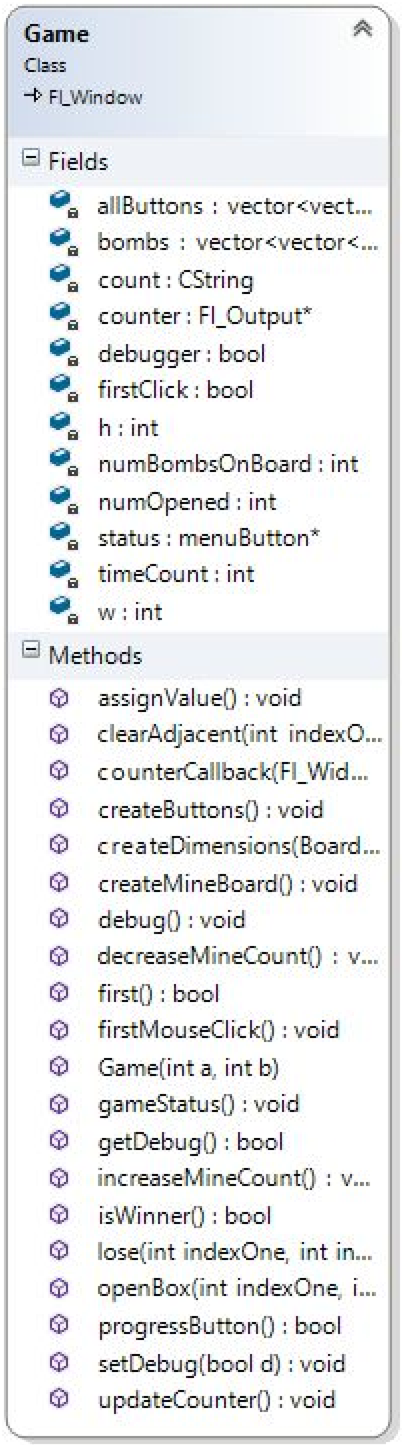
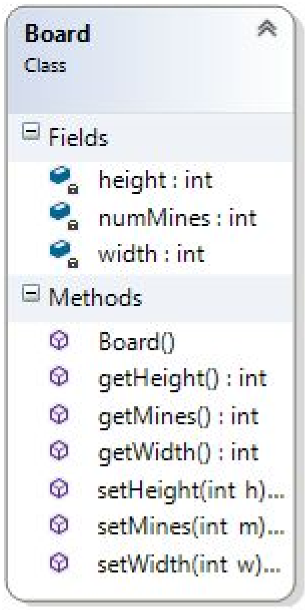
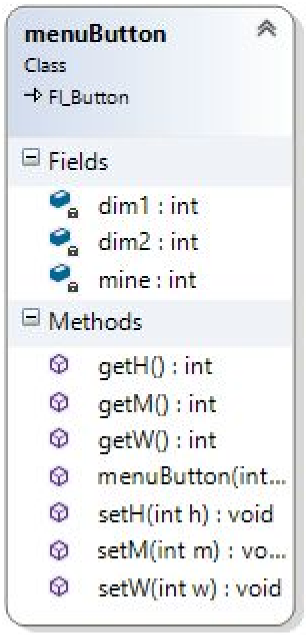
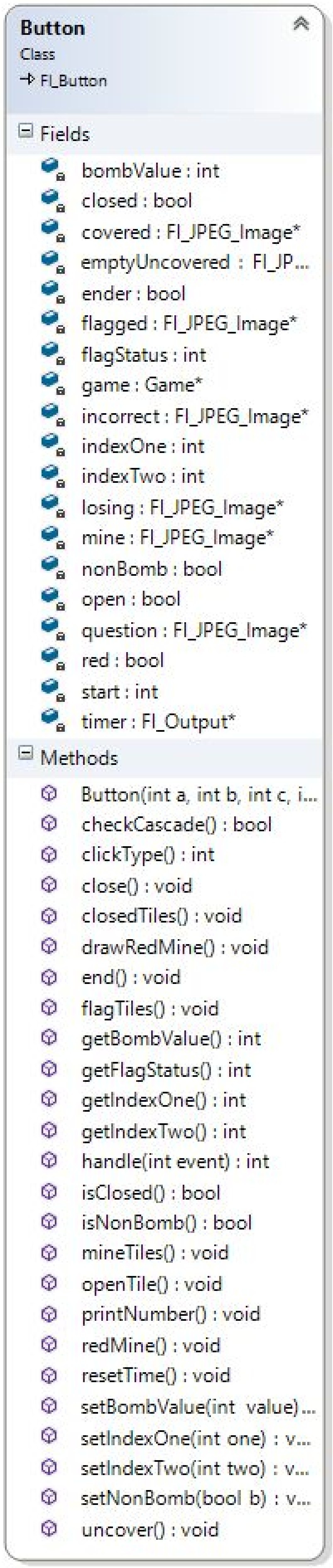
Final Design: UML Class Diagrams









Minesweeper Project Final Design

- Menu bar is open upon compilation
    - Four different buttons giving the different types of difficulty/custom
    - About button that shows the developers information
- Upon selection, new window pops up
    - Depending on debug mode, a board is created of a Buttons (derived from Fl_Button) that have new qualities: bearing flags, mines, and cluevalues
- The game is begun with the event of a left or right click
    - Timer begins ticking
    - The handle() function of each button (override the virtual funtion of Fl_Widget) handles the different types of events that can affect the game:
        - Simultaneous click (with the event of FL_PUSH on both mouse buttons)
        - Left click (FL_RELEASE)
        - Right click (FL_RELEASE)
    - Each event is associated with different types of reactions; flags and question marks are only involved in right clicking, and winning, losing, mine and number revealing are only associated with the left click
- The Game class is the one that deals with interactions between buttons
    - This includes cascading and the action of simultaneous clicks that check the nature of the surrounding tiles
    - The game is in charge of setting up the array of Buttons, setting the window dimension (in our version, it is derived from Fl_Window)
    - It displays the widgets on its window, such as the counter
    - I opted to make the widget in the Button class because its nature depends on the first click on a Button, which I found to be easier to track in that class itself
- Interactions between buttons ultimately determine the act of winning or losing the game
    - Uncovering all non-mines results in a win
        - The Game class can track how many tiles are open as every time a tile is uncovered, a boolean value *open* in the Button class can be turned true
            - Traversing through all the Buttons and counting how many are open can determine the winning condition
        - Losing is uncovering a red mine
            - However, the board must show the location of all the mines
            - Since this condition involves other tiles and not just the losing tile, the Game class must also be in charge of addressing the conditions of loss
- The constant information sharing between Game and the Buttons involves letting each Button have a pointer towards the Game it is a part of
    - This was done in the constructor; it allows the Button to affect elements of the Game such as calling the function lose() when a mine is clicked upon
    - This is necessary because the event of mouse clicks must somehow translate to actions on the Gameboard, and this is a fairly good solution
- Pressing the Status button allows for a new Game to be made, whether lost or won
    - This is done via a callback; the Status is a button also derived from Fl_Button and its callback involves creating a new Window/Game essentially duplicating that of the previous Game
    - A new type of game with different dimensions can be made by simply referring back to the Menu window that remains on the screen