Although my general software design follows many common design scenarios, including using interfaces for “play” tile and for “special” tile types, and delegation (such as for coordinates), the real difference in my design is how I interpret the player adding words.

First of all, I already conferred with Charlie Garrod about this, and he says this is fine as an implementation of game mechanics.

For visualization of the following, see “homework/4/designs/designs.pdf.
How it works is, instead of the player adding tiles one by one to the board, they *type* their word into a given text box area on bottom, and my code reads the word they type (an equivalent of isValid()) and finds all possible ways the player can add the word to the board. The function reads in an arraylist of LetterTiles and returns an ArrayList of ArrayLists of Coords => ArrayList<ArrayList<Coord>> which represents all the locations the letters can be assigned. The user would then have the ability to choose one of the highlighted locations the word can go, or, if “hints” are turned off and the found locations are not visible to the player, the player would have essentially typed the word they want to insert, then clicked on a location (non-highlighted) and the game would auto-fill in the player’s word. If the location clicked has two possible directions (right or down) an option would be presented to choose up or down.

As for the rest of the code, I have special tiles that implement the SpecialTile interface which stores their locations, icons and a function that calls the defining property of each special tile. The letter and blank tiles implement Tile and store location, icon, character (either a letter, or “_” for blank tile), and score value (0-10). After much planning, I decided not to unite special tiles and letter tiles under one interface “Tile” because of the vastly different purposes and accessibility of the two types of tiles. Letter tiles are important to word-validation algorithms and displaying in the GUI and are randomly given to players. Special tiles are purchased, hidden and don’t have game value unless a tile is placed on top of it. So I made them separate groups of classes.

I made a “Square” class to replace the idea of using one or more two-dimensional arrays that each store separate things such as score modifiers or letter tiles in play. A Square class can be represented in a 2D array in Board.java, and each Square can keep track of everything on that square. It would have an owner for each SpecialTile to help the GUI display them correctly. Another implementation I’m considering is only having the active player’s class be in charge of keeping track of which SpecialTiles it has used, and only display it’s own.

I will be using the Observer Pattern to implement the various actions the special tiles can do. A simple Square.activateSpecial() function can call SpecialTile.activate() which will then do its thing. It would take in a parameter of the Board class (activate(Board board)) because of the varying tasks they do, such as changing the items in Squares to changing player scores in the current move. Therefore, Board (and maybe possibly Game.jaav if needed in the future) would be passed in as a parameter for maximum efficiency in modification of the current round.