Presentation Walkthrough

Walkthrough Gameplay (~2 minutes)**:**

(Please note that when the project is presented, it will be accompanied by a powerpoint with screenshots to accompany each of the stages of gameplay.)

Welcome to our Phase 2 Project. In this project, we implemented an app that contains three games: a turn-based RPG named Ghost Hunt, a puzzle-solving game called Sliding Tiles, and a mathematical logic challenge --- Sudoku. These three games are packaged into a "Game Center" where they can be accessed. Each registered user of the app will also have access to their personal profile as well as a per-game and per-user scoreboard. We will discuss each of these components in detail.

When the app is ran with emulator, you will then be taken to an interface where a login prompt is displayed. If you are opening this project for the very first time, you will not be able to login since you do not have a signed-up account. To obtain an account, kindly click on "Sign up here" icon which will send you to the sign-up page.

Here, you will need a username and a password. Using regular expression, username input is restricted to uppercase and lowercase letters and underscore. Then, simply login and you will be taken to the "Game Center" page. Here you will have access to our games and your profile. By clicking on "User Center", you will be taken to your personal profile page. Here you can view your highest scores for each of the games. You can also change your password and upload your own profile picture.

Returning to the "Game Center" page, if you choose to click on any of "Ghost Hunt", "Sudoku", or "Sliding Tiles" button, you will be taken to that game's starting page. In your first run through the program, there will not be any loaded games. However, you will have the option to save an unfinished game and load it from this page later. Sliding Tiles is a game where the objective is to sort the tiles and move them in order, Ghost Hunt is a game where you, using the arrows buttons, help your character escape the map before the ghost catches him, and Sudoku, well, is Sudoku. In both Sliding Tiles and Sudoku you will be able to choose levels of difficulty while in Ghost Hunt the maps gradually becomes more challenging.

A scoreboard can also be accessed from the Game Center via button click. On it, the current player's ranking and score will be displayed along with the top 5 attempts from all users. However, if the app is opened for the first time, no data will be recorded.
After a game is finished, a scoreboard will also be displayed, and hopefully you have moved up the ranks.

Design Pattern Usage (~5 minutes):

(Please note that there are many more places where we implemented these design patterns. Due to the time constraint, we cannot list every example.)

Model View Controller:

Purpose: Easier unit tests, more organized code. This design pattern organizes code by their functions: "Model" stores information, View interact with users, and Controllers manipulates the model, updates view, and handles user interactions.

Example: The whole project follows this general pattern, more specifically, the three games strictly follows this pattern. This classification can be seen through the way we named our classes: classes that end in “FileHandler” are the “Model” files, classes that end in “Activity” are the “View” files, and classes that end in Controller are the “Controller” files.

Dependency Injection:

Solved Problem: Prevents the instantiation of a class without its required attributes. Reduce dependency and produces more testable code.

Example: Cell.java class in sudoku folder, in which the constructor, line 60, contains parameters that is needed for its attributes.

Observable:

Solved Problem: Connects model and view

Example: GameController.java in slidingtiles folder, on line 64, calls method notifyObserver() which notifies SlidingTilesGameActivity.java class in the same folder (the class is made an Observer by line 103 in its class body).

Strategy Pattern:

Solved Problem: Calculates different games’ scores differently.

Example: ScoreBoard.java abstract class and each of its subclasses that implements calculateScore() method according to the context of the game in which the scoreboard is displayed.

Singleton:

Solved Problem: Ensures that only one instance of FileHandler, “Model” of MVC, exists. This ensures that the same instance of GameState.java is shared among all associated files.

Example: Singleton GhostHuntFileHandler.java class in ghosthunt folder, which contains an instance of GameState (line 51), ensuring that one copy of the GameState is used by Model, View, and Controller.

S.O.L.I.D. Principle (~4 minutes):

Single Responsibility:
A classes have one specific purpose. An example of which is GameController.java class, which only handles game logic.

Open/Close Principle:
The app itself is open for extension, amore specific example of which is its Game enum, which can have additional elements. The app is also closed for modification in that an additional game does not affect codes in the existing games.

Liskov Substitution Principle:

Class Player.java and Ghost.java in ghosthunt folder both inherit Entity.

Interface Segregation Principle:
Undoable, Saveable, Loadable are all small interfaces which does not have unused methods that are of interest to the classes that extended them.

Dependency Inversion:
GameController.java in ghosthunt folder line 211, in which processEntityMove() method is called . However, the exact way that the entity handles the move does not affect the functionality of processEntityMove() method.

User Interface design core principles (~1 minute):

1. Clarity and Simplicity. Users should be well-informed and confident, and our ui serves both.
2. Flexibility. Most of our user interfaces are not static, they look good under different situations.
3. Familiarity. Similar user interfaces follow the same design pattern, so users should be familiar with the user interfaces.
4. Efficiency. Our ui is designed for users to complete their main tasks in the most efficient way without losing the results of their work.
5. Colour Theory. Our colour scheme is very nice, the colours blend together naturally.

Scoreboard (~1 minute):

There are strictly two different scoreboards, one of which is displays per-game highscores, one of which displays per-user highscores. Per-game scoreboards inherit superclass ScoreBoard.java. They sort each of the users by the score of the their attempts and saves them in "save_gh_scoreboard.ser", "save_sliding_tiles_scoreboard.ser", and "save_sudoku_scoreboard.ser" which belong in their corresponding FileHandler classes. Each per-game scoreboard displays the top five attempts' score and username in five TextView, and the current player's username and rank in another TextView.

Per-user scoreboard can be found in User Center page, which displays the high score for each of the games game using ListView. The reason ListView is implemented here but not in per-game scoreboards is that only the top five will be displayed in a per-game scoreboard while new games might be added, making per-user scoreboard larger. This ties back to SOLID principles were the per-user scoreboard is designed to be open for extension.

Unit Test (~1 minute):

Unit testing are primarily done on the Controller files in our MVC design, that is files that end in Controller.

100% Test Coverage Cases:

- SolvableGenerator.java in slidingtiles folder
- SudokuGameController.java in sudoku folder
- GameController.java in ghost_hunt folder

- GhostHuntScoreboardController.java in ghost_hunt folder
- GameState.java in ghost_hunt folder

All Implemented Functionalities (explicitly required):

- Code coverage where applicable (unit test)
- Always generate a solvable SlidingTiles game board (SolvableGenerator.java class in slidingtiles folder)
- Added two games and accommodating scoreboards
- Per-user scoreboard
- Undo feature for at least one of the new games (Ghost Hunt undo five moves)
- Autosave for at least one of the new games (Sudoku and Ghost Hunt)
- Design Pattern implementations (see Design Pattern Usage section)
- Updated test cases along with refactoring

All Other Functionalities (Features):

- Signup page
    - Used regular expression to restrain username to a permutation of uppercase letters, lowercase letters, and underscores. This ensures that usernames can be written into valid file names.
- Login page
- Game Center page
- User Center
    - Change Profile Picture
        - Default picture as our beloved Professor Danny Heap
    - Change Password
    - ListView format of per-user scoreboard can be scrolled down
- Sliding Tile Game
    - Displays current play name
    - Displays number of moves
    - Change Background
- Sudoku
    - Autosave on every move
    - Three levels of difficulty
    - Displays current play name
    - When a visible cell is selected, all other cells with the same value becomes highlighted.
    - Maximum three hints provided
    - Incorrect placement of numbers are displayed and taken into account in score calculation
    - Toast “Wrong Answer” becomes a more severe warning after four wrong answers
- Ghost Hunt
    - Autosave on the start of every level
    - Manual save game function
    - Restart level (player receives fresh undos but the number of moves does not reset)

    - Ghost AI that follows the player along a trajectory that is closest to a straight line
    - Ghost cannot catch player at the exit
    - Beating the levels and failing levels produces different end-game Toasts
- Overall visual upgrade