SYSC 3110 Milestone 4 Documentation

Group 17:
Oyindamola Taiwo-Olupeka - 101155729
Oluwatomisin Ajayi - 101189490
Edidiong Okon - 101204818
Ese Iyamu - 101081699

TABLE OF CONTENT

Section	Content	Completed By
A	Design Decisions	Oyindamola Taiwo-Olupeka
B	UML diagrams	Oluwatomisin Ajayi
C	User Manual	Ese Iyamu
D	ReadMe	Edidiong Okon

A. Design Decisions

For the ScrabbleGame project, we collectively decided on nine(9) classes and one(1) interface- ScrabbleModel, ScrabbleBoardFrame, ScrabbleBoard, TileBag, Player, AIPlayer, ScrabbleController, ScrabbleStart, MainFrame are classes and ScrabbleView is an interface. We implemented the MVC pattern in this milestone. In addition to these classes, we also have the ScrabbleModelTest which tests that our game works.
The model was the contents of the game, and it notifies the view when any change has been made. The is the graphical representation of the model, it displays the look of the content and subscribes to the model to ensure that its appearance reflects the current state of the board model. Lastly, the controller handles the input, therefore setting the behaviour of the input. It updates the model according to the input from the user and the view observes the updates.
This is subject to change as the project progresses.
In a traditional Scrabble game, 2-4 players randomly choose 7 tiles to place on their tile rack. During their turn, they pick a combination of tiles and play on the board. Whatever they decide to play must be a legal word in the ScrabbleWords.txt file which is read by a buffered reader.

Breakdown of classes:

1. **ScrabbleModel:**
   This class initializes the scrabble game. It generally controls the entire flow of the game by distributing 7 random Tiles to each player, verifying the legality of moves and holding all vital information. It keeps track of the 2-4 players (specifically, the order they play in, their score, and the start and end of their turns). And lastly, it determines the winner(s) at the end of the game by comparing all the players' scores and picking the player(s) with the highest score.
   The play method asks for the number of players and then enters a loop that plays words. The game loops through the players until a player has no more tiles and the tile bag is empty, and the game is over. ScrabbleGame keeps track of each player's score and determines the winner by whose score is the highest.
   It makes use of ScrabbleWords.txt file with a buffered Reader and verifies the legality of every played word in the Scrabble game.

2. **ScrabbleBoard:**
   This class initializes the scrabble board. It stores the default scrabble board which includes all the bonuses (double letter score, triple letter score, double word score and triple word score). The board is a 15x15 array of characters.

3. **TileBag:**
   This class is the representation of the tile bag used in a scrabble game. The tile bag is an array of characters and it stores the instances of every character in the game. For example, the character ‘A’ occurs 9 times in the bag, ‘Z’ occurs 1 time in the bag, and there are 2 blank tiles in the bag. The TileBag class also uses a Map to set each character with their point value. Playing ‘A’ is 1 point, ‘Z’ is 10 points and a blank tile is 0 always.

4. **ScrabbleBoardFrame:**
   This class initializes a scrabble board frame and is in charge of the Graphical User Interface (GUI)of the board.
   This class has 4 nested classes. We decided to use nested panel classes to make our work organized.
   These classes are,

- BoardPanel - which initializes the board which holds the grid layout for the traditional scrabble board.
- RackPanel- rack panel that represents the rack for every player. With each turn, the label on the panel changes per player move.
- ScorePanel- Store the player's name and their scores. It also has information about the values associated with the tiles for user understanding.
- ButtonPanel- initializes all the buttons for the game. Such as play, clear, pass, and end game.

5. **Player:**
   This class represents the players in the ScrabbleGame. It holds the name, score, and tiles It is a dependent class that provides information for the model and frame.

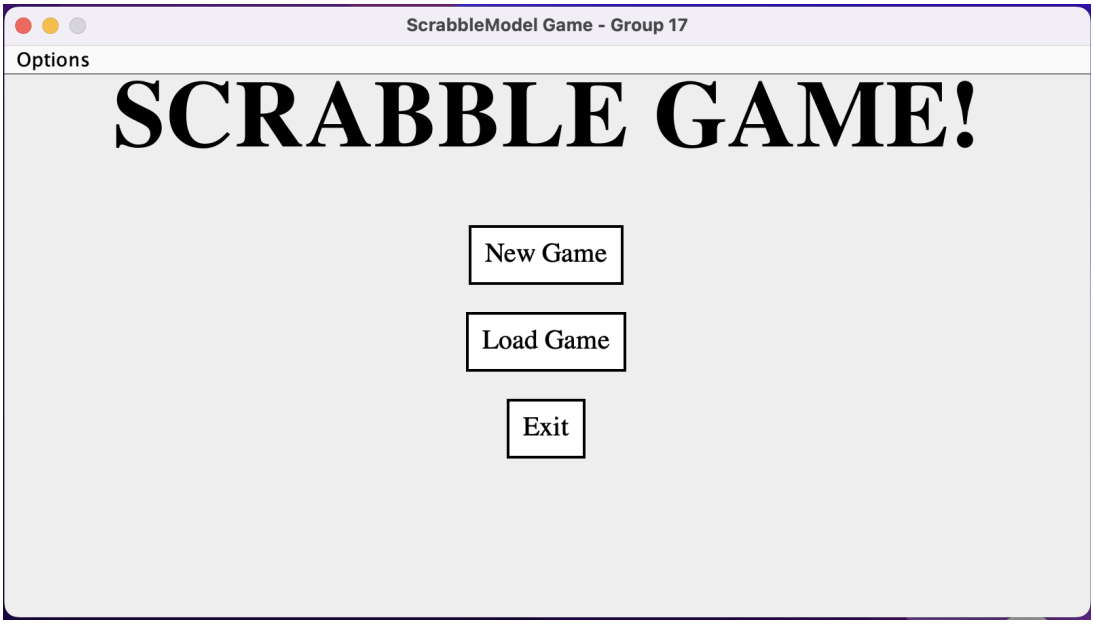
6. **AIPlayer:**
   This class initializes an AI player that extends the Player class. This means that the computer player can make moves just like a human player during their own turn. Ideally, it would want to make a move that would result in the highest score.

7. **ScrabbleStart:**
   Starting screen of the game. Asks players to input their names and initializes some buttons. Only 4 players (Human or AI players) maximum can be in the game.

8. **ScrabbleController:**
   Observes the user inputs using an action listener and mouse listener and performs the actions based on user input.

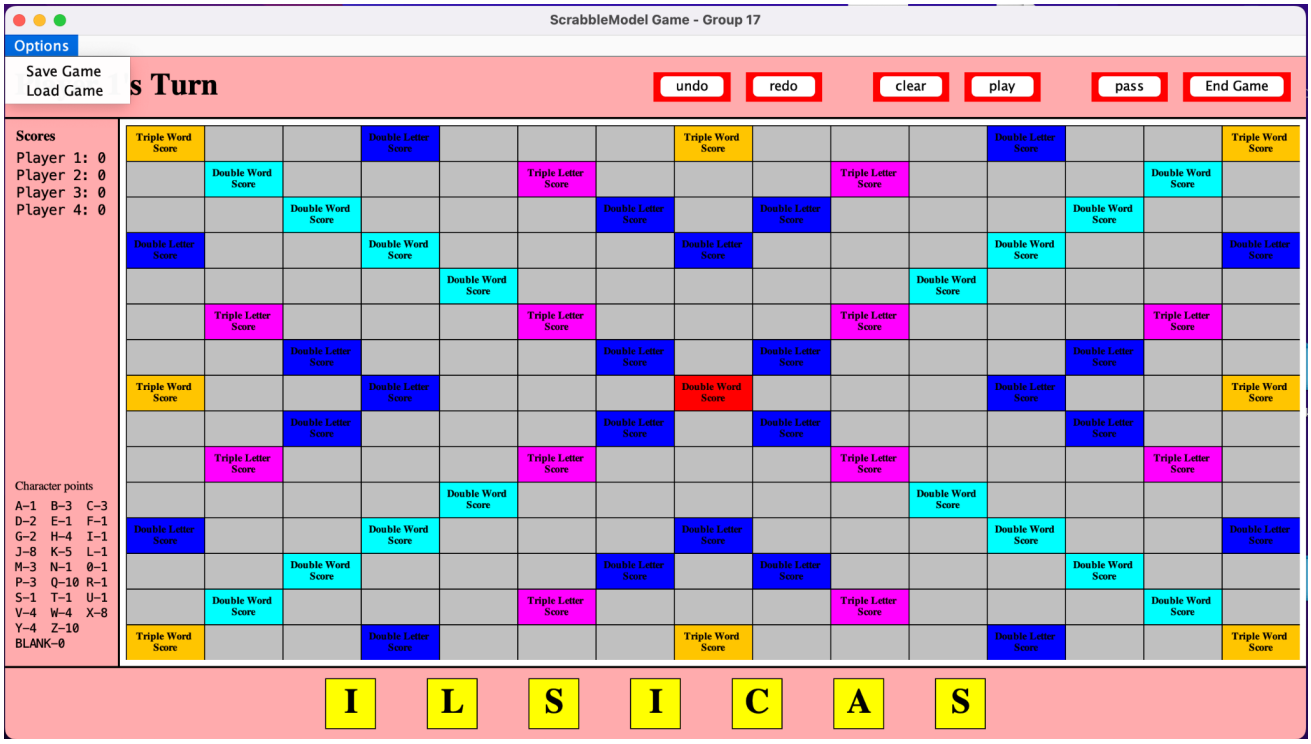
9. **ScrabbleView:**
   Manages the views of the game.

10. **MainFrame:**
    The main class that starts the game. It calls ScrabbleStart first for stylistic choices.



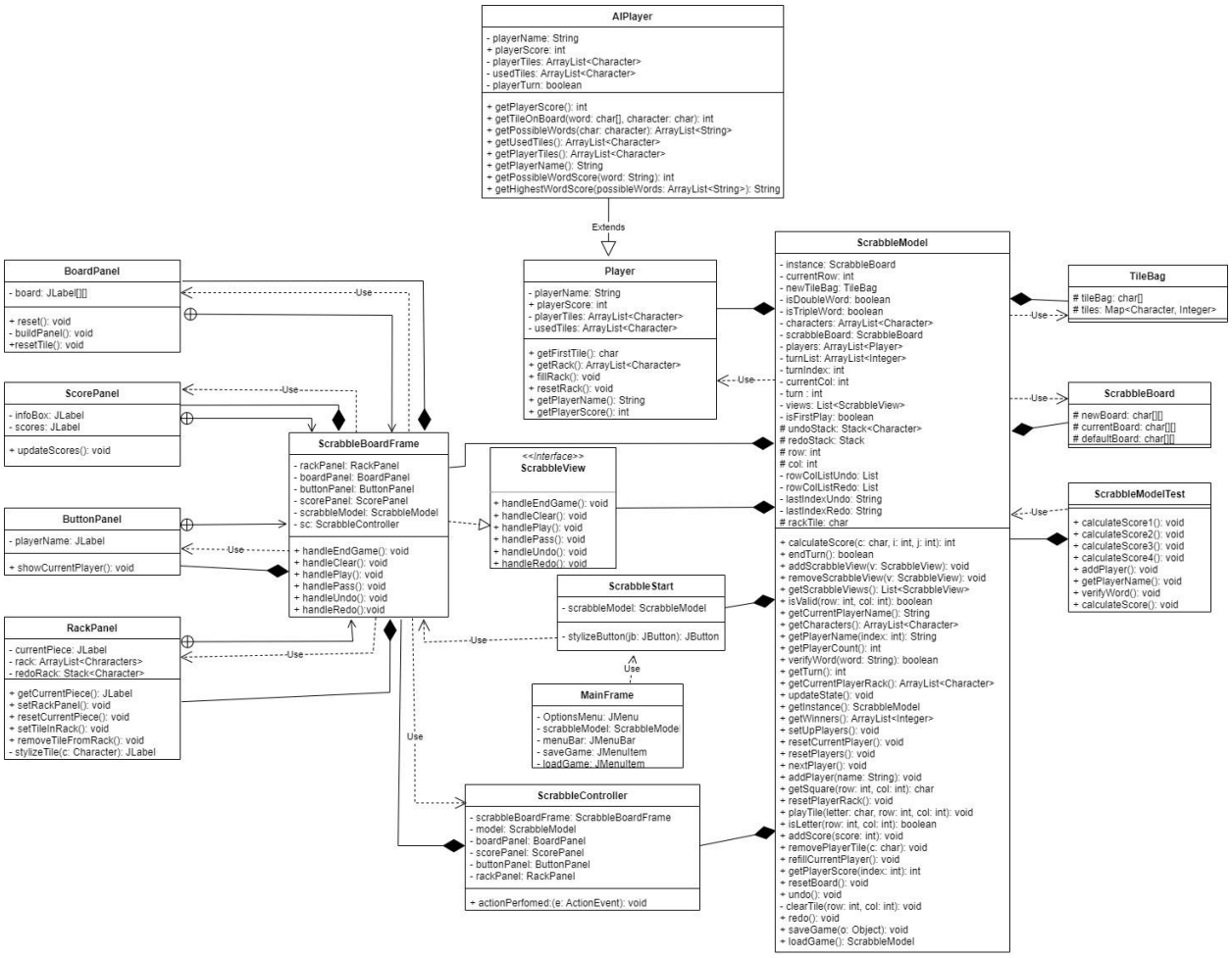
11. **ScrabbleModelTest:**
    This is a class that tests our scoreCalculated method. Which is a major factor we consider to be a success factor to prove whether our game is functional or not. It imports the Junit unit testing framework.

To conclude, the classes have individual responsibilities but work together to make our Scrabble Game functional. All classes are subject to change at each milestone.
This is currently how our game looks:

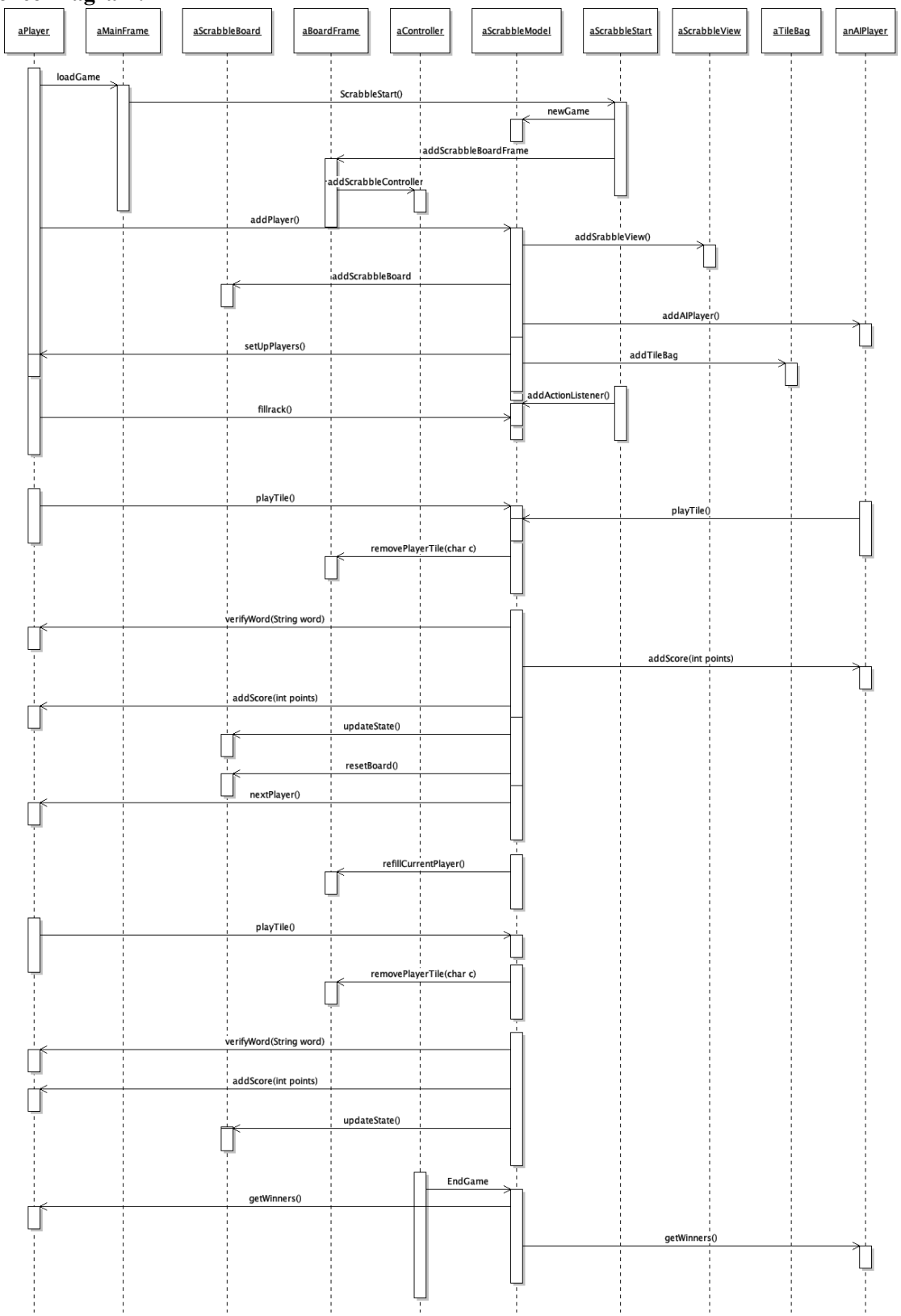


B. UML Diagrams

Class Diagram:



Sequence Diagram:

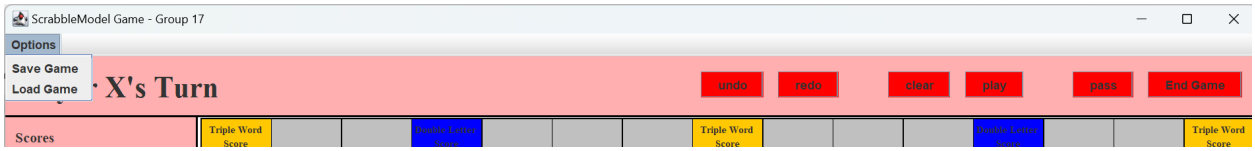


- Games loads.
- Players are added to the game.
- Tile rack is created and a scrabble board is displayed.
- Players start playing. Once a tile is played, it is removed from the rack.
- A winner emerges based on the final score.
- Game ends.

C. User Manual

To Play the board game Scrabble:

- Click on the jar file to start the game.
- Enter the names of the players. If no names are entered players would be assigned names.
- Click start game
- Players are able to use the following buttons for gameplay: - “clear”, “play”, “ pass”, “End Game”, “undo”, and “redo”.



Undo:- This button removes the most recent tiles played by the player from the board (e.g a player plays “BREED” if they click the undo button the tiles D would be taken out. If they click on the undo button again, it would take out the E tile next

Redo:- This button puts the player's most recent tile back on the board (e.g from the example given for the undo button, if the player presses the redo button, it would place back the most recently removed tile back on the board, in this example that would be the letter tile E. pressed once again, the letter tile D would be put back on the board).

Clear:- This button is used to clear the letter tiles placed by the player on the board. This button can only be used before Play is pressed.

Play: - This button is used to “play” the letters placed by the Player. Once this button has been pressed the players' played letters are locked and cannot be changed.

Pass: - This button is used by the players to pass their turn to the next player.

End Game: - This button is used to end the game after every player in the game has had at least 1 turn.

Save Game:- This menu item enables players to save the current progress of the game.

Load Game:- This menu item enables players to load a previously saved game and continue from where they stopped.

- The first player starts by clicking the letter they want on the board from their rack and then clicking on the space* they want to place the tile on the board.

- Players are allowed to place their next letters only next to letters tiles already on the board. Words can be formed horizontally or vertically only from existing letters.
- Player's score is calculated and given back as many tiles as they used to play to replace their rack. E.g say a player's rack was [A, F, H, A, G, T, U] and they played letters "H, U, T". The player's rack is filled with 3 letters and their rack would be ["A, F, D, A, G, F, B].
- All new words(horizontally or vertically) formed by the auditions of new tiles must be comprehensible. Tiles can't be moved around once they've been played and scored.
- Up to two blank tiles can be used.**
- Game ends when all letters have been drawn or when players have used up all their tiles or when all possible moves have been made.

Note:
* The first player has to put their first letter in the middle of the board which is also a premium square..
** Player would have to state what letters the blank represents and it remains that letter till the end of the game.

D. ReadMe

Scrabble
A simplified version of the classic word game, Scrabble, built using Java and java.util packages.

Milestone 4
A GUI version of the Scrabble game where players use a mouse to click the tiles in the tile rack and place them on the board.

Authors
Oyindamola Taiwo-Olupeka
- Worked on the ScrabbleBoardFrame, MainFrame classes and the ScrabbleView interface.
- Updated the design decision document to reflect the refined design.
- Added Java Doc comments to classes and methods.

Oluwatomisin Ajayi
- Worked on the ScrabbleModel, ScrabbleBoard, and TileBag class
- Created the updated UML class and sequence diagrams.
- Added Java Doc comments to classes and methods.

Ese Iyamu
- Created the Player, AIPlayer and ScrabbleController class.
- Worked on the user manual aspect of the updated documentation.
- Added Java Doc comments to classes and methods.

Edidiong Okon
- Worked on the ScrabbleStart classes and the test cases.
- Wrote the README file and documentation.
- Added Java Doc comments to classes and methods.

Known Issues

- Currently the undo/redo method is functional but when we undo all the tiles played, you have to hit the clear button to reset the turn or else you get an index out-of-bounds error.
- The AI player is not fully functional. We have not been able to give the player the full functionality that a Human player would have.
- There is an issue with allowing invalid words.

Change Log

- The ScrabbleController and ScrabbleView classes were added to align with the MVC pattern.
- The Player class was refactored to work better with the model class.
- We created the AIPlayer class to represent the computer player ("ScrabbleBot").
- We implemented the observable for better MVC communication.
- We fixed the JAR file.
- We implemented Serializable for the frame, model and player classes
- We added a menu bar with menu items (Save Game and Load Game) to the MainFrame class just for better functionality.
- Also added a load button to the first card.
- We added methods and buttons for the save and load functions for the game.
- We added methods and buttons for the undo and redo functions for the game.

Roadmap

October 23, 2022,

- Text-Based version of the game played with the keyboard.

November 11, 2022.

- GUI (View and Controller).
- Unit Testing for the Model.

November 21, 2022.

- Creation of blank tiles, premium squares and AI players.
- Design of Scrabble move generation algorithm/method.

December 9, 2022.

- Addition of multiple game-level undo and redo features.
- Save and load features.
- Creation of custom boards.