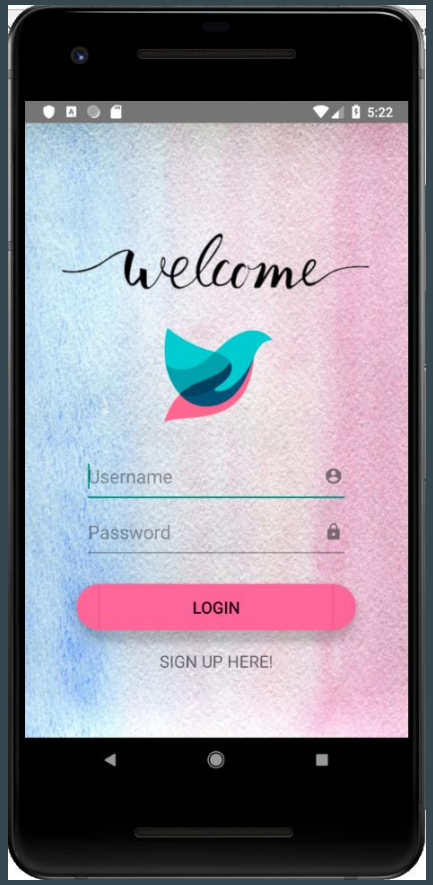
Project Phase 2

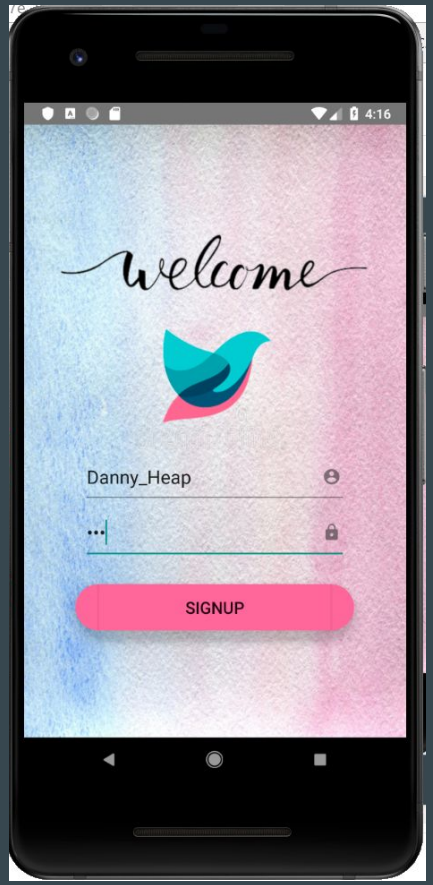
Group 0640

Gameplay Walkthrough

Signup & Login

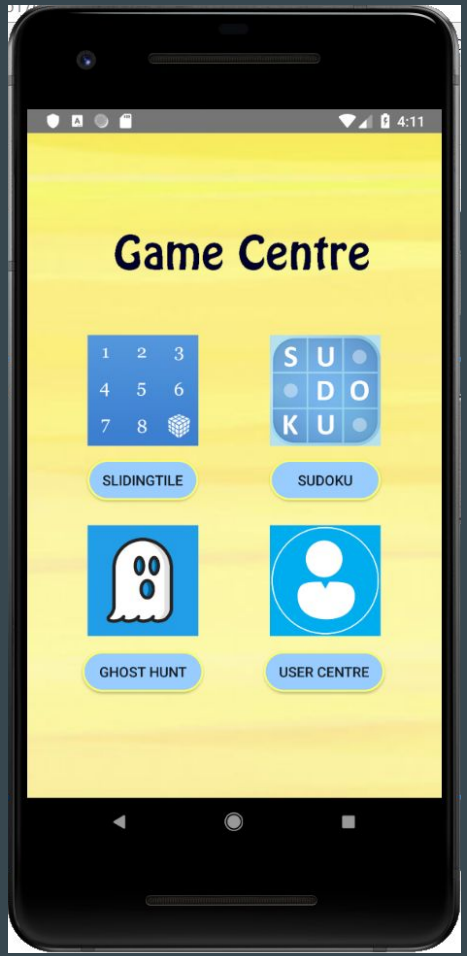
- Username
  - Regex restrains input
- Password





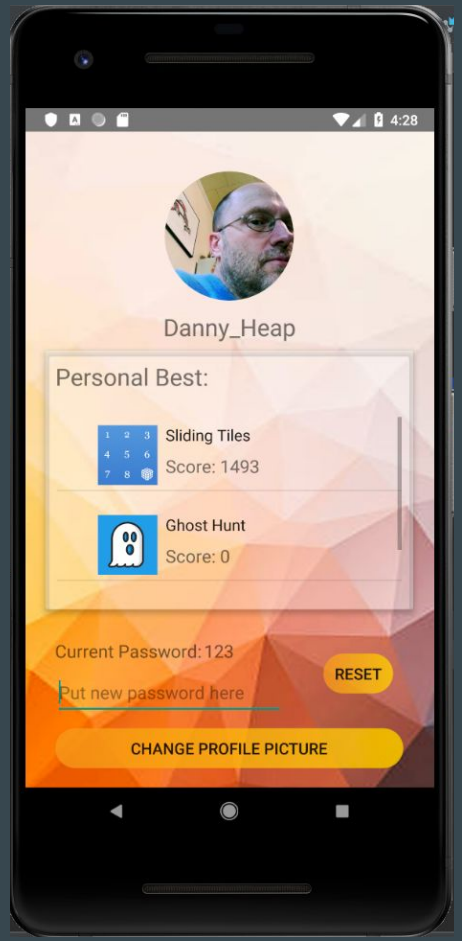
Game Centre

- Access to the 3 games
  - Sliding Tile
  - Sudoku
  - Ghost Hunt
- Access to User Center



User Centre

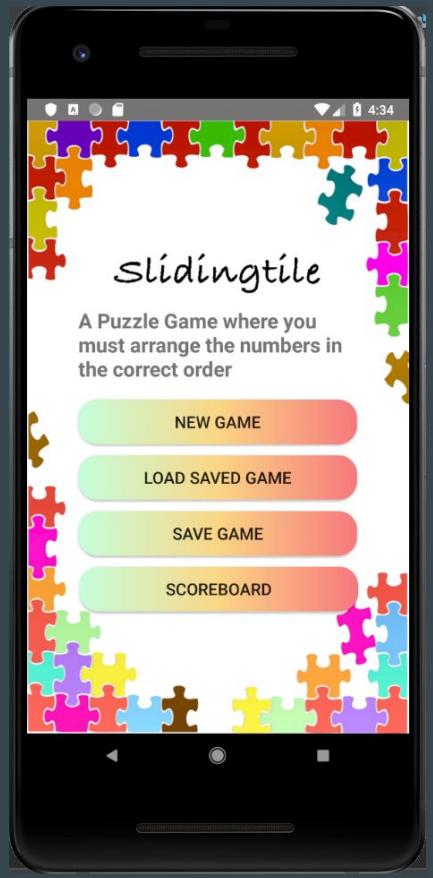
- Per-User Scoreboard
  - ListView display with scroll bar
  - Per-game high score
- Password Status
  - Display current pw
  - Reset password option
- Upload Profile Picture
  - Default image is Professor Heap



Game Starting Activity

Access to:

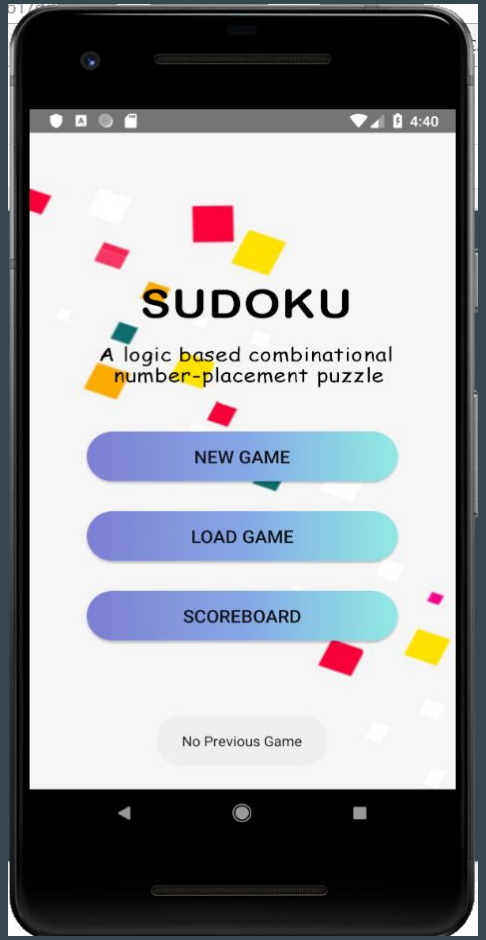
- New Game Button
- Load Game Button
- Manually Save Game Button
- Scoreboard



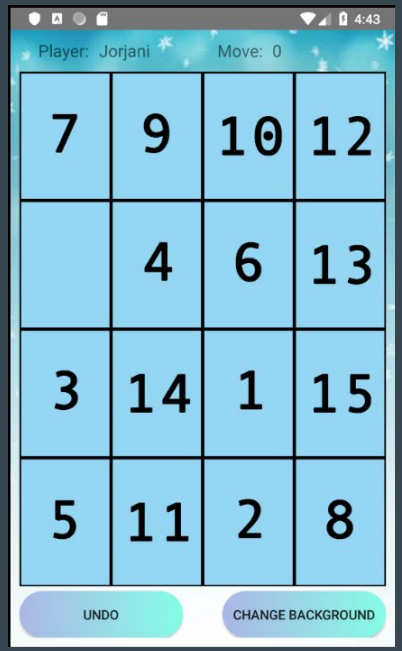
Game Starting Activity

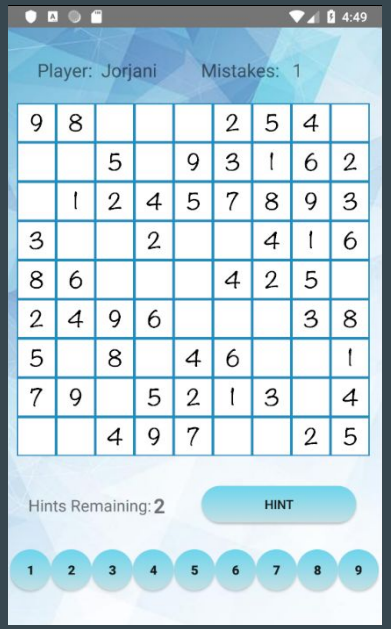
Access to:

- New Game Button
- Load Game Button
- Manually Save Game Button
- Scoreboard



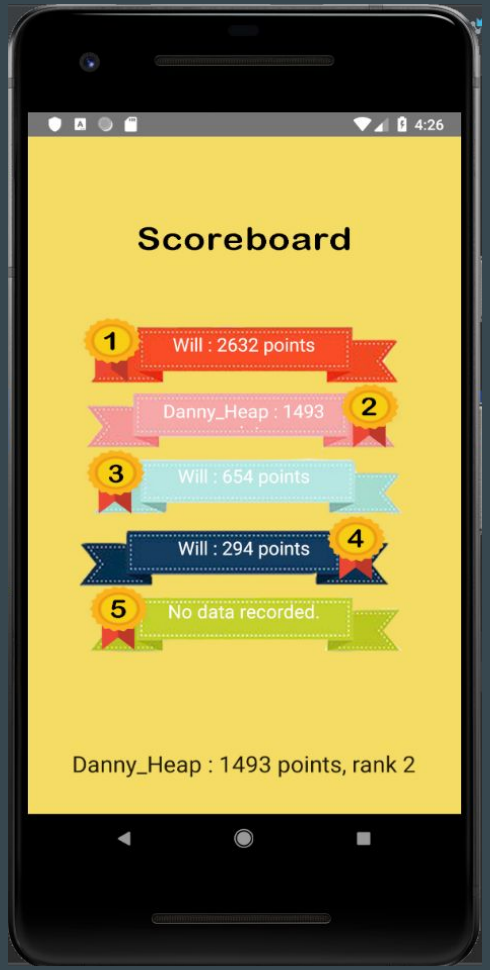
Game Activity



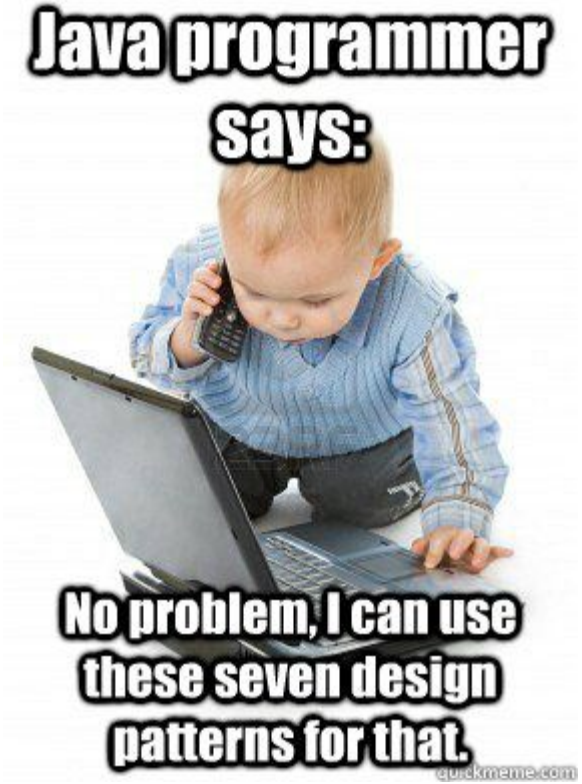


Scoreboard

- Displays top 5 of all attempts
- Displays current user's score and rank



Design Patterns



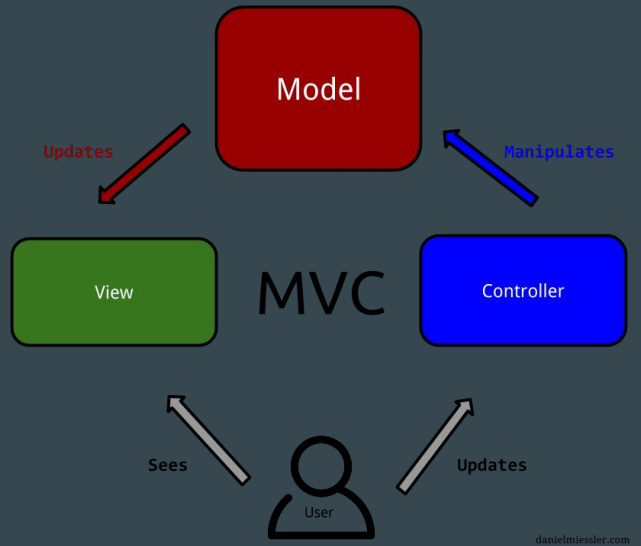
1. Model View Controller

ghost_hunt
- Board
- Direction
- Entity
- FileHandler
- GameController
- GameState
- Ghost
- GhostHuntGameActivity
- GhostHuntScoreboardActivity
- GhostHuntScoreboardController
- GhostHuntScoreboardFileHandler
- GhostHuntStartingActivity
- GridViewAdapter
- Player
- Tile

slidingtiles

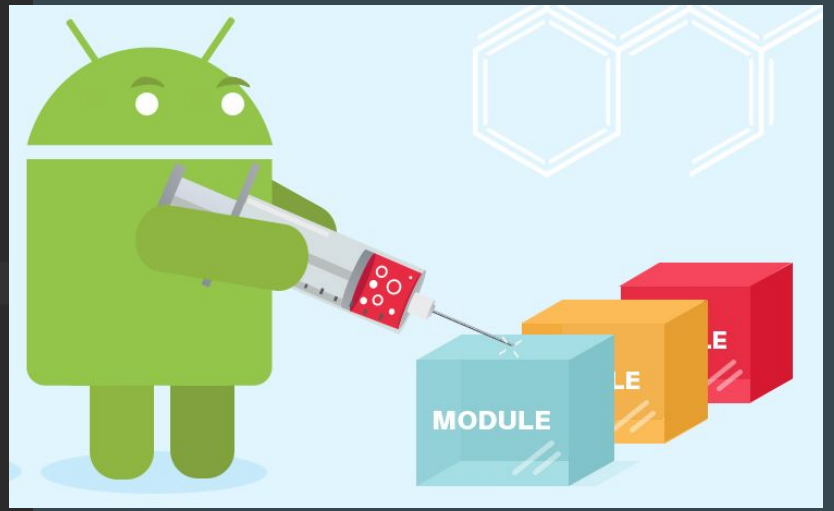
sudoku

slidingtiles
- Board
- GameController
- GameState
- GestureDetectGridView
- SlidingTilesAdapter
- SlidingTilesFileHandler
- SlidingTilesGameActivity
- SlidingTilesImageProcessor
- SlidingTilesScoreBoardActivity
- SlidingTilesScoreBoardController
- SlidingTilesScoreBoardFileHandler
- SlidingTilesStartingActivity
- SolvableGenerator
- Tile



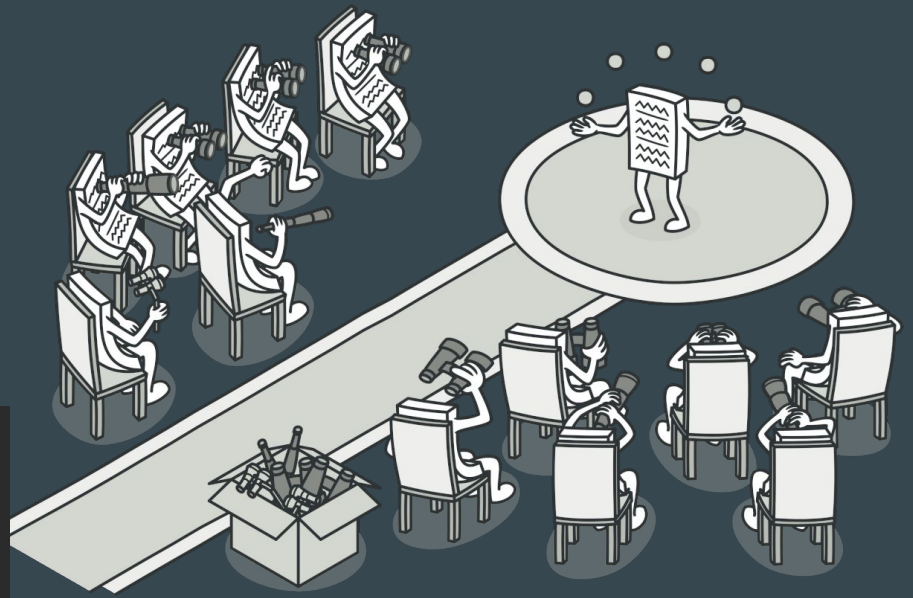
2. Dependency Injection

```
/**
 * Constructor of the cell.
 *
 * @param value     the preset value of the cell.
 * @param isVisible the visibility of the cell.
 * @param position  the position of the cell in the board.
 */
Cell(int value, boolean isVisible, int position) {
    this.value = value;
    this.isVisible = isVisible;
    this.position = position;
    if (!isVisible) {
        background = blankBackground;
    } else {
        background = numberBackground;
    }
}
```



3. Observable

```
void increaseWrongCounter(){
    wrongCounter++;
    if (wrongCounter == 4){
        setChanged();
        notifyObservers();
    }
}
```



```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    gameState = fileHandler.getGameState();
    createCellButtons( context: this);
    setContentView(R.layout.activity_sudoku_game);
    setUpGridView();

    gameController = gridView.getGameController();
    gameController.setGameState(gameState);
    gameController.addObserver( o: this);
    gameState.addObserver( o: this);
```

4. Strategy Pattern

```
public abstract class ScoreBoard {
```

```
public abstract int calculateScore(ArrayList<Integer> array);

/**
 * Updates currentUser's high score, if needed.
```

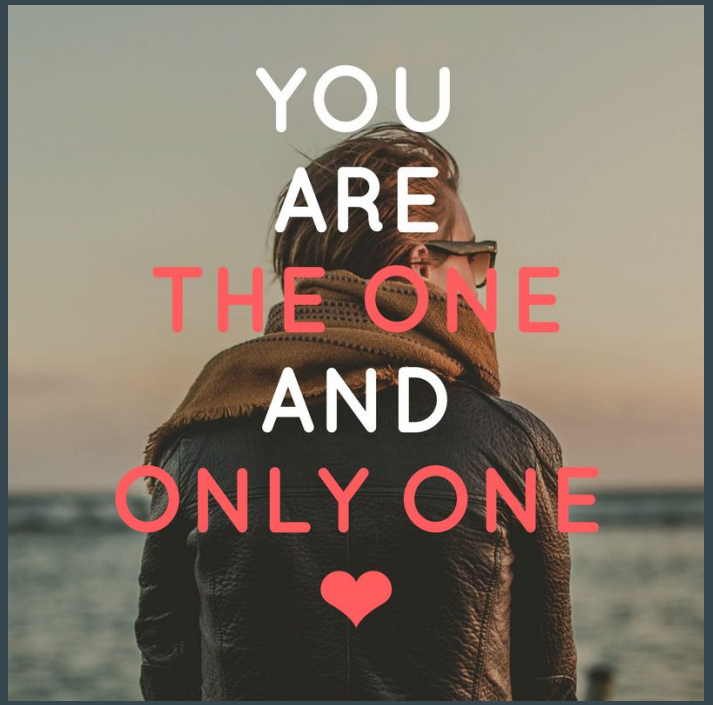
```
/**
 * ScoreBoard class for game.
 */
private ScoreBoard scoreBoard = new SudokuScoreBoardController();
```

5. Singleton

```
 * Handler for file IO in Ghost Hunt. Made into singleton.
 */
class FileHandler implements Savable, Loadable {
```

```
 /**
  * Sole instance of file handler.
  */
 private static final FileHandler INSTANCE = new FileHandler();
```

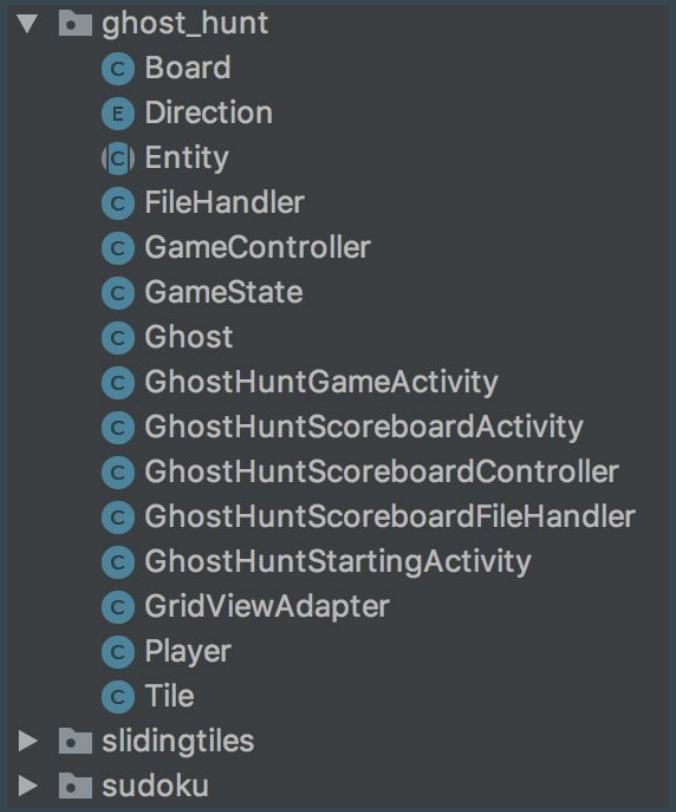
```
    /**
     * Private constructor for singleton.
     */
    private FileHandler() {}

    /**
     * Return the singleton instance.
     * @return sole instance of file handler
     */
    static FileHandler getInstance() { return INSTANCE; }
```

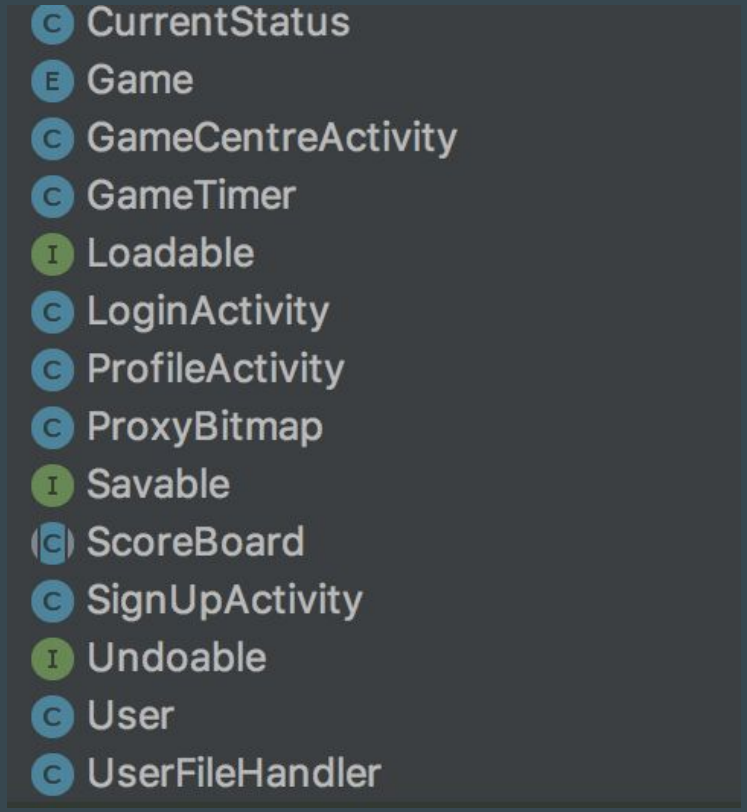


S.O.L.I.D

1. Single Responsibility

1. Single Responsibility





2. Open/Close principle

2. Open/Close principle

Game.java

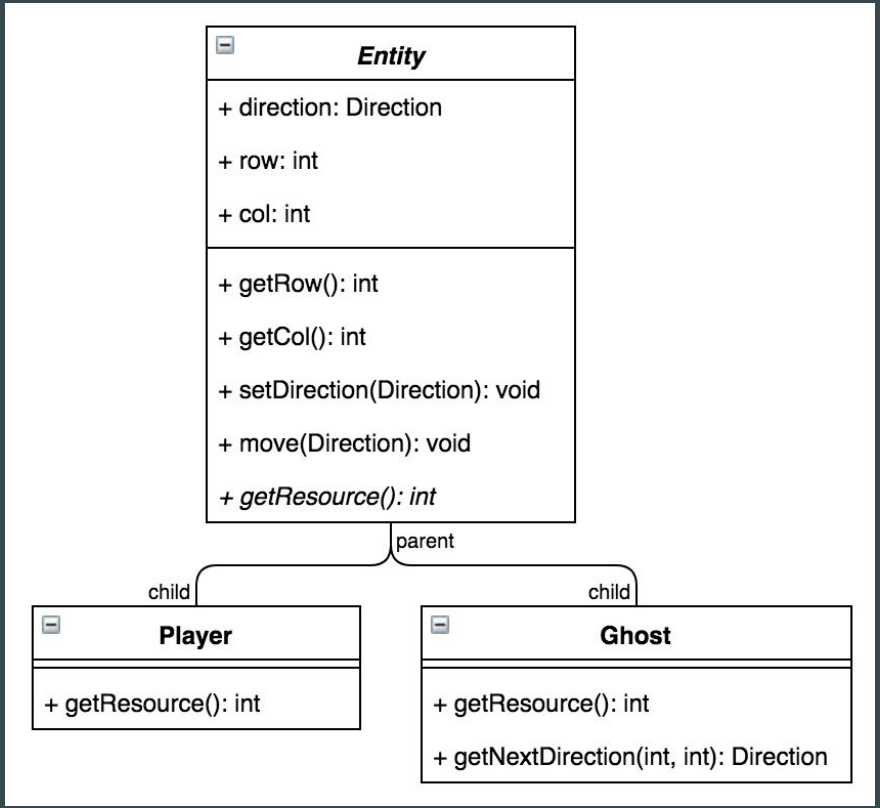
```
1   package fall2018.csc2
2
3   /**
4    * Enumeration of all
5    */
6   public enum Game {
7       SlidingTiles,
8       Sudoku,
9       GhostHunt
10  }
11
```

User.java

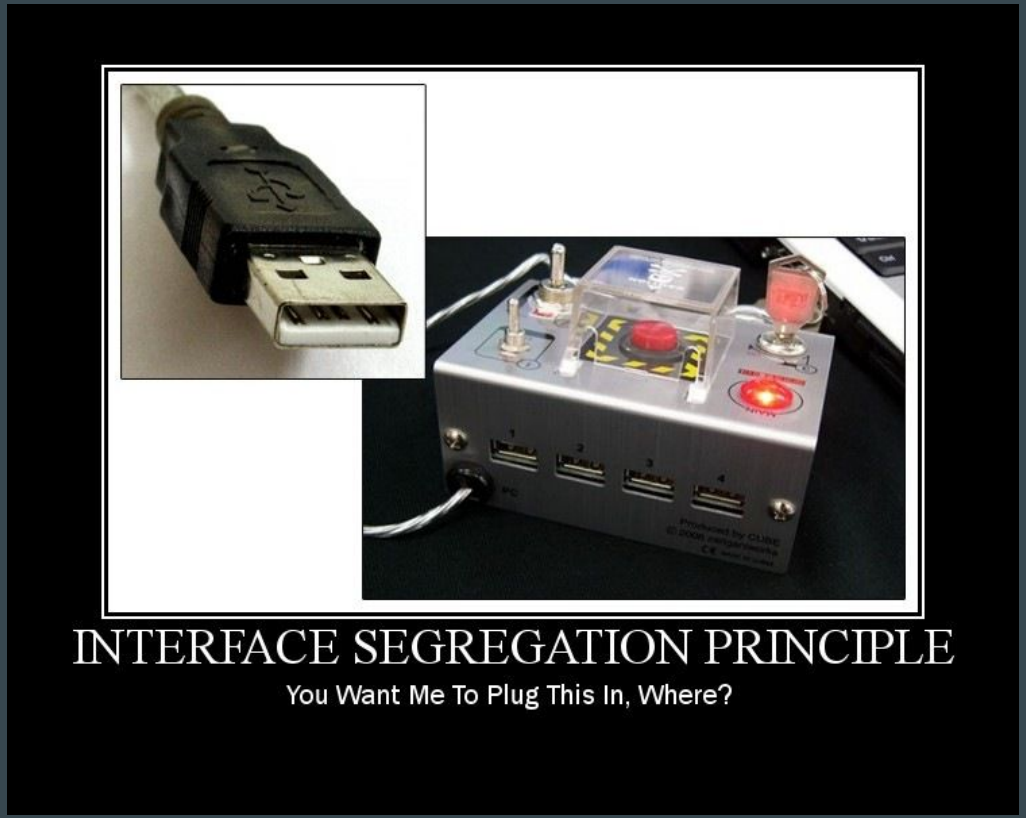
```
3   import ...
8
9   /**
10   * Model class, excluded from unit test.
11   * The user class, with user information.
12   */
13  public class User implements Serializable {
14
15      /**
16       * Username and password of the user.
17       */
18      private String username, password;
19
20      /**
21       * Mapping from game to highest score. Initialized for
22       */
23      private Map<Game, Integer> scores = new HashMap<>();
24
25      {
26          for (Game game : Game.values()) {
27              scores.put(game, 0);
28          }
29      }
```

3. Liskov Substitution Principle

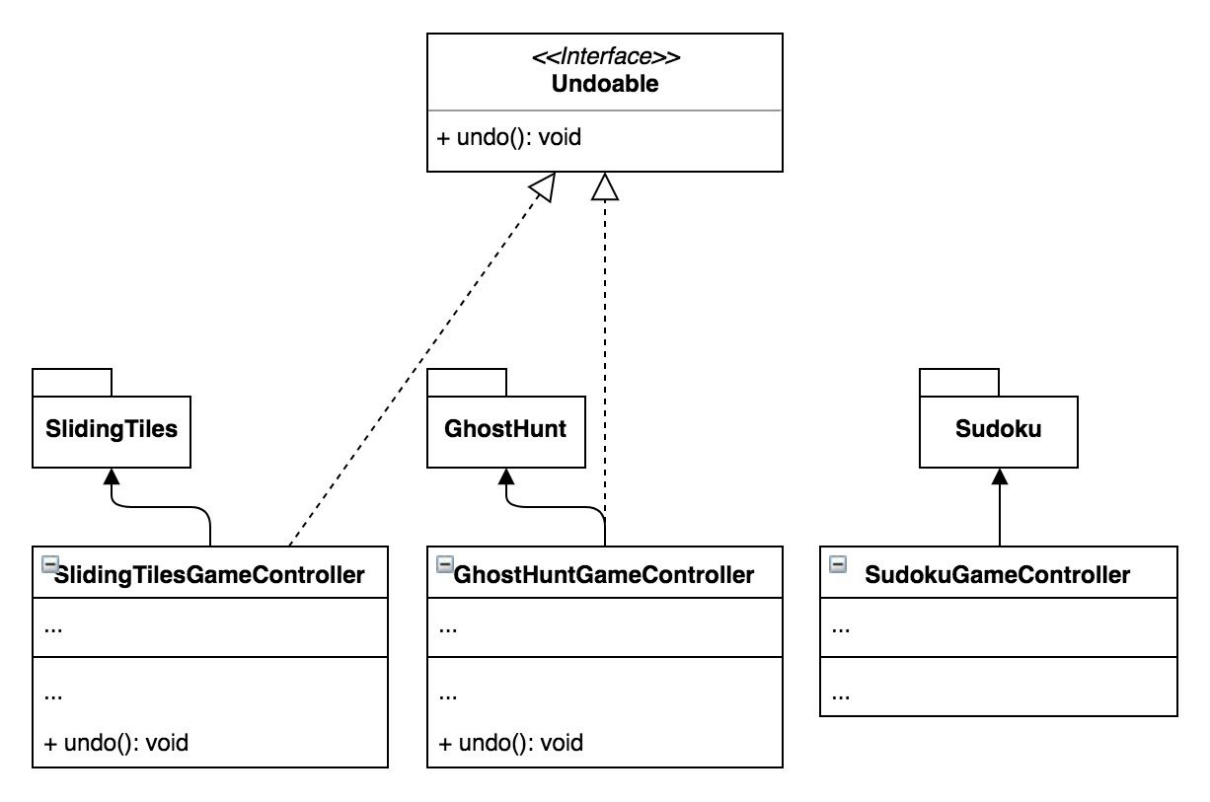
3. Liskov Substitution Principle



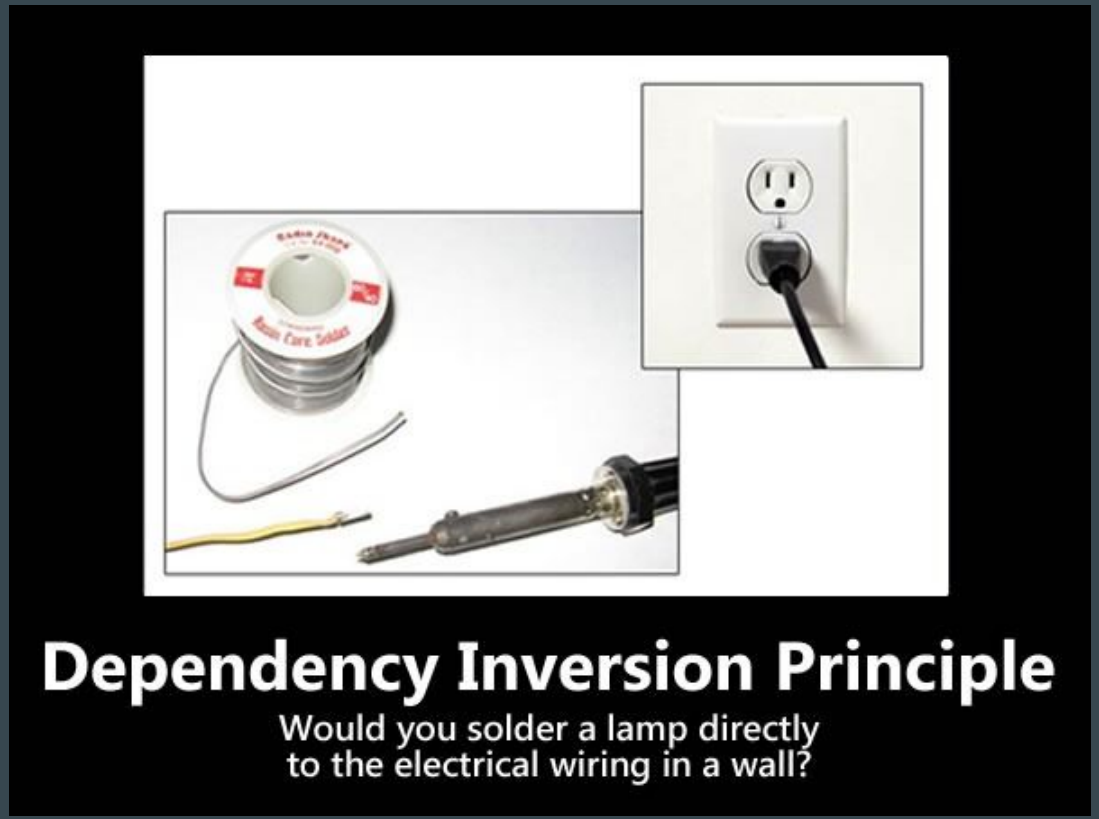
4. Interface Segregation Principle



4. Interface Segregation Principle



5. Dependency Inversion



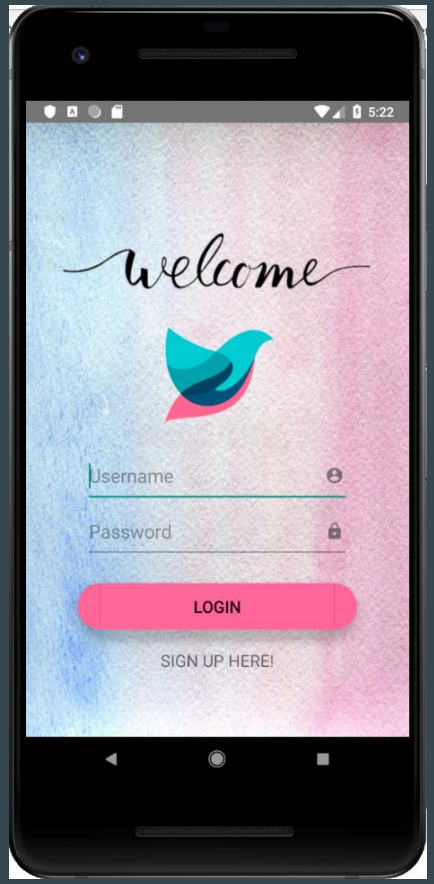
5. Dependency Inversion

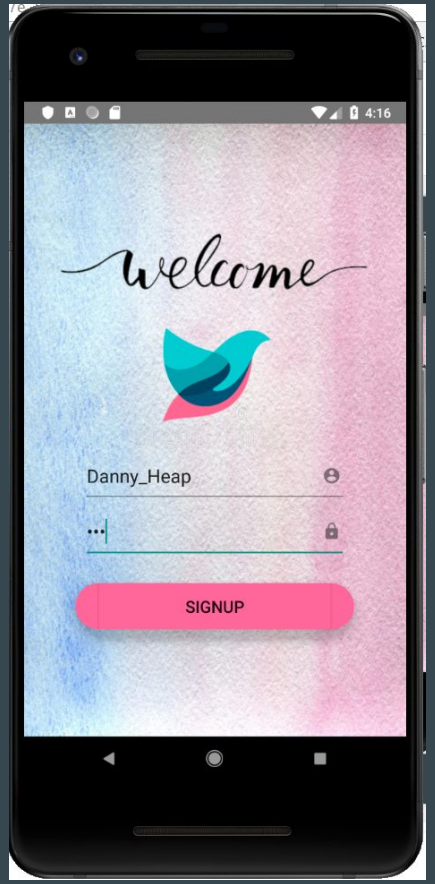
```
206     /**
207      * Process an entity's move on certain direction.
208      * @param entity entity to make move
209      * @param direction direction of move
210      */
211 @   private void processEntityMove(Entity entity, Direction direction) {
212         if (isValidMove(entity.getRow(), entity.getCol(), direction)) {
213             entity.move(direction);
214             appendMove(entity, direction,  isValid: true);
215         } else {
216             entity.setDirection(direction);
217             appendMove(entity, direction,  isValid: false);
218         }
219         notifyChange();
220     }
```

```
62      /**
63       * Entity makes a move.
64       * @param move direction of move
65       */
66      void move(Direction move) {
67          switch (move) {
68              case UP: row--; break;
69              case DOWN: row++; break;
70              case LEFT: col--; break;
71              case RIGHT: col++; break;
72          }
73          this.direction = move;
74      }
```

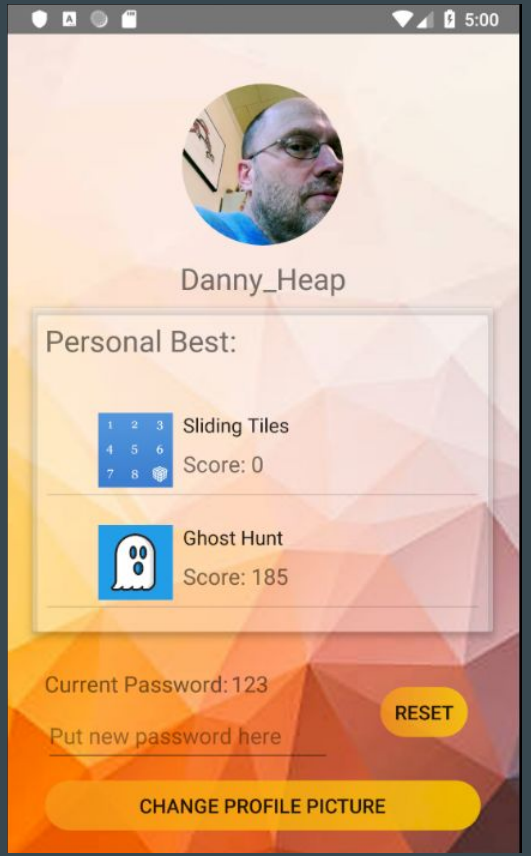
UI

1. Clarity & Simplicity

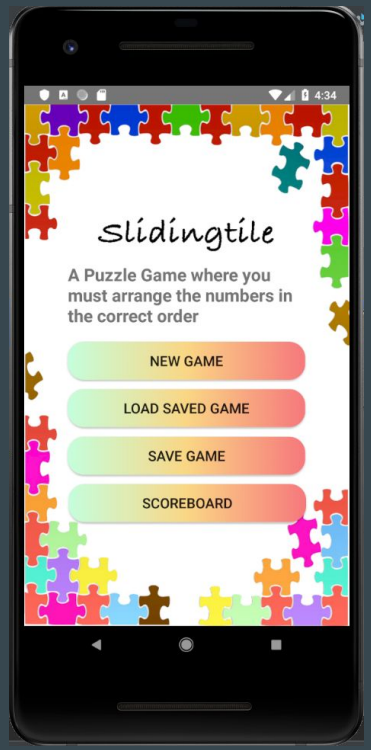


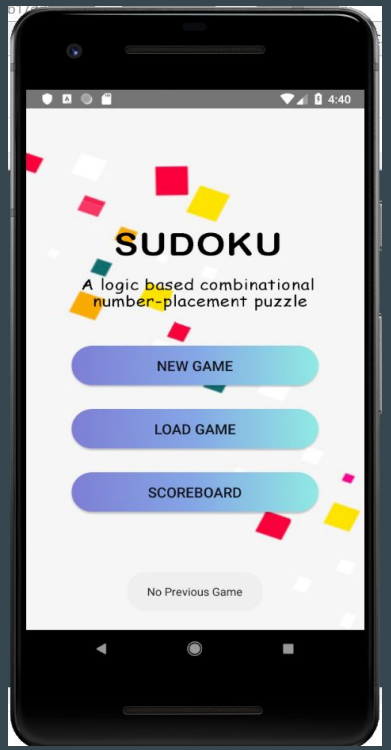


2. Flexibility

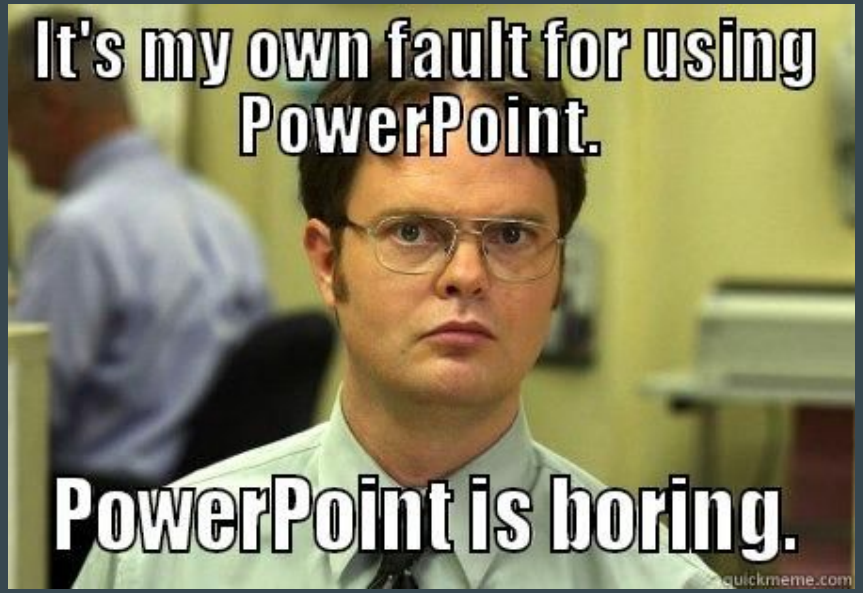


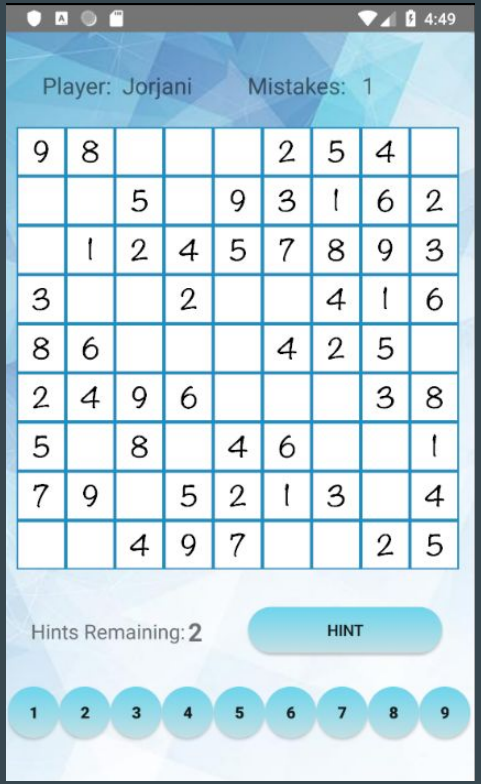
3. Standardized Format





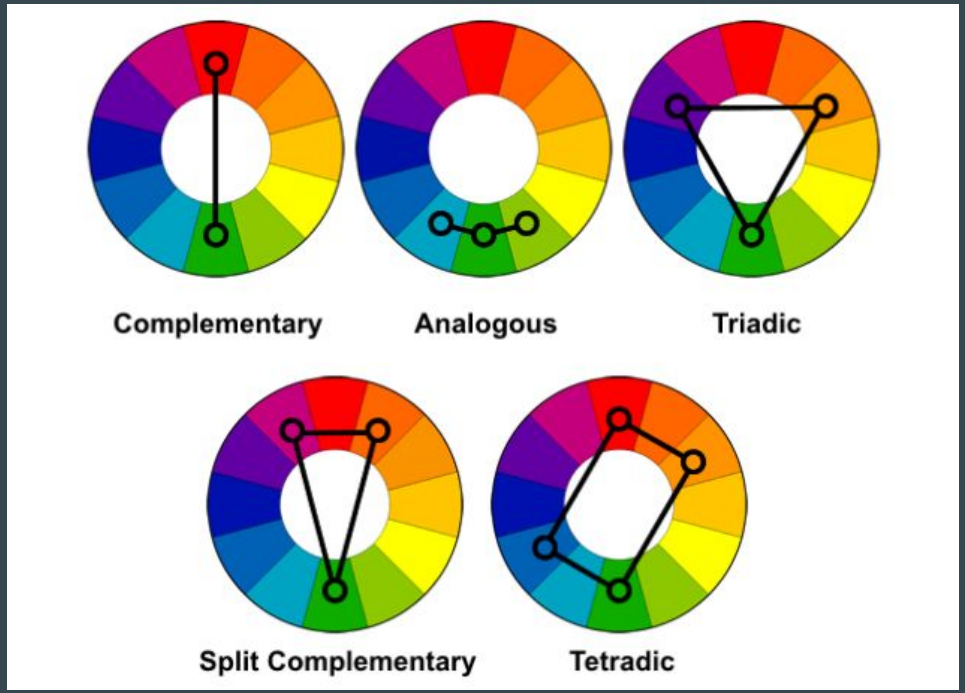
4. User Friendliness



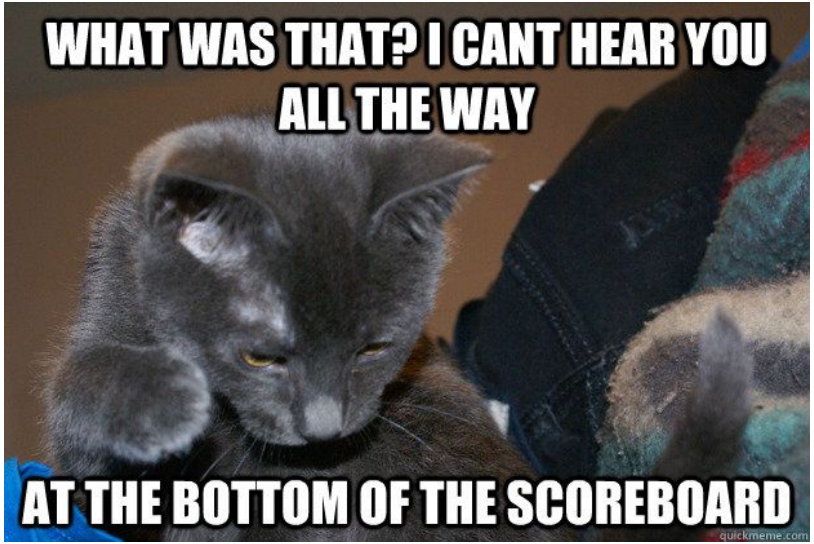


5. Colours


THINGS
AREN'T ALWAYS
#000000
AND
#FFFFFF

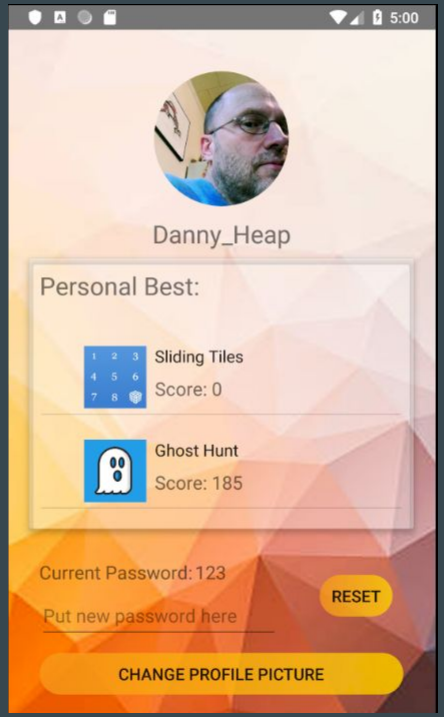

Complementary
Analogous
Triadic
Split Complementary
Tetradic

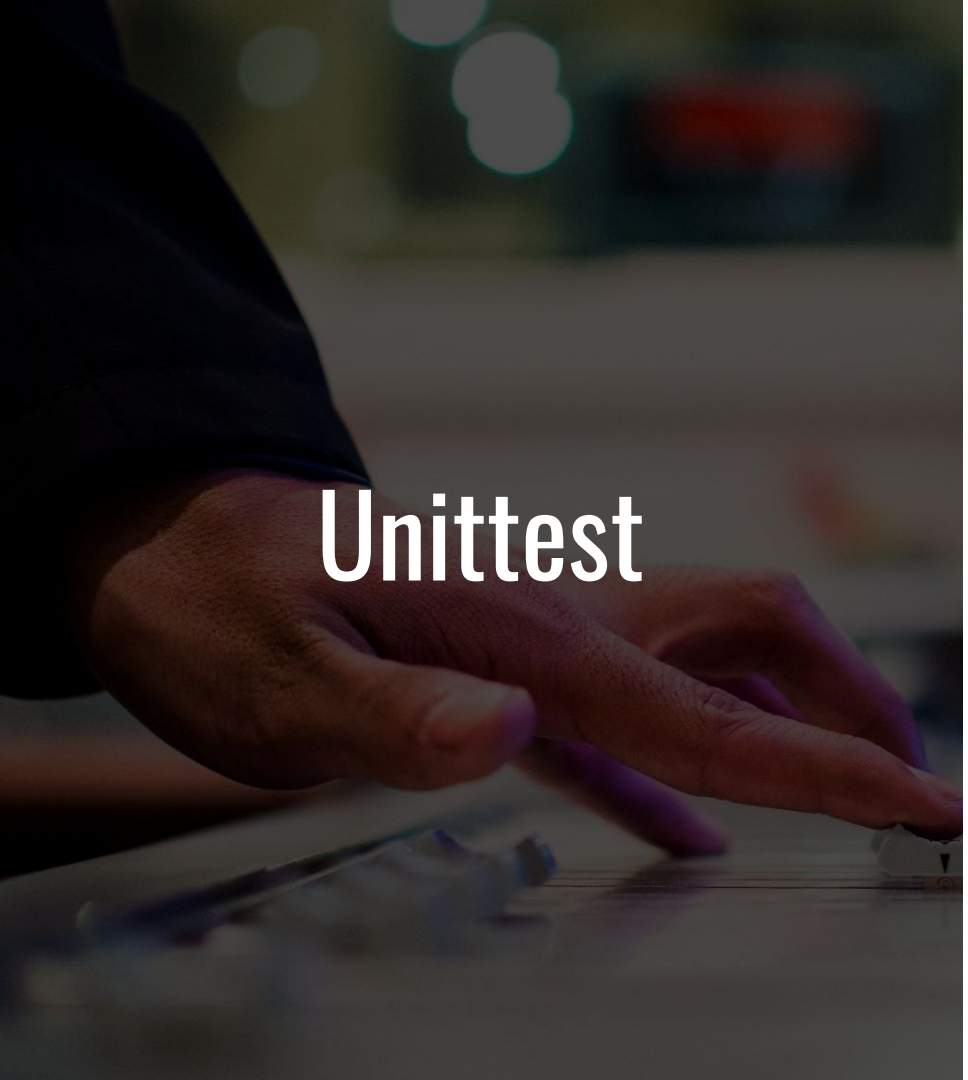
Scoreboard


WHAT WAS THAT? I CANT HEAR YOU ALL THE WAY
AT THE BOTTOM OF THE SCOREBOARD
quickmeme.com

Per-User

Per-Game





Unittest

- 100% coverage
- 0% coverage

Implemented Functionalities

1. Requirements

- 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

2. 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 score board 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

More Features

- 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


END OF PRESENTATION
THANK YOU!
memegenerator.net


Q&A