UML changes explanation:

Ultimately there were minimal changes between milestone 3 and 4, the majority of the changes came from making several classes (Tile, GameModel, Player, Board and TileBag) implement the java module Serializable to implement the saving/loading functionality. Minimal changes were made to both Board, GameFrame and GameModel to handle the undo/redo functions and custom board via XML files supported. There were also minor changes to fix various bugs or glitches in the code, and ensure that the project was behaving as intended.

Data structure
Class:
Board:
- Data structures used:
  - 2D array: Tile[][]
  - Explanation:
    - A scrabble board is always a fixed size, so a 2D array is appropriate as you'd never need to modify the size. It's also ideal as it allows for O(1) access to any cell.
  - Operations:
    - placeTile(row, col, tile) updates the 2D array cell
    - getTile(row, col) retrieves the tile from a specified location
    - removeTile(row, col, tile) sets the tile back to Null
    - display() loops over the array
    - isEmpty() loops over the array
  - 2D array: Premium[][]
  - Explanation:
    - Since the board is always a fixed size, a 2D array is appropriate for listing the premium tiles.
  - Operations:
    - getPremium(row, col) returns the type of premium tile
    - setPremium(row, col, type) sets the tile as a specified premium tile

BoardLoder:
- Data structures used:
  - No data structures used
  - Explanation:
  - Works by calling the function importXML that will draw data from the XML file and save it into usable board variables.

Dictionary:
- Data structures used:
  - ArrayList<String> acceptedWords
  - Explanation:

the dictionary contains a dynamic list of words loaded from a CSV. ArrayList allows dynamic resizing as words are added or removed, and it provides fast indexed access for checking or iterating through words

Operations:

acceptedWords.add(word) adds a new word

acceptedWords.contains(word) checks if the word already exists

Player:

Data structures used:

ArrayList<Tile> hand

ArrayList<PlacedTile> placedTiles

Explanation:

ArrayList is dynamic, which is important because the number of tiles can change during the game. ArrayList allows indexed access and iteration, which is useful for displaying hand or processing placed tiles

Operations:

hand.add(tile) adds a tile to the player's hand

hand.remove(tile) removes a tile once it's placed

placedTiles.add(new PlacedTile(...)) tracks moves for validation and scoring

showHand() iterates through the ArrayList

playWord() iterates through the ArrayList

PlacedTile:

Data structures used:

no data structures, just 2 int variables and 1 Tile variable

Explanation:

the Tile object stored in the variable "tile" stores most of the needed information, the row and col int variables store the position where the tile is placed

Tile:

Data structures used:

no data structures just 1 String variable

Explanation:

only needs to store a single string, has a getter function and the String is set on construction of the object

TileBag:

Data structures used:

List<String> tiles

Explanation:

Array list allows for dynamic addition and removal of tiles

Operations:

addTiles(letter, count) populates the bag

drawTile() randomly removes a tile and constructs a Tile

isEmpty() checks if the bag is empty

GameModel:

- Data structures used:
  - ArrayList<Player> players
  - Dictionary acceptedWords
  - ArrayList<Tiles> placedTiles
- Explanation:
  - Array list is dynamic and ordered, making it easy to iterate through the list for players turns
  - Dictionary is shared between all players and is static as the accepted words don't change mid game.
  - Array list is dynamic and ordered, which makes it very easy to add tiles into it and remove them for each player's turn.
- Operation:
  - addPlayer(Player) adds a player dynamically
  - nextPlayer() iterates through the player list to change the current player