Bonus Coding for Assignment 1-2

All of these bonuses, combined, will add to an extra two marks to your overall assignment (to the maximum permitted by your overall practical mark). In order to qualify for these marks, your assignment must be fully compliant with the requirements laid out in the assignment document, and it must pass all the tests— the test scenarios should all operate as expected.

You must include a readme file (preferably in .txt format) in your submission that lists what bonuses you're implementing (you may implement them all of course), and contains any explanations the bonus requires.

NOTE: As these are bonuses, they will be held to a very strict marking standard.

Bonus 1: Splash Screen Progress Bar

Difficulty: Easy

Implement, in your splash screen, a working progress bar. It must read, "Please wait while game assets are loading…" inside the bar itself. The bar should fill in over the duration of the splash screen. You will need to implement a delay of some sort. See Hybrid 6.

Bonus 2: Better User Feedback

Difficulty: Easy.

The UI is not very clear as to whose turn it is. Make it clearer. You can do so by having an extra message appear in the output window (the pink zone), and by highlighting the correct player name in the scoreboard area.

Bonus 3: Better mouse handling.

Difficulty: Medium

By adding *only one more listener*, permit the user to make moves by clicking on the board directly. You may not add listeners to each square on the board. Lab 3 MouseTest may be of value to you for hints on how to implement this.

In your file, give me a quick explanation of the method you used and how you implemented it. One paragraph will do.

Bonus 4: Saving and Loading.

Difficulty: Hard

There are two menu options under File that are currently disabled: Save and Load. Include two methods in the Model, "saveGame" and "loadGame". The file they save to should have a relative path and be in the main directory. The code should ensure there is a file to load in the

first place (if there is none, the load fails). It should report progress (no save file/save successful/load successful) upon completion.

Include the name of the file in your documentation so I know what to look for, and an explanation of the format you've chosen to save your file in. I will test it by deleting the save file (fair warning).

Bonus 5: AI Opponent:

Difficulty: Hardest

The AI logic should exist in its own class, which is a separate Model and should therefore have "Model" in its class name, but you may call it what you want apart from that. It should play white (player 2). While the AI doesn't need to be good, any move it should attempt to do should be valid. (ie: it may not be trying moves entirely at random until it finds one that's valid, though it may have a list of valid moves and pick one of those. But it would be more fun if it was a more challenging opponent than that).

In your readme file, add an explanation of your logic. Name your methods, and give me a bit of an explanation of how you implemented your opponent and what it does. A couple paragraphs will suffice.

The numbers:

Easy: .5 (but there's two of them)

Medium: +2

Hard: +3

Hardest: +4

Total: 10 (x0.2% final bonus multiplier), for up to 2% extra marks.

Final Notes:

Ensure your bonuses do not break the MVC pattern. It would be kind of embarrassing to wind up breaking things when you were trying to add functionality instead.

But most importantly, have fun with this. Make this code your own.

"If I win, it's a bonus. If I lose, the sun still comes up the next day, and it's all good." – Ashley Barty