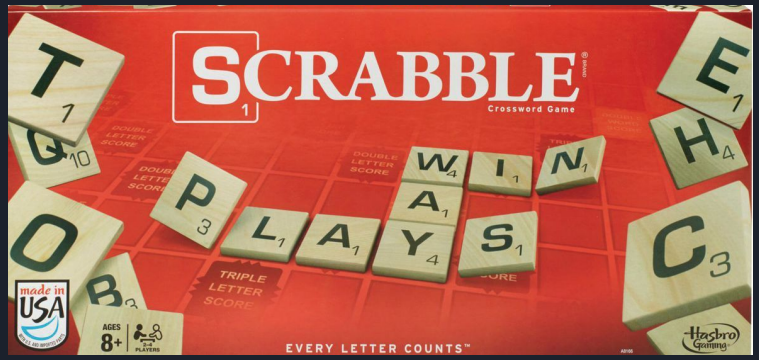
Scrabble

By: Davit, Victor, Jazli, Umair, Clark, Francisco, and Claire

Overview of Scrabble



Domain: Gaming Platform

Specification: Scrabble game imitation involving 2 players, data persistence, Java Swing UI, a score calculator, and validity checkers.

Scrabble is a skill-based board game traditionally played using wooden pieces that involves creating words. The longer the word and the more complex the word, the more points players achieve.

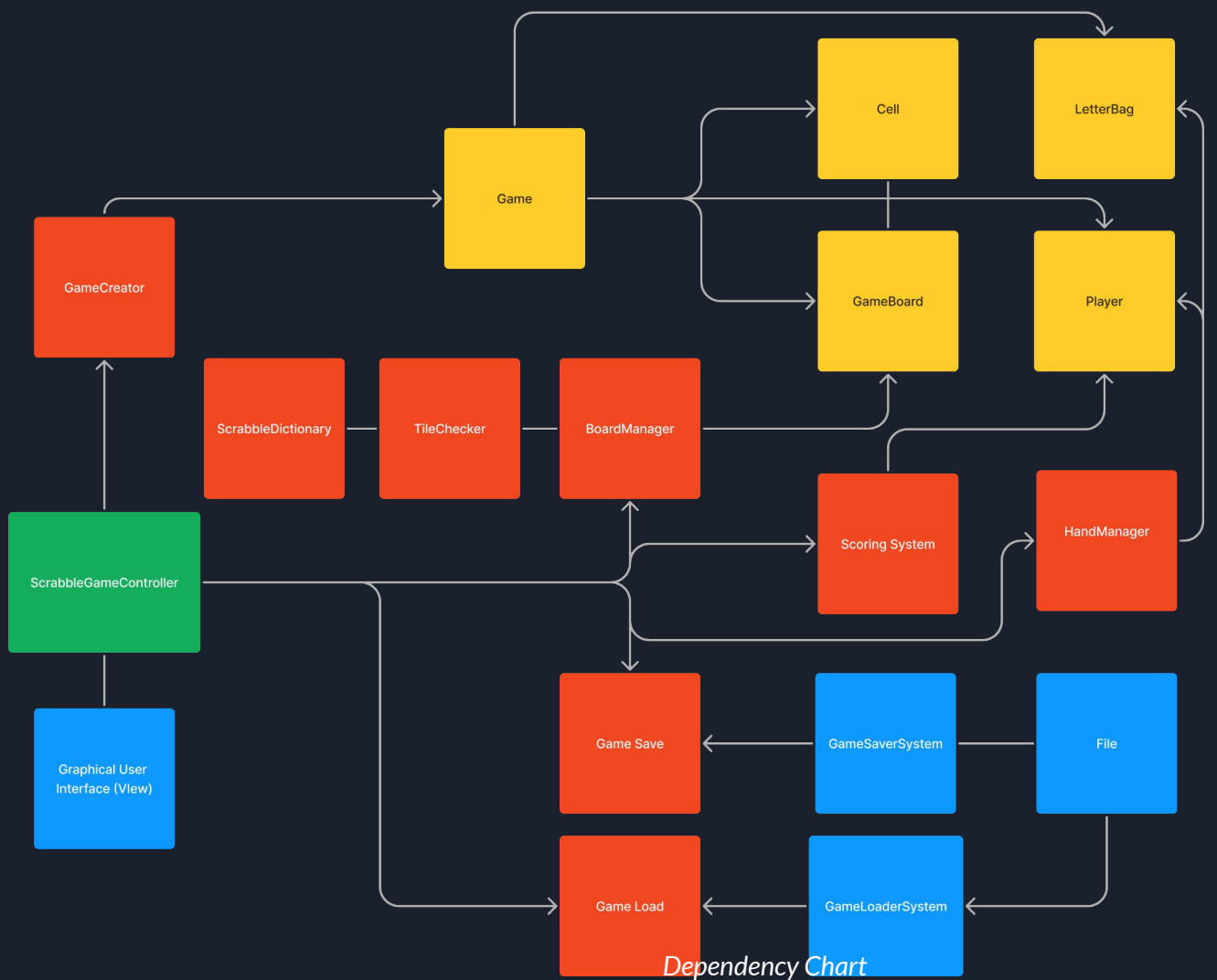
Repo: https://github.com/CSC207-2022F-UofT/course-project-scrabble

Project Specifications

- Interactive game board and tiles
  - Gameplay follows the rules of Scrabble (e.g. each player has 7 tiles , words must be formed in specific ways)
  - The UI should respond to a user's inputs (e.g. when the user clicks on buttons, tiles)

- Automatically checks the correctness of placed words and calculates the score
  - Checks words against a dictionary
  - Scores words by adding the value of each tile and applying the correct multipliers

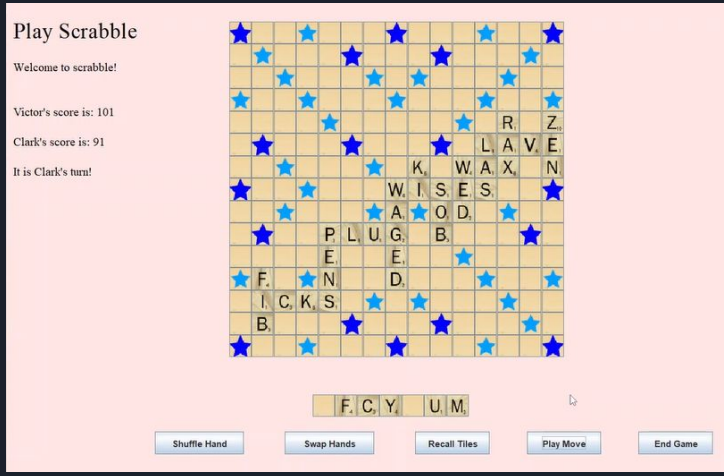
Clean Architecture Dependency Chart

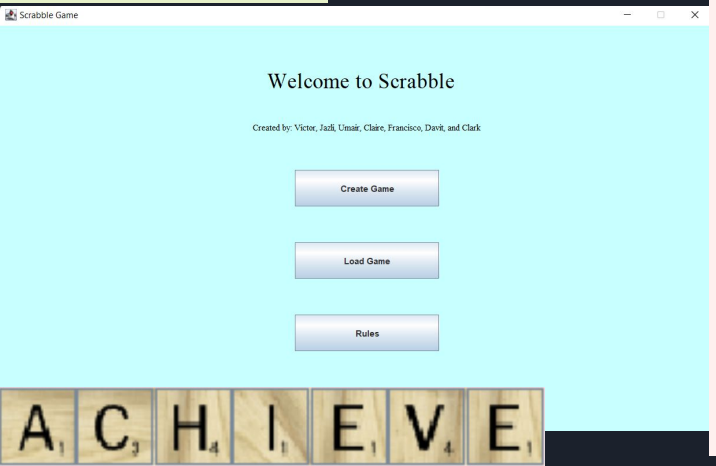


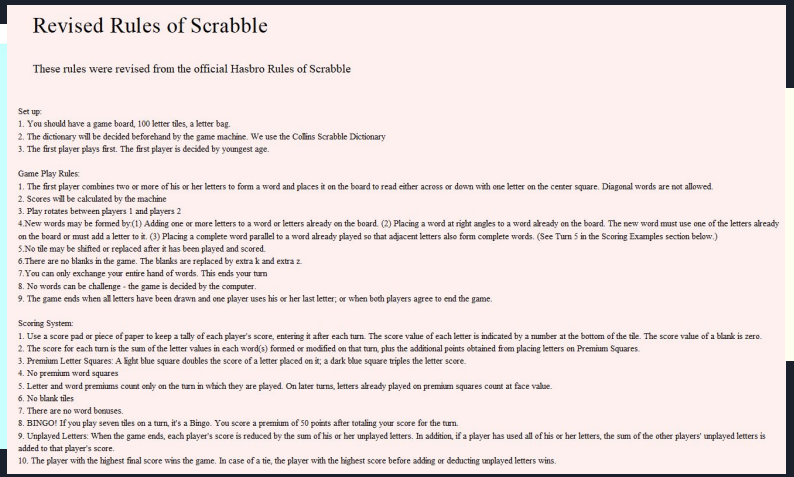
Dependency Chart

Graphical User Interface

- UI consists of pages with interactions
- Each scrabble tile is represented by a button
- Know where each button is and which coordinate they relate to.

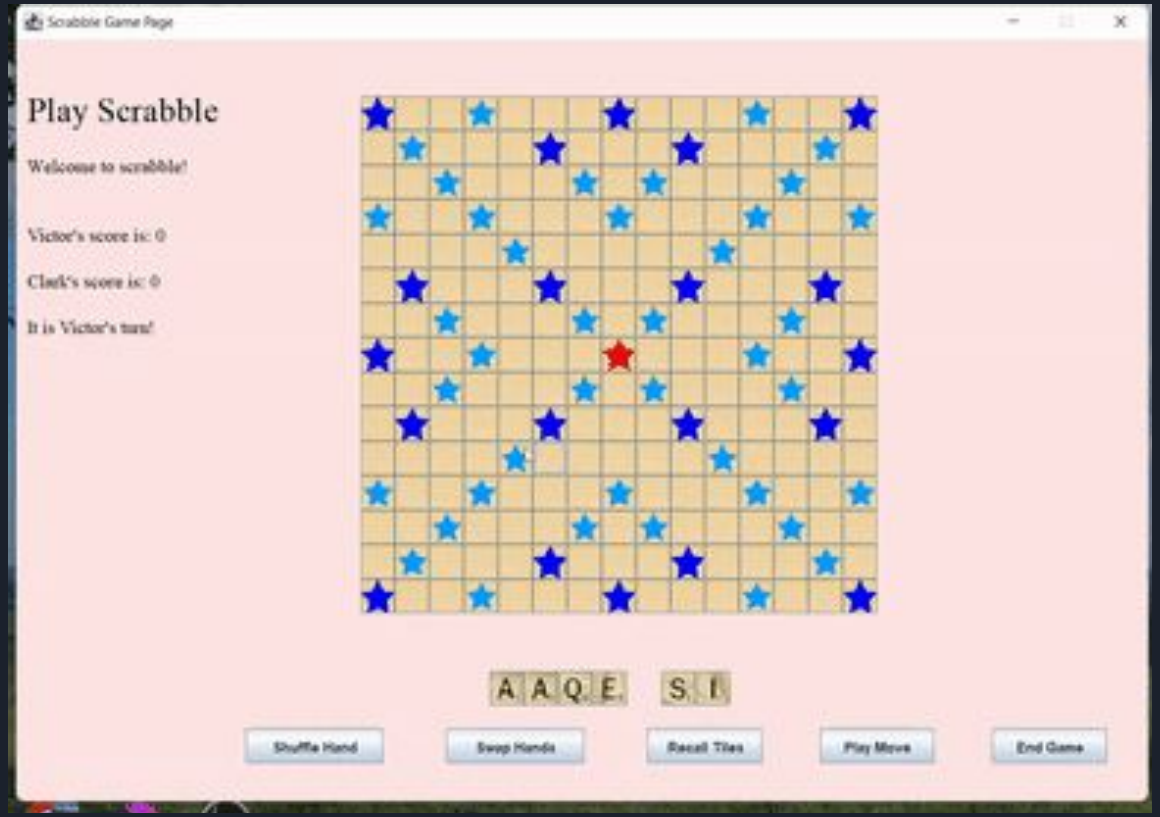






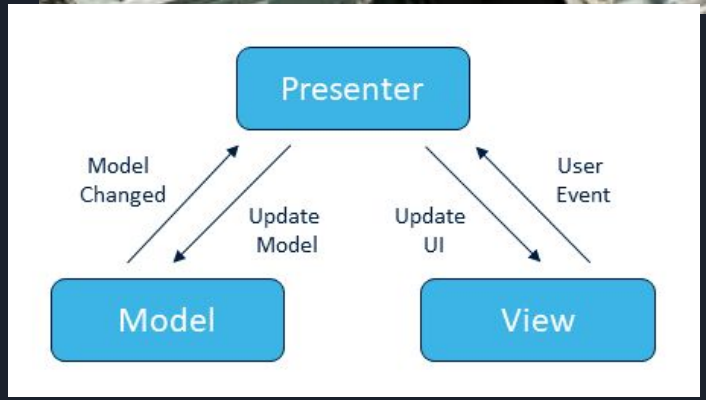
Graphical User Interface

Biggest challenge: coordinate for where tiles would be clicked while following Clean Architecture



Controller: ScrabbleGameController

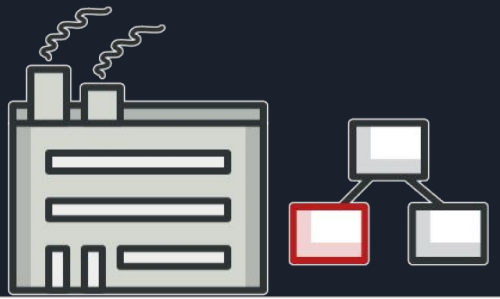
- Works as an intermediary between the View and the Usecases
- Sits in the controller layer of clean architecture
- Follows MVP architecture



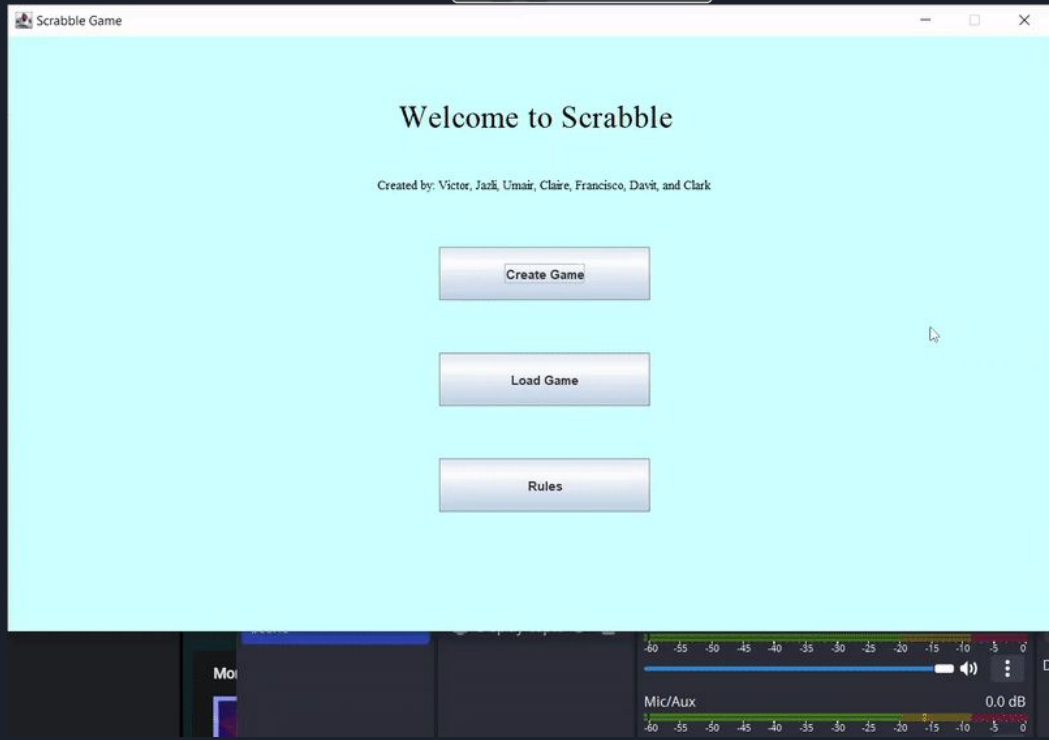
Challenges Faced

- Difficult to separate the functionality of the usecase from the controller and vice versa
- Breaking down large usecases into sub-usecases
- Leveraging interfaces to separate layers effectively
- Makes use of the facades provided by the usecases

Use Cases: Create Game



- User inputs names for Player 1 and Player 2
- Within use case layer of Clean Architecture
- Then presses “Start Game”
- Initializes a brand new game state
- Design Pattern: Factory Method



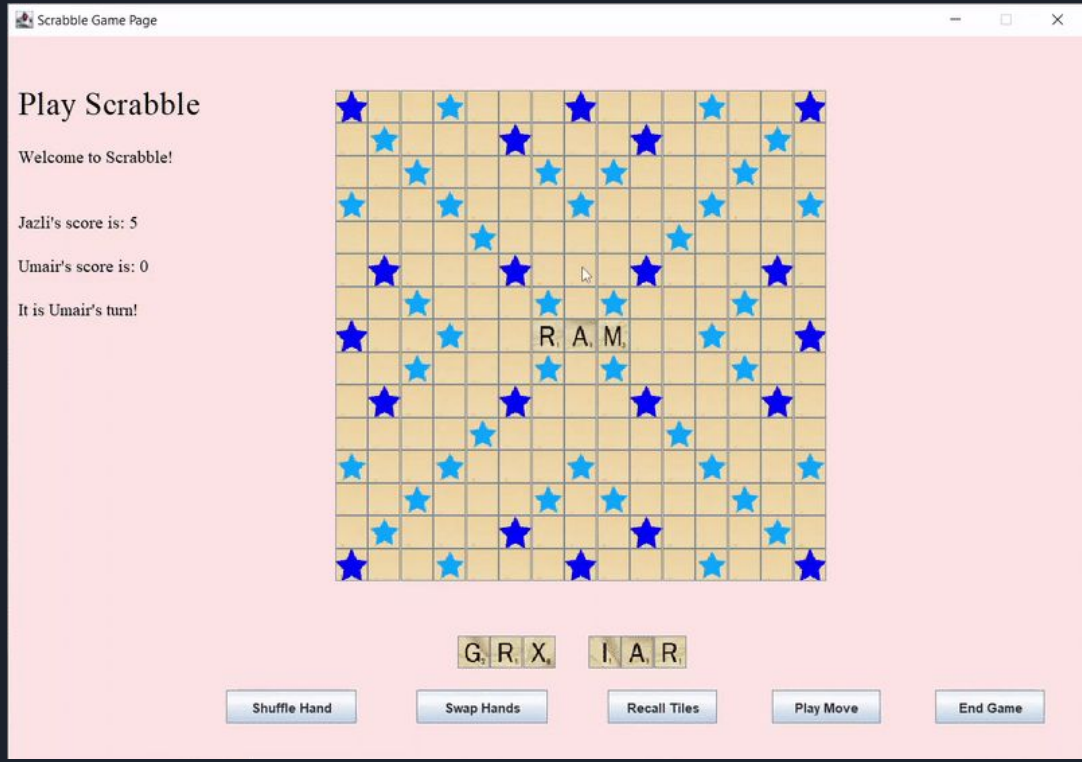
Demonstration: Create Game

Data Gateways: Save Game and Load Game

Save Game **Load Game**

- User presses “Play Move” button
- Calls save game to save game state to file
- Design Pattern: Singleton

- User presses “Load Game” button
- Loading the same game state from file
- Design Pattern: Memento



Demonstration: Save Game & Load Game

Use Cases: Board Manager

Triggered when user places tiles/words on the board or recalls the tiles.

Updates the Game Board entity with the inputs received from the controller class.

Within the Use Case layer of Clean Architecture.

Implement smaller use case methods:

- Place Tile
- Place Word
- Reset Move

Demonstration: Place Tile & Place Word

Design Patterns & SOLID

Design Pattern: Mediator

SOLID:

- Single Responsibility is to update the board entity.
- Adheres to Open/Closed Principle.
- (ISP) No unnecessary Interface methods
- (DIP) Depends on Interfaces

Demonstration: Reset Move

Testing & Challenges Faced

Tests for methods:

- checkLetterTest
- checkWordTest
- resetMovesTest

Both Happy and Unhappy flow

Challenges:

- Adding new small use cases
- Unique conditions for smaller use cases
- Bugs with Reset Move use case

```
BoardManagerTest > checkLetterTestOverlap PASSED
BoardManagerTest > checkWordTestUnhappy PASSED
BoardManagerTest > checkLetterTestValid PASSED
BoardManagerTest > checkWordTestHappy PASSED
BoardManagerTest > resetMovesTest PASSED
```

Use Case: Tile Checker

Validating a set of inputs (lists of coordinates) by checking for adjacency and generating a list of words that needs to be scored.

Design Idea

- SOLID
  - Single responsibility is to validate a set of coordinates.
  - Since there aren't any subclasses, it adheres to the LSP by default.
  - Depend on Scrabble Dictionary.

- Tile Checker is part of the Use Case layer of clean architecture.

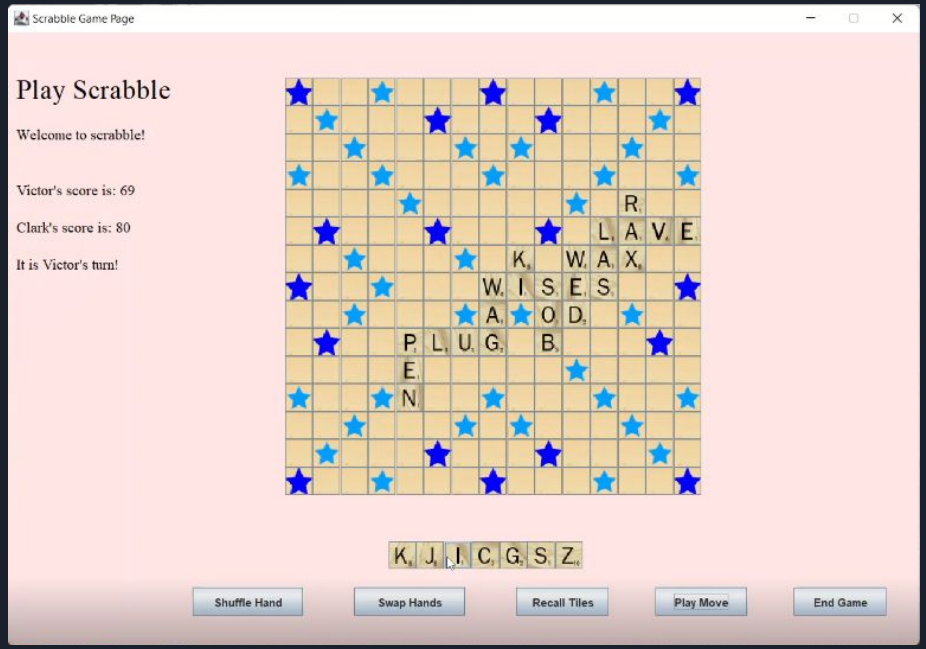
Problem Faced

Corner cases:

- Tiles on the edge of the board
- Multiple words formed
- Using coordinates instead of words

Solution

- Added test cases
- Using helper functions



Use Cases: Scrabble Dictionary

Functionality

- Constructs a searchable list from a text file
- Determines the validity of entered words
- Locates words on the board using tile coordinates

Design

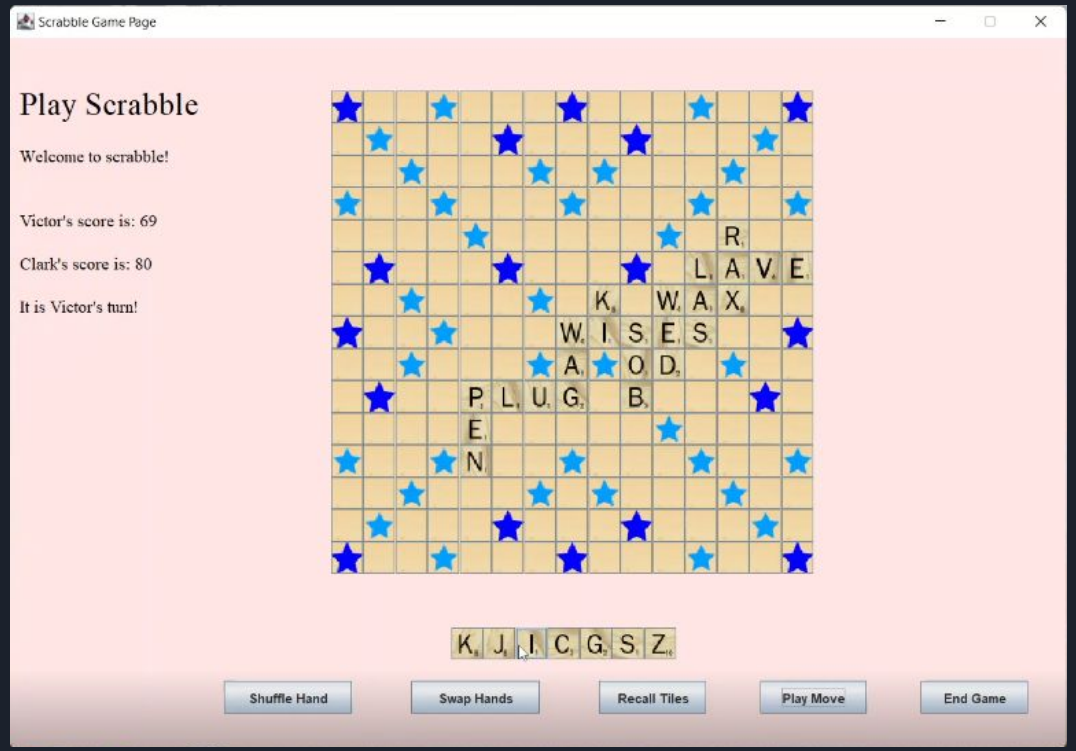
- SOLID: no interfaces or subclasses
- Clean Architecture: Use case
- Singleton Pattern

Challenges

- Static methods
- Multiple functions
- File handling

Use Case: Draw Tile

- Triggered at end of turn
- Fills Player's hand from bag
- Within use case layer of Clean Architecture



Design Principles

Design Pattern:

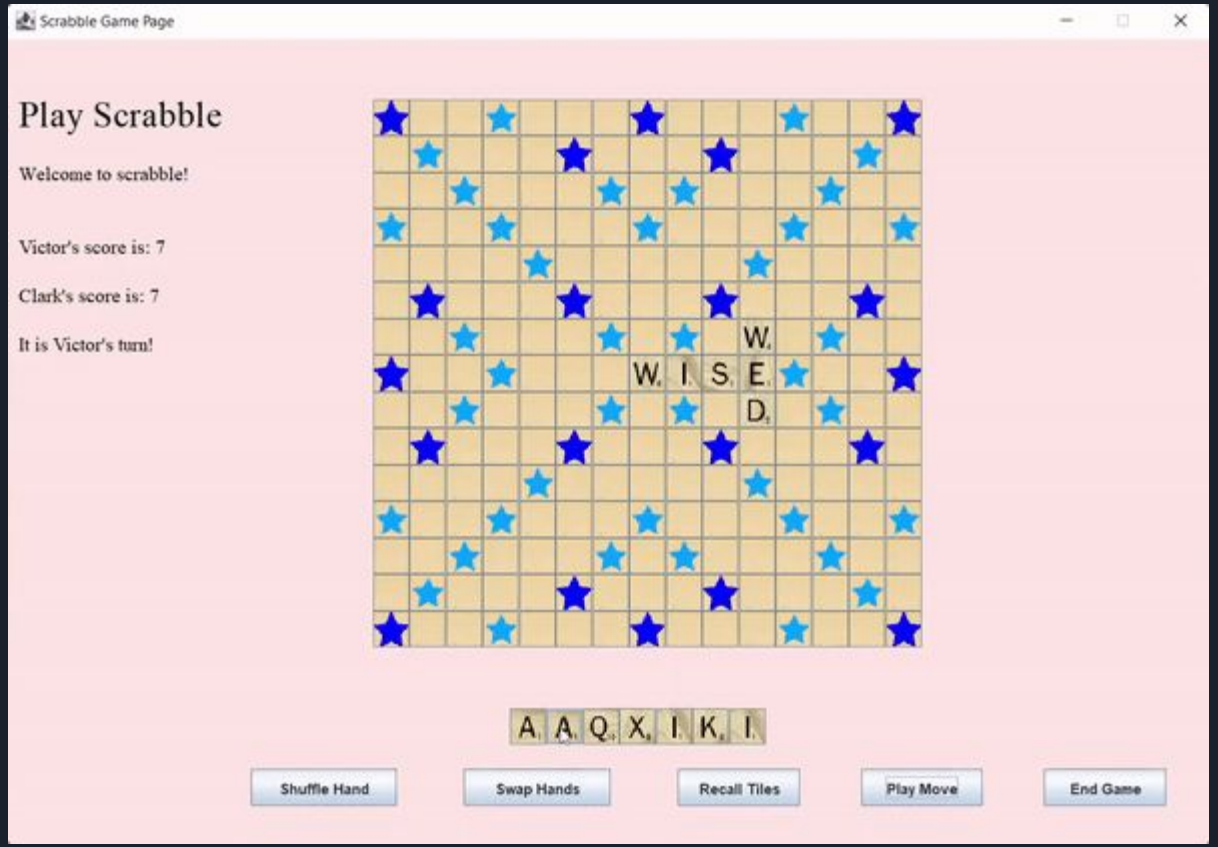
- Mediator
- Reduce Dependencies

SOLID:

- Single role to draw tiles
- Open-Close Principle
- Not dependant on other classes

Use Case: Scoring System

- Triggered after valid move is played
- Scores move depending on positioning on board
- Within use case layer of Clean Architecture



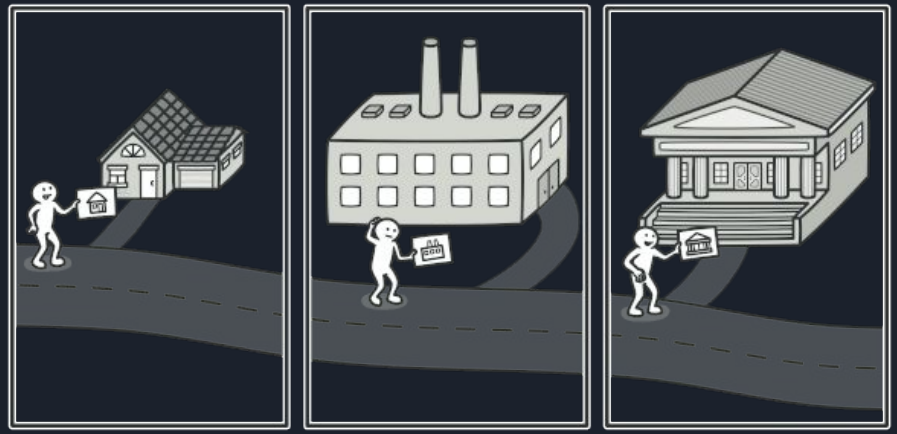
Design Principles

Design Pattern

- Visitor
- Separate algorithm from object operated on
- Easier to adapt

SOLID

- Sole purpose to score valid moves
- Interfaces adequately split
- No dependency on other classes



Challenge Faced

Problem

- Discrete bugs within Draw Tile from inadequate testing

Solution

- Increased unhappy flow testing from milestone 4
- Resolved many errors throughout the program with better testing

Improvements/Future Work

- AI
- Multiple Players
- Online/Web Based Tournament


THE
END