The Use of Patterns in your design

We want to reduce the amount of repetitive code that we produce. To do this we want to look at what possible patterns there are in the game.

If we look at how we decided to generate the colour palette, we can see this. When Grid is designing each cell, it calls from Data to find what colour it is supposed to be. This was meant so that we do not need separate classes to design the exact same board but with different colours. This is also implemented when we colour the robots. Instead of assigning them colours individually, we referenced the array of colours to assign the correct one.

Moving to a different line of thinking, when navigating to from the main menu to varying menus to change different settings. We wanted the processes to mimic each other so that it is easier for the user to understand what is going on. This was also done so that the flow of the code could follow the same pattern with only slightly altered. Building on this, the simple and complex boards follow similar patterns so the code does as well. Again, making it easier to read and understand the code.

This idea of incorporating as much as possible into the code, and having the different classes follow the same logic was attempted through-out the program.

We also utilized the Grasp principle of Information expert throughout our project. While deciding on which classes would be responsible for a function, we asked ourselves which class had the most information regarding the task. This can be seen I classes like grid and complexGrid. These classes held the information on the simple board and complexBoard and build the visuals of the boards. So it made sense for these classes to be responsible for any visual changes on the board such as highlighting chosen squares and changing the position of the robots.

Some patterns that could improve our project are low coupling and high cohesion. We tried to implement these patterns as best we could but if we had more time for this project we could have utilized these patterns to a much greater extent. This would reduce the amount of repetitive code and make it much easier to see what each class is responsible for.