

All of our classes but one had no changes made to the uml and data structure. The only one is scrabblegame. It was replaced by 4 new classes( ScrabbleView, ScrabbleModel, ScrabbleGUI and ScrabbleController) as our game changed from a text-based version to a GUI-based version, following the MVC pattern.

The scrabbleGame class has been refactored into scrabbleModel to implement MVC. Now, the ScrabbleModel class makes use of an ArrayList to store 2 to 4 players in order to rotate between them. We also make use of an index and modulus to track the current turn and cycle through players. The ArrayList gives  $O(1)$  access to get the current player and also a simple iteration to display the scores. The ScrabbleModel class also makes use of a new ArrayList to store all registered views (observers). The model calls `notifyViews()` to update each view each time the state of the game changes. We chose ArrayList because there are usually only 1–2 views, which gives  $O(1)$  access to any view and makes it easy to iterate over them for notifications. This class now contains all game logic such as `playerList`, `gameBoard`, `bagOfTiles`, `gameDictionary`. We removed the `Scanner userInput` field as input is now handled by the Controller.

ScrabbleView is a new class that has the `update()` method that all views must implement.

ScrabbleGUI (implements ScrabbleView) is a new class that uses 2D arrays for visual components. The `boardButtons` is a `JButton[][]` array to represent the 15x15 grid, allowing any board cell to update in  $O(1)$ . The `handButtons` is a `JButton[]` array of size 7 that represents the 7 tiles of a player. It also is  $O(1)$ . Arrays were chosen over ArrayLists because the sizes are fixed. It contains all GUI components (JFrame, JPanels, JButtons, JLabels) It is the class that does the visual representation. ScrabbleGUI is linked to the model and controller so it can update the visual representation of the game when the state changes and send the user's input back to the controller.

ScrabbleController (implements ActionListener) is a new class that controls interactions between the GUI (view updates) and Model (game state). It handles all the user actions with the `actionPerformed()` method.

## **New relationships**

Dependency: Main creates ScrabbleModel, ScrabbleGUI, and ScrabbleController

Implements: ScrabbleGUI implements ScrabbleView interface, ScrabbleController implements ActionListener interface

Composition: ScrabbleModel contains a List of ScrabbleView objects

Association: ScrabbleGUI observes ScrabbleModel and interacts with ScrabbleController, ScrabbleController controls ScrabbleModel and updates ScrabbleGUI

Previous classes that had relationships with scrabbleGame have the same relationship with ScrabbleModel