# Homework 3

Due Friday, March 15th at 11:59 pm

100 Points

For this assignment we will continue to work on our ConnectX game, but we will be adding new requirements. The old requirements are still required, unless they are replaced by a requirement in this prompt.

**New Requirements**

- At the start of each new game, the users will be able to specify the number of players for each game. They must enter at least 2 players, and at most 10 players. Note that `IGameBoard` does not care about the number of players at all, this is just a requirement of `Connect4Game`.
- After selecting the number of players, each player will be able to pick what character will represent them in the game. You can assume they will always enter just a character. Always convert the character to upper case. Each player must have their own unique character, and the game should not allow a player to pick a character that is already taken.
    - Note: if you have a string, you can call the `charAt(0)` method to get the first character. If your string only has one character, this is an easy and safe way to convert a string that was provided by user (`scanner.nextLine()` returns a string) and convert it to a character.
    - Note: The List interface has a method called `contains(Object obj)` that will return true if the `obj` already exists in the List.
- At the start of each new game the players can choose whether they want a fast implementation or a memory efficient implementation. The game must check to make sure they entered either f, F, m, or M as a choice at this option.
    - Remember, if you use == to compare strings, you will check to see if the memory location is the same. Use the `equals()` method to check if they have the same abstract value.
- The number of players and their characters can change if they choose to start a new game.
- The choice of the fast or memory efficient implementation can be changed if they choose to start a new game.
- Our game must now have two different implementations of our `IGameBoard` interface
    - We already have our fast implementation which uses a 2D array. However, we will now be checking to make sure the implementation is efficient. So you should not be checking the entire row if you are looking for the horizontal win, you should be checking starting at the last position played.
    - Our memory efficient implementation is detailed below.
    - Our `Connect4Game` class should be able to easily switch between these two implementations. Remember, programming to the interface allows for the code to be the same except for the call to the constructor. You should not have essentially 2 different versions of the main function (one for each method) but one main function that has an if statement to call the appropriate constructor of your `IGameBoard` object.

- Additionally, you must add an abstract class called `AbsGameBoard` that implements `IGameBoard`. This abstract class will not contain any private data, but it will contain the overridden implementation for the `toString` method. That method can be implemented without accessing private data just by calling primary methods. It just can't be a default method in the interface since the interface cannot override a method from a class. Your other implementations (`GameBoard` and `GameBoardMem`) should extend `AbsGameBoard` to inherit the `toString` method

**New Implementation**

You will need to provide a new class called `GameBoardMem` that will extend `AbsGameBoard` which implements the `IGameBoard` interface. This new implementation will be slower, but more memory efficient.

To represent the game board you will use a `Map`. `Map` is another Generic collection that uses a `key` – `value` pair. The `key` field of our `Map` will be an `Integer` that represents the column number. The `Value` associated with that column number will be a `List` of `Characters`. If there is nothing in that column, then there will be no `key` or `List` in the `Map` to represent the column. Do not fill the `List` with `' '` like we did with the array. The memory efficiency of this implementation comes from the fact that and empty board does not use up any space in memory. As tokens are added to that column, you add them onto the `List` for that column. So the front of the `List` is row 0 and the token on the top of the column will be the last character in the `List`.

There is a video available on Canvas that discusses some `Maps` in more detail. Please note that at the end of the video it mentions "How to use Maps in HW 3." This old slide is referring to a different homework project, and not relevant to our current assignment. Please ignore that slide and part of the video.

While this may seem like a lot of work to create this new implementation, remember that your new implementation only needs to provide code for your primary methods, and the constructor. All of your secondary methods should be handled by default implementations in the interface, and `toString` is provided by `AbsGameBoard`.

**Example Board**

Consider the following Board:

```
| 0| 1| 2| 3| 4| 5| 6| 7| 8|
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |X |  |  |  |  |
|  |  |  |  |Y |  |  |H |  |
|  |O |X |X |Y |  |  |O |H |
```

In our implementation from Homework 2, this board would be represented in memory as an 8 by 8 2D array of characters. Blank spaces in the board are represented by that position in the array containing a `' '` character. While we can quickly access any place on the board by jumping to that index in the array, our board takes a lot of memory because it is always storing 64 characters. Now consider what this board would look like in our memory efficient implementation.

In our memory efficient implementation we would have a `Map<Integer, List<Character>>` with our `key` being the number of the column. Columns 0, 5 and 6 are currently empty, so those `key's` won't even exist yet in our `Map`. For our columns that do have characters in them, we will have `keys` and matching `Lists`. So if we asked our `Map` to return the `value` for `key` 4 it would return a `List` of `Characters` with `<'Y', 'Y', 'X'>` in it. The front of the list is the bottom row, and as new tokens are added, we can add more characters to the `List`.

Our Key Value pairs for this board would be as follows:

| Key | Value |
|---|---|
| 1 | <'O'> |
| 2 | <'X'> |
| 3 | <'X'> |
| 4 | <'Y', 'Y', 'X'> |
| 7 | <'O', 'H'> |
| 8 | <'H'> |

**Contracts and Comments**

All methods (except for the main) must have preconditions and post-conditions specified in the Javadoc contracts. All methods (except for main) must also have the `params` and `returns` specified in the Javadoc comments as well. You must include a complete interface specification in `IGameBoard`, and write the contracts not included in the interface specification in both implementations. You must provide correspondences in both implementations.

While contracts were not re-graded for correctness in Homework 2, they will be in this assignment. You will need to make corrections to your contracts.

You code must be well commented. The Javadoc comments will only be enough for very simple methods. Remember, your comments will help the grader understand what you are trying to do in your code. If they don't understand what you are trying to do, you will lose points.

**Sample input and output**

```
How many players?
12
Must be 10 players or fewer
How many players?
1
Must be at least 2 players
How many players?
4
Enter the character to represent player 1
X
Enter the character to represent player 2
O
Enter the character to represent player 3
```

```
X
X is already taken as a player token!
Enter the character to represent player 3
y
Enter the character to represent player 4
h
How many rows should be on the board?
8
How many columns should be on the board?
12
How many in a row to win?
4
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
f
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Player X, what column do you want to place your marker in?
2
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |X |  |  |  |  |  |  |  |  |  |

Player O, what column do you want to place your marker in?
7
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |X |  |  |  |  |O |  |  |  |  |
Player Y, what column do you want to place your marker in?
13
Column cannot be greater than 11
Player Y, what column do you want to place your marker in?
4
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |X |  |Y |  |  |O |  |  |  |  |

Player H, what column do you want to place your marker in?
7
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |H |  |  |  |  |
|  |  |X |  |Y |  |  |O |  |  |  |  |

Player X, what column do you want to place your marker in?
3
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |H |  |  |  |  |
|  |  |X |X |Y |  |  |O |  |  |  |  |

Player O, what column do you want to place your marker in?
1
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|   |   |   |   |   |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |   |   |O  |   |   |   |   |

Player Y, what column do you want to place your marker in?
4
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |Y  |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |   |   |O  |   |   |   |   |

Player H, what column do you want to place your marker in?
8
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |Y  |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |   |   |O  |H  |   |   |   |

Player X, what column do you want to place your marker in?
4
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |X  |   |   |   |   |   |   |   |
|   |   |   |   |Y  |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |   |   |O  |H  |   |   |   |

Player O, what column do you want to place your marker in?
6
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |X  |   |   |   |   |   |   |   |
|   |   |   |   |Y  |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |   |O  |O  |H  |   |   |   |

Player Y, what column do you want to place your marker in?
3
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |X  |   |   |   |   |   |   |   |
|   |   |   |Y  |Y  |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |   |O  |O  |H  |   |   |   |

Player H, what column do you want to place your marker in?
5
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |X  |   |   |   |   |   |   |   |
|   |   |   |Y  |Y  |   |   |H  |   |   |   |   |
|   |O  |X  |X  |Y  |H  |O  |O  |H  |   |   |   |

Player X, what column do you want to place your marker in?
6
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |X  |   |   |   |   |   |   |   |
|   |   |   |Y  |Y  |   |X  |H  |   |   |   |   |
|   |O  |X  |X  |Y  |H  |O  |O  |H  |   |   |   |

Player O, what column do you want to place your marker in?
2
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |X  |   |   |   |   |   |   |   |
|  |  |O |Y |Y |  |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player Y, what column do you want to place your marker in?
2
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |Y |  |X |  |  |  |  |  |  |  |
|  |  |O |Y |Y |  |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player H, what column do you want to place your marker in?
6
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |Y |  |X |  |H |  |  |  |  |  |
|  |  |O |Y |Y |  |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player X, what column do you want to place your marker in?
5
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |Y |  |X |  |H |  |  |  |  |  |
|  |  |O |Y |Y |X |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player O, what column do you want to place your marker in?
5
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |Y |  |X |O |H |  |  |  |  |  |
|  |  |O |Y |Y |X |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player Y, what column do you want to place your marker in?
3
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |Y |Y |X |O |H |  |  |  |  |  |
|  |  |O |Y |Y |X |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player H, what column do you want to place your marker in?
5
| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |H |  |  |  |  |  |  |
|  |  |Y |Y |X |O |H |  |  |  |  |  |
|  |  |O |Y |Y |X |X |H |  |  |  |  |
|  |O |X |X |Y |H |O |O |H |  |  |  |

Player H Won!
Would you like to play again? Y/N
y
How many players?
2
Enter the character to represent player 1
r
Enter the character to represent player 2
t
How many rows should be on the board?
5
How many columns should be on the board?
5
How many in a row to win?
3
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
m
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Player R, what column do you want to place your marker in?
0
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|R |  |  |  |  |

Player T, what column do you want to place your marker in?
4
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|R |  |  |  |T |

Player R, what column do you want to place your marker in?
0
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|R |  |  |  |  |
|R |  |  |  |T |

Player T, what column do you want to place your marker in?
4
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|R |  |  |  |T |
|R |  |  |  |T |

Player R, what column do you want to place your marker in?
0
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|R |  |  |  |  |
|R |  |  |  |T |
|R |  |  |  |T |
Player R Won!
Would you like to play again? Y/N
y
How many players?
2
Enter the character to represent player 1
x
Enter the character to represent player 2
e
How many rows should be on the board?
5
How many columns should be on the board?
5
How many in a row to win?
3
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
d
Please enter F or M
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
k
Please enter F or M
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
F
| 0| 1| 2| 3| 4|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Player X, what column do you want to place your marker in?
…
```

**The Program Report**

Along with your code you must submit a well formatted report with the following parts:

Requirements Analysis

Fully analyze the requirements of this program. Express all functional requirements as user stories. Remember to list all non-functional requirements of the program as well.

Design

Create a UML class diagram for each class and interface in the program. Also create UML Activity diagrams for every method in your `Connect4Game` class and every method in your `GameBoard` or `GameBoardMem` class (except the constructor). If a method is defined as a default method in the interface, you need to update it to remove any mention of the 2D array or other private data of the `GameBoard` class, and instead refer to primary methods. If a method is provided as a default method in the interface you only need to provide an Activity Diagram for the default method, you do not need to include the diagram for each implementation as well. Your diagram for `toString` should not reference private data since it will exist in the abstract class.

Testing

No test cases are required for this assignment

Deployment

Provide instructions about how your program can be compiled and run. Since you are required to submit a makefile with your code, this should be pretty simple.

**Additional Requirements**

- You must include a makefile with your program. We should be able to run your program by unzipping your directory and using the make and make run commands.
- Remember, this class is about more than just functioning code. Your design and quality of code is very important. Remember the best practices we are discussing in class. Your code should be easy to change and maintain. You should not be using magic numbers. You should be considering Separation of Concerns, Information Hiding, Encapsulation, and Programming to the interface when designing your code. You should also be following the idea and rules of design by contract.
- A new game should always start with player 1
- You class files should be in the cpsc2150.connectX package
- Your code should be well formatted and consistently formatted. This includes things like good variable names and consistent indenting style.
- You should not have any dead code in your program. Dead code is commented out code.
- Your code must be able to run on the school Unix machines. Usually there is no issue with moving code from intelliJ to Unix, but some times there can be issues, especially if you try to import any uncommon java libraries.
- Code that does not compile will receive a 0.
- Your input and output should match the example input and output provided. The TAs will prepare scripts to help with the grading process, and their scripts will rely on the input and output matching my example.
- Your UML Diagrams must be made electronically. I recommend the freely available program draw.io, but if you have another diagramming tool your prefer feel free to use that.
- While you need to validate all input from the user, you can assume that they are entering numbers or characters when appropriate. You do not need to check to see that they entered 6 instead of “six.”

**Tips and Reminders**

- Do your design work before you write any code. It will help you work through the logic and help you avoid writing code just to delete it later.
- Carefully consider how you can use your available methods to solve problems both in your main function, and in methods in the GameBoard class.
- When writing your code, try to consider how things may change in the future. How can you design your code now to be prepared for future changes and requirements?
- **BEWARE THE IDES OF MARCH!** This project is due the last day before Spring Break begins. No one wants to still be here working on this assignment, so get started early!

**Submission**

You will submit your program on Handin. You should have one zipped folder to submit. Inside of that folder you'll have your package directory (which will contain your code), your report (a pdf file) and your make file. Make sure your code is your java files and not your .class files.

**Academic Dishonesty**

This is an INDIVIDUAL assignment. You should not collaborate with others on this assignment. See the syllabus for further details on academic dishonesty.

**Late Policy**

Your assignment is due at 11:59 pm. Even a minute after that deadline will be considered late. Make sure you are prepared to submit earlier in the day so you are prepared to handle any last second issues with submission. No late assignments will be accepted.

**Checklist**

Use this Checklist to ensure you are completing your assignment. Note: this list does not cover all possible reasons you could miss points, but should cover a majority of them. I am not intentionally leaving anything out, but I am constantly surprised by some of the submissions we see that are wildly off the mark. For example, I am not listing "Does my program play Connect 4?" but that does not mean that if you turn in a program that plays Checkers you can say "But it wasn't on the checklist!" The only complete list that would guarantee that no points would be lost would be an incredibly detailed and complete set of instructions that told you exactly what code to type, which wouldn't be much of an assignment.

- Can I change the size of my game board?
- Can I set the number in a row needed to win?
- Can I change the number of players in my game?
- Can each player select their own character to represent them?
- Does my game prevent two players from picking the same character?
- Did I create an interface for my Game Board?
- Did I provide my interface Specification?
- Did I move my contracts from my GameBoard class to my Interface?
- Did I implement all secondary methods as default methods in my interface?
- Do I have both implementations of my game board?
- Did I provide an abstract class with the toString method, and do both of my implementations extend it?
- Do my players get to choose which implementation to use?
- Does my game allow for players to play again?
- When players start a new game can they change the size of the board or the number to win?
- When my players start a new game, can they change the number of players and the characters that represent them?
- When my players start a new game, can they choose which implementation to use?
- Does my game behave exactly the same for both implementations?
- I don't have two copies of my main function, 1 for the fast implementation and one for my memory efficient implementation.
- Did I correct my contracts?

- Are my implementations efficient?
- Does my game take turns with the players?
- Does my game correctly identify wins?
- Does my game correctly identify tie games?
- Does my game run without crashing?
- Does my game validate user input?
- Did I protect my data by keeping it private, except for public static final variables?
- Did I encapsulate my data and functionality and include them in the correct classes?
- Did I follow Design By Contract?
- Did I provide contracts for my methods?
- Did I provide correspondences to tie my private data representation to my interface?
- Did I follow programming to the Interface?
- Does my Game board still print correctly with changing board sizes and more than two players?
- Did I comment my code?
- Did I avoid using magic numbers?
- Did I use good variable names?
- Did I follow best practices?
- Did I remove “Dead Code.” Dead Code is old code that is commented out.
- Did I make any additional helper functions I created private?
- Did I use the static keyword correctly?
- Did I add in new user stories that capture my new functionality?
- Did I update my activity diagrams to ensure any secondary methods don’t refer to private data?
- Did I create a class diagram for my new implementation and abstract class?
- Did I provide a working make file?
- Does my code compile and run on Unix?
- Is my program written in Java?
- Does my output look like the provided examples?