Patterns in Blokus

Polymorphism

When designing the Player class we came to the realization that it would be beneficial to create child classes to distinguish between a CPU player and a human player. We determined this to be beneficial since the game is to handle a CPU turn differently than it does a human turn.

Coupling

Unfortunately, as we progressed further into designing the logic in control of looping the game and handling the AI, classes started to couple more. If given more time it is likely that we would've changed where methods are located to decreases dependencies between classes. This results in code that is slightly more difficult to extend as it very often causes issues with the connections between classes.

Information Expert

As game logic continued to be developed, the code was all written within the BoardUI class. As it could technically have been written in it's own class, it did all end up being stored in the same class which means that the BoardUI class is an Information Expert for the games logic.

Cohesion

This pattern was kept in mind the most when designing the game. Each class' methods are very strictly related to one another and they're focused on what they need to do for that specific class. In our Player class, the information is stored for choosing the moves which the CPU will make and how to generate hints for a human player. The methods for getting this information and setting it are all located in the class as well.