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
- 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

The majority of the data structures and code between milestone 1 and 2 is very similar, the major differences arise from the way which the code now interacts with the terminal, and how the user interacts with the data structures within the code. In milestone 2, everything is interacted with through the user interface, which takes the classes GameModel, GameView, GameFrame and GameController to receive user inputs, update the relevant information to the model and push the changes to both the view and the model.