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.

When looking at the Data class, it will hold all knowledge about the game setup. This allows for simple data passing when setting up the game and our board does not need to know how to interact with the settings. This again allows us to change how the GUI is implemented makes any changes to the settings low impact on the game.

From these design implementations we believe that future networking should be easier than if we had designed Ricochet Robots without them.