AI - CHECKERS SCORING CODE

One submission (FirstinitialLastname.java - choose one team member name) in the online lab survey per 2-4 person team. Due halfway into lab next week (2:25pm or 3:40pm). Include team member names. Your TA will run a tournament for everyone to watch.

Code and test multiple heuristics to "score the board" that you designed in last week. You will submit the code for your best playing heuristic in lab next week.

1. Download https://moss.cs.iit.edu/cs100/assign/CS100-CheckersFinal.zip Un-compress the files into a folder called CS100-CheckersFinal.
2. Open Eclipse. Click “File”… “Import”… from the Eclipse main menu. Expand “General”, select “Existing Projects into Workspace”, and click “Next”. In the “Select Root Directory” box, browse to your CS100-CheckersFinal folder, select the folder and click “Finish”. You should now see the new project in Package Explorer.
3. Expand the package, and “src”, and “submissions”. TestName.java is a shell that you can copy multiple times and code your different heuristics in. You should name each file (and class name in the file) with a descriptive word for the heuristic so when you compete each one against the heuristics we provide, you can keep track of the results and choose your best one for submission. The one you submit in lab next week must be named with first initial, last name of someone on your team (e.g. Bob Smith -> BSmith)
4. You need to do the following coding.
    a. In the constructor, update the name (this will be displayed when competing) and the section.
    b. The evaluateBoard method is where your code your board scoring heuristic. It must return an integer, positive values if good for you, negative values if bad for you. The relative size of the integer you return should reflect how good or bad the board is.
    c. Four boolean methods are available for you to call. They all take a single argument, a position (char) on the board (a 2D array of characters). See the board layout below and a sample method call in TestName.java.
        - ownChecker (char tile): own regular checker pieces
        - ownKing (char tile): own king checker pieces
        - oppChecker (char tile): opponent's regular checker pieces
        - oppKing (char tile): opponent's king checker pieces
    d. Compiling - The classpath file should be set up to have all the libraries and documentation included, so compiling should be automatically set up when opening the project in Eclipse.
    e. To run, a new run configuration must be added.
        1. At the top, click the small down arrow to the right of the green run button and choose “Run Configurations”
        2. Near the top left, click the “New launch configuration”. Name the configuration “CheckersPlay”
        3. In the “Main” tab, enter “checkers.Play” in the “Main class” text box
        4. In the “Arguments” tab, enter in the program arguments in the “Program arguments” tab. The only argument that is required is “-f <file> File submission testing, only include class name”. These arguments can be changed any time afterwards by the same steps as before.
        5. Click either 'Run' or 'Apply' and 'Close'
        6. Now to run, the green run button can just be used
    f. When running, you are given the option to compete the code you wrote (and chose with the –f argument) against 6 different heuristics we have provided. 1 and 2 are just return a random number. 3, 4, 5, 6 are progressively more complex heuristics, but they are not necessarily progressively more difficult to defeat. Four games total are played, with each opponent allowed to go first twice. Games are limited to 150 moves total, then a tie is called.

```
// the board, rows are numbered from top to bottom and left to right (from 1 to 8).
char [][] position
```

0	1	2	3	4	5	6	7	8
1		W		W		W		W
2	W		W		W		W	
3		W		W		W		W
4	_		_		_		_	
5		_		_		_		_
6	B		B		B		B	
7		B		B		B		B
8	B		B		B		B	

You should write your evaluateBoard method assuming the Evaluator's side to always be row 1 of the board, regardless of the color. When we swap colors we swap sides too

Arguments

Usage: java -cp <classpath> checkers.Play [-hV] [-d <num>] -f <file>

-h Print this help message.

-V Optional print out board after each move.

-d <num> Optional number of depth to search for possible moves. Must be even. Default is 6

-f <file> File submission testing, only include class name