By the Phase 2 due date, you must commit a file called `WALKTHROUGH.pdf` that contains an outline of your presentation. You should cover at least these points:

- 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?

Unit Test Coverage:
Excluding certain classes(view/model) listed in our group_0601 “README.md” file, these are our following unit test coverage for the three games:

Sliding Tiles

- 81%

Pong

- 94%

Cold War

- 90%

Most Important Classes:

- Pong/ColdWar/SlidingTiles Controller - Contains all the logic needed for the game to run.
- FileSaverModel - Key component for saving and loading save files / scoreboard.
- Pong/ColdWar/SlidingTiles View - Allows us to display the view to the user.
- Pong/ColdWar/SlidingTIles GameInfo - Contains all the info for the game such as the current score or lives etc.

Design Patterns Used and their benefits:

- MVC - Helped us divide our user interface application into three components. The Controller classes contains logic for the application, which acts on the model and view classes. The View classes presents information to our users, and has an update mechanism controlled by the controller classes, that updates the view whenever data changes. The model classes encapsulates data, including saving and loading game files. It provides functions to access its data that are used by the view component to display the appropriate user’s data. By separating these three components, it allows for independent unit testing.
- Observer Pattern - In sliding tiles for example, the GameActivity class was the observer and Board class was the subject. When a user plays the game and makes a move(swaps two tiles), the board state changes and notify its observer(GameActivity), who then updates the view to display the current board state to the user. This design patterns ensures flexibility, implementation and testing of dependent objects, that is objects that change depending on what the user does.

ScoreBoard Design:

- Made it serializable and saved it in a file called “master_save_file.ser”
- We store all the scores and the person that got that score. Whenever we want to see the highscores we go through the list of scores.
- They get displayed through ScoreBoardActivity, top 9 scores.