SYSC3110 Group Project

Milestone 3: Group 40
Change Log / Explanation of Design Decisions

Group Members
Ahmad Alkawasmeh - 101108706
James Grieder - 101177911
Daniel Kuchanski - 101182041
Ibtasam Rasool - 101186050

Decision 1: MenuController

We wanted the menu controller to be separate from any controllers that are used to control the normal course of gameplay. From the player perspective, they will generally only use the menu at the beginning or end of a game, or in exceptional circumstances for undo/redo. The menu options are not meant to be a core part of the gameplay.

Separating the implementation of the controller for the menu means that as the features of the game continue to grow, there is a clear division between what methods/input options are related to normal gameplay, and what are related to setting up/closing the game, or reverting actions in the game that do not happen often. Once implemented, the undo/redo features will be relatively fixed (unless there is a complete overhaul to the game logic), just as the menu options will be. From the programmer perspective, this ensures that future updates to the core game logic will be done in the GameController and TrayController classes, increasing cohesion.

Decision 2: AI Helper Class

For implementing AI, we decided to use a separate AIHelper class. A subclass was not needed, as there was no need to add any additional fields to the player class. An interface was unnecessary, as it would be a redundant addition that would only require the addition of another class. For these reasons, the AIHelper class was designed with delegation in mind. The original implementation was a series of AI methods in the Board class, Player class, and Game class. These methods were moved to the newly created AIHelper class to improve cohesion.

Decision 3: AI Input Format

We maintained the same input format that the user previously would use to add a word to the board on the text based interface (i.e. WORD H8 to place WORD at the coordinate H8). The AI methods are focused on searching the dictionary, the player tray, and the letters on the board for a suitable move, and then constructing the exact

same input that a user player would have constructed for the text interface. Doing this increases code reuse, specifically for the getInput() method that is used to take a command from the user, and also reduces coupling between the AIHelper class and the Board, Player and Game classes.

Decision 4: Placing Words

Placing words on the board is handled through a word placement basis instead of on a tile placement basis. This allows for an entire word to be checked and kept track of, meaning each turn is more easily defined. This also allows for the word legality checks to incorporate all the positions of the word.

Decision 5: Handling touching words

Touching words are handled by taking the word that is placed first, and checking the values that intersect it (in the opposite direction). When a word touches another word, a word by word comparison occurs as opposed to only comparing touching board letter values. This decision will allow for an easier implementation of the undo and redo functions in future iterations of the game.

Decision 6: Placing values on the center of the Board

Through a breadth first search all letter values on the board can be traced back to the center square. If that is not possible, then the check fails and the word is not deemed to be valid and will therefore not be placed.

Decision 7: MVC

The MVC design pattern allows for the encapsulation of an extensive list of data that the view needs to update itself; this allows for any game action to be processed in the model and have the appropriate view information transported to the view this allows for high cohesion and loose coupling between the model and the view. Specifically, we used a GameEvent object to handle the large amount of information that must be passed between classes during each turn of the game.

Decision 8: Blank tile value handling

The blank tile is given its' value by first prompting the player with a *JOptionPane,* once the tile has been selected and placed on the board. Closing or canceling this action results in the placement being canceled, and the tile returning to the Player's tray.

Possible Improvements

- Several of our classes could be refactored due to being quite large
    - GameEvent class, Game class

- Some methods contain repetitive and/or lengthy code that could be improved upon:
    - checkVertical, checkHorizontal, getPossibleWordPosition, isWordConnectedToCenter, placeWord

- Need to add implementations for blank square test methods

- There are some adjustments to names of methods

Note: these are potential improvements to the code. Actual issues with game functionality are mentioned in the “Known Issues” section of the readme.txt file.