gamecentre.viewAndController.ResultBoardActivityControllerTest	2	0	0.650s	100%
group_0617.csc207.gamecentre.viewAndController.ScoreboardActivityControllerTest	1	0	0.173s	100%

result for **databaseHelper**

debugAndroidTest > group_0617.csc207.gamecentre.dataBase

group_0617.csc207.gamecentre.dataBase

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Tuple		100%		n/a	0	5	0	10	0	5	0	1
DatabaseHelper		100%		83%	2	16	0	71	0	10	0	1

result for **controllers**

debugAndroidTest > group_0617.csc207.gamecentre.viewAndController — Source Files — Sessions

group_0617.csc207.gamecentre.viewAndController

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed
MovementController		100%		100%	0	5	0	16	0	3	0
ResultBoardActivityController		100%		100%	0	7	0	17	0	3	0
ScoreboardActivityController		100%		n/a	0	3	0	16	0	3	0
LeaderboardActivityController		100%		100%	0	11	0	26	0	3	0

2 Most important classes

- DataBaseHelper, which is the controller that create, add, update, and get data from SQlite database. All the user information: (username, password, highest score for three games under three difficulties) are stored in the database through DataBaseHelper.
- GenericBoardManager is the parent class of all types of BoardManager. It defines the basic interface to make a Board-like game works. This allow our controllers to know disregard types of BoardManager it is actually using. This also enables us to have other classes like Builder that specifically deal with GenericBoardManagers.
- GenericStartingActivity stemmed from originally independent activities for each Game. However, we noticed a great amount of overlap in code among the three classes and many of them are not game specific. Hence, we now put the overlapping components to this class and have each game's own activity to extend from it. On top of eliminating redundant code, this will also decrease the amount of workload in adding games. Bugs related to starting activities can also be dealt with at the same place.

3 Design Patterns

- BoardManagerBuilder (we used ealier, but delete it before uploading) used Factory Pattern, the Builder class itself also uses Singleton Pattern. Using these patterns gives us an unified syntax to construct BoardManager for different game. It also leads to a cleaner code in constructing a starting activity for all game. Also, it leads to less code in the activities and better test coverage.
- BoardManagerSaveLoader used Singleton pattern. The pattern allow us to eliminate duplicate code in the activities. It also leads to better test coverage.
- Activities, GenericBoardManager, and MovementController employ the MVC pattern. These are View, Model, and Controller respectively. This design pattern greatly reduce the logic needed in activities and leads to better coverage.
- LeaderboardListViewAdapter, ScoreboardListAdapter, CustomAdapter follows adapter Pattern, allows the interface of an existing class to be used as another interface. Here they are responsible for showing leaderboard, scoreboard and gameboard respectively.

4 Design of scoreboard

Score per user per game per difficulty are stored in the SQLdatabase

Each time when the game finished, the controller of the resultBoardActivity will compare the result score with the score in the database, and will store the higher one into database.

When the user click the scoreboard button in game center, it will trigger the scoreBoardActivity, and the controller for that will get the data (i.e., the highest score for all three games under different difficulties), build a data string and send it to scoreboard adapter, which will process the data string and present it as a table in relative xml.

Similarly, When the user click the leaderboard button in game center, it will trigger the leaderBoardActivity, and the controller for that will get the data (i.e., the highest score for all the players under that specific game, specific difficulty), build a data string and send it to leaderboard adapter, which will process the data string and present it as a table in relative xml.