

# PIC 20A: Homework 3 (due 2/23 at 5pm)

## Submitting your homework

The zip file you extracted to find this pdf includes files called `TicTacToe.java` and `ConnectFour.java`. In this assignment, you will edit these files and submit them to Gradescope.

- Upload `TicTacToe.java` and `ConnectFour.java` to **Gradescope** before the deadline.
- **Name** the files exactly as just stated.
- Do **not** enclose the files in a folder or zip them.  
Do **not** submit `Player.java`, `TwoPlayerBoardGame.java`, or `TestGames.java`.  
You should be submitting exactly **two files** and they should have the **extension** `.java`.
- Be sure that your code **compiles and runs** with `Player.java`, `TwoPlayerBoardGame.java`, and `TestGames.java` using **Adoptium's Temurin Version 11 (LTS)**.

## Tasks

1. Watch the first season of the IT Crowd to learn more about COMputers.
2. Open `Player.java`. Understand what it accomplishes.  
During this assignment **you should not edit** `Player.java`.
3. Open `TwoPlayerBoardGame.java`. Understand what it accomplishes.
  - (a) `TwoPlayerBoardGame` is an **abstract** class, so one cannot instantiate it. It is written to be inherited from and to provide a framework for designing simple two player board games.
  - (b) The `play` method is marked **final** and so it cannot be overridden. Its definition consists of generic code that will allow playing a two player game and it makes use of a number of **abstract** methods.
  - (c) The `TicTacToe` and `ConnectFour` are concrete classes (`TestGames.java` instantiates them) which inherit from `TwoPlayerBoardGame`. They must override the methods that are marked **abstract** in `TwoPlayerBoardGame`.

During this assignment **you should not edit** `TwoPlayerBoardGame.java`.

4. Open `TestGames.java`. Understand what it accomplishes.
  - (a) Once you have edited `TicTacToe.java` and `ConnectFour.java`, this file will allow you to play Tic-Tac-Toe and Connect Four against unimpressive AI.
  - (b) It'll also let two humans play the games against one another.  
This will help you to test your games for bugs.
5. Confirm that you can compile `TestGames.java` (which forces the compilation of `Player.java`, `TwoPlayerBoardGame.java`, `TicTacToe.java`, and `ConnectFour.java`). As you edit `TicTacToe.java` and `ConnectFour.java`, you should frequently recompile and run `TestGames.java`.
6. Open `TicTacToe.java`. Understand what it accomplishes.
  - (a) Instance fields called `X0`, `row`, and `col` have been declared.
    - i. I decided that X always begins and I intended for `X0` to store whether it is X's move or O's move.
    - ii. `row` and `col` were created to store the most recent move.
  - (b) The `receiveMove` and `generateMove` methods have already been written for you. You can see how they make use of `row` and `col`. The implementations also give a very brief introduction to the `Scanner` and `Random` classes (note that the static fields `console` and `random` belonging to `TwoPlayerBoardGame` store references to instances of these classes).
  - (c) The constructor definition was the shortest code I could write to make everything compile. You need to edit this definition appropriately.
  - (d) The remaining methods are given the shortest definitions that allow compilation. You will edit their definitions so that the code runs as demonstrated in `demo.txt`.
7. Open `ConnectFour.java`. Understand what it accomplishes.
  - (a) Fields called `RY`, `col`, `ROWS`, `COLS` have been declared.
    - i. I decided that Red always begins and I intended for `RY` to store whether it is Red's move or Yellow's move.
    - ii. `col` was created to store the most recent move.
    - iii. The static fields `ROWS`, `COLS` store the most common dimensions of Connect Four.
  - (b) The constructor definition was the shortest code I could write to make everything compile. You need to edit this definition appropriately.
  - (c) The remaining methods are given the shortest definitions that allow compilation. You will edit their definitions so that the code runs as demonstrated in `demo.txt`.
8. Edit `TicTacToe.java` and `ConnectFour.java` so that you can play the games like in `demo.txt`. Some useful lines in `demo.txt` are...
  - (a) Line 11: I purposefully typed invalid numbers.
  - (b) Line 31 and 130: I purposefully typed a position that had already been filled.
  - (c) Line 70: I used distinct numbers so you can understand how positions are described.
  - (d) Line 161: I setup a draw between Roy and Moss.
  - (e) Lines 488 and 491: I purposefully selected a full a column.
  - (f) Line 779: Moss decided to mess with Jen before winning.