SPUR 2017

I initially wrote the plugin using a system of logics that each had child specs, which had child events and actions and predicates and all that. When I was writing it first I really wanted to make sure things were working so I had my parser writing files as it parsed each spec, starting with the first spec and working it's way up. Looking back on this it's not a very good idea as now when I was creating a new file that inherited events for example, it would take all the events from the previous file and write them first, then read the new events from the parser. The better way of doing this was to have my parser complete everything and then start creating files. This is not only better for the creating of files but readability and now adding new features is very easy as before it would be very difficult to add the new features. On July 16th I started refactoring all my code from the ground up to a new way of parsing and a new way of creating files.

The purpose of this file is to explain the new re-factored code so new features can be added easily. The code is heavily commented but the flow I feel needs to be explained a bit further.

The Files:

Activator.java – This file is here, I believe to added the plugin to the Rodin files and allow it to show up when you open Rodin.

/Handlers/SampleHandler.java – This is the start of the code really. Renaming this file is a huge hassle as you need to update all the references to it, which Eclipse won't do automatically for you so it's better just to leave it with this name. The *initialization()* method is the method hat runs when the button is clicked. It will get the current open file in the editor and convert all it's lines of code into an *ArrayList <String>*. I originally had this as an array but now if you want to add multi-file support, the code will now need to get all the current active files and store add all their lines to the *ArrayList<String>* '**evtCode**'. The *cleanCode()* method will remove all the blank lines and comments. It will only remove them from the internal systems so you can still read your comments in Rodin. It will then create a new parser object and pass the '**evtCode'** in as a parameter.

/Parcel – this is a folder contained the main meat of the plugin. The files can mostly be exaplained later but the main one is **Parser.java**.