Project 3, Fascicle 4:
Out of Sync

Entire project due by May 18. **Suggested completion date for this Fascicle:** May 10

Async Game AI

As demonstrated in lab, we want our Minimax game AI to run in a background task so that the main WPF UI doesn't lock up while the AI is computing is best move. To do that, you need to run the `FindBestMove` inside of a task using `Task.Run`, then `await` the return value of that task before applying the AI's move and rebinding the UI. Any function that uses `await` must be marked as `async` in its signature, so your view model's `ApplyMove` method will need to be `async`, and you must change the return type to `Task` instead of `void`. That means the MouseUp event handler that calls `ApplyMove` must also be `async`, so it can `await` the result of `ApplyMove`. Easy!

However, you must be careful to prevent the user from making changes to the board while the AI is doing its work. The Undo button definitely modifies the board, but so too does moving the mouse (which probably calls `GetPossibleMoves` when the mouse enters a square, and that function calls `ApplyMove`....). This can't be allowed, so you will need to modify your UI code to lock down all UI interactions while the `ApplyMove` is running. I recommend setting the `IsEnabled` property of your window to false before calling `ApplyMove`, and then back to true after `ApplyMove` returns. Then add checks in all UI methods to return if `IsEnabled` is false.