CS 5004 Module 8 Worksheet
Controllers & Views

Each worksheet is to help you remember and understand the materials contained in each module so that you can get the most out of your learning experience. We start by asking questions that are covered in the module but will also ask you to apply the knowledge in new ways. By taking the time to answer each question carefully using your own thoughts and words, these worksheets can serve as the basis for review for the more formal assessments in this class.

For this worksheet, we are providing you with this starter code that contains three interfaces for the game of TicTacToe implemented as a text-based model-view-controller application: `TicTacToeModel`, `TicTacToeView`, and `TicTacToeController`. The `TicTacToeModel` extends the `ReadonlyTicTacToeModel` interface that can be used to pass model data to the view. We are also providing you with a fully functional implementation of the model and a `Driver` class that is constructed for a console-based game of TicTacToe. Your job in this worksheet will be to implement the missing pieces.

1. Implement the `TicTacToeView` interface in a class named `TicTacToeTextView`. It should use an `Appendable` to display the different elements of the game for the user. Be sure to convert any `IOException` that is thrown by the `Appendable` into an `IllegalStateException`.

2. Implement the `TicTacToeController` interface in a class named `TicTacToeTextController` that implements the logic of the game of TicTacToe. As the controller, it should tell the model what to do and the view what to show while handling the user input. It should gracefully handle any errors that arise and let the user know what happened (in other words, it should not print a stack trace, instead print a message that lets the user know what happened.

Once you have completed these two classes, the `Driver` class should allow you to play a fully-functional game of TicTacToe using the keyboard as input and the console for output.

For the remainder of this worksheet, we want you to practice writing test cases for the controller that is isolated from the model. We have started this in the `TicTacToeControllerTest` class. This is not an exhaustive list of tests but rather it is a start that will give you an opportunity to practice working with mock classes. We have started a mock of the model for you in the `MockTicTacToeModel` class. Feel free to add additional mock classes as you work through these exercises.

3. Finish the implementation of the `testQuitWithoutMove()` test by filling in the expected value for the model log and for the view log.

4. Implement a test for input where the q comes instead of an integer for the row.

5. Implement a test for input where the q comes instead of an integer for the column.

6. Implement a test for input where non-integer garbage comes instead of an integer for the row.

7. Implement a test for input where non-integer garbage comes instead of an integer for the column.

8. Implement a test for input where the move is integers, but outside the bounds of the board.

9. Implement a test for input where the move is integers, but invalid because the cell is occupied.

10. Implement a test for input where the move is valid but there is no winner.

11. Implement a test for input where the move is valid and there is a winner

12. Take a moment to think about what tests might be missing from this worksheet and list them here.

Some other tests I can think of are the following:

- A test for input where the row and col are double or long.
- A test for input where the game is not end.
- A test for input where the game is not end but there is a winner.
- A test for input where the game is not end and there is no winner.
- A test for input where the game is end and there is no winner.