**How does this design account for possible future networking?**

One of the aspects we wanted to account for when building this game was the possible of future networking. We attempted to do this in serval ways, mainly by trying to ensure that any classes that dealt with the logic of the game or how the game is implemented was separate from how the game is displayed. This way there would have to be minimal changes when attempting to build this in a browser. Another was how we dealt with data and how we built the data class.

Building on the point about having separate classes and functions for how the game works and how the game is displayed. This allows us to, if desired, scrape the display classes and start from scratch – possibly in a language that is friendlier to browsers, without having to reimplement what we have already built that is a core aspect of the game. Redesigning how the robot moves around the board will take much longer then redesigning how board is displayed.

Another aspect would be how data is stored. At the moment we are only saving the robots locations to Data.java when the user clicks save because there is simply no reason to monitor and store exactly where each robot is at every moment. This could be easily amended so that when a user moves a robot to sends a signal to the other player(s). Data.java on the other hand could be entirely be replaced with a database that stores the games of different accounts different games. From our implementation of Data.java we already know what we would want to save for each specific game.

From these two design implementations we believe that future networking should be easier than if we had designed Ricochet Robots differently.

**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.

This idea to incorporate as much as possible into the code was attempted through-out the program.

Moving to a different line of thinking, when navigating to from the main menu to different menus in order to change different settings.