WALKTHROUGH
Group_0672

- What is your unit test coverage?
- What are the most important classes in your program?
- What design patterns did you use? What problems do each of them solve?
- How did you design your scoreboard? Where are high scores stored? How do they get displayed?

GameCenter Main Classes :

- User (Serializable)
- UserManager (Singleton)
- ReadLoadFileManager
- ScoreBoard (abstract super Class)
- ScoreBoardManager
- GameManager (interface)
  - UndoableGameManager (subInterface, which can undo)

- ***Sliding Game***
  - Tile
  - Board
  - BoardManager (MVC) (UndoableGameManager)
  - MovementController (MVC)
  - SlidingMainActivity
  - GestureDetectGridView (MVC)
  - SlidingScoreBoard (child Class of ScoreBoard)
  - PictureManager

- ***Snake Game***
  - Point
  - Grid (Singleton)
  - Snake
  - SnakeManager (MVC) (UndoableGameManager)
  - SnakeGameController (MVC)
  - SnakeMainActivity
  - GameView (MVC)
  - SnakeScoreBoard (child Class of ScoreBoard)

- ***Mole Game***
  - MoleManager (MVC) (GameManager)

- MoleController (MVC)(Observable)
- MoleMainActivity (Observer)
- MoleScoreBoard (child Class of ScoreBoard)
- MoleStartingActivity
- SelectMoleGameActivity

Tests :

- UserAndUserManagerTest :
  contain tests which cover **User, UserManager, ScoreBoard, ScoreBoardManager**

- SnakeAndSnakeManagerTest :
  contain tests which cover **Point, Grid, Snake, SnakeManager, SnakeGameController, SnakeScoreBoard**

- BoardAndTileTest :
  contain tests which cover **Tile, Board, BoardManager, MovementController, SlidingScoreBoard**

- MoleControllerAndMoleManagerTest :

Design Parttern :

- Use **Serializable** to save all the game condition for each user
- Let UserManager to be a **Singleton** class, since we only need one UserManager to manage all the users.

- **<u>SlidingGame:</u>**
  - Use **MVC** design pattern, there is a MovementController to handle action between gameview and board manager. Tile, Board can be regarded as model.
  - Use **Iterator** in Board to iterate Tiles.
  - Let Board to be **Observable**, and SlidingMainActivity be **Observer.** Then whenever the board has changes , the game view will be updated.

- **<u>SnakeGame:</u>**

- Use **MVC** design pattern, there is a SnakeGameController to handle action between gameview and snake manager. Snake can be regarded as model.
- Let SnakeGameController be **Observable**, and SnakeMainActivity be **Observer,** then whenever user click to save the SnakeMainActivity will notice and pause game to save.
- Let Grid to be a **Singleton** Class, since we only one grid.

- **<u>MoleGame:</u>**

  - Use **MVC** design pattern, there is a MoleController to handle action between gameview and mole manager. MoleManager can be regarded as Model.

  - Let MoleController to be **Observable**, and MoleMainActivity be **Observer.** Then whenever the controller has changes , the hole background and background music will be updated.

Scoring System :

In scoring system, there are three ScoreBoardManager instances, each of them manages all the ScoreBoards from one type of the three games.
In each type of game, the game manager can be regarded as one round of the game which contains all the conditions of that round.
Each game manager comes with a score board that records the score of that round.
Once a round of game finished, the scoreboard manager of that type of game will add the scoreboard of that round into its scoreboards list, then sort the list to store the first 5 highest scoreboards.
Then user can click a button called ”Score History“ to check the score ranking.