ASSIGNMENT 1: DESIGN

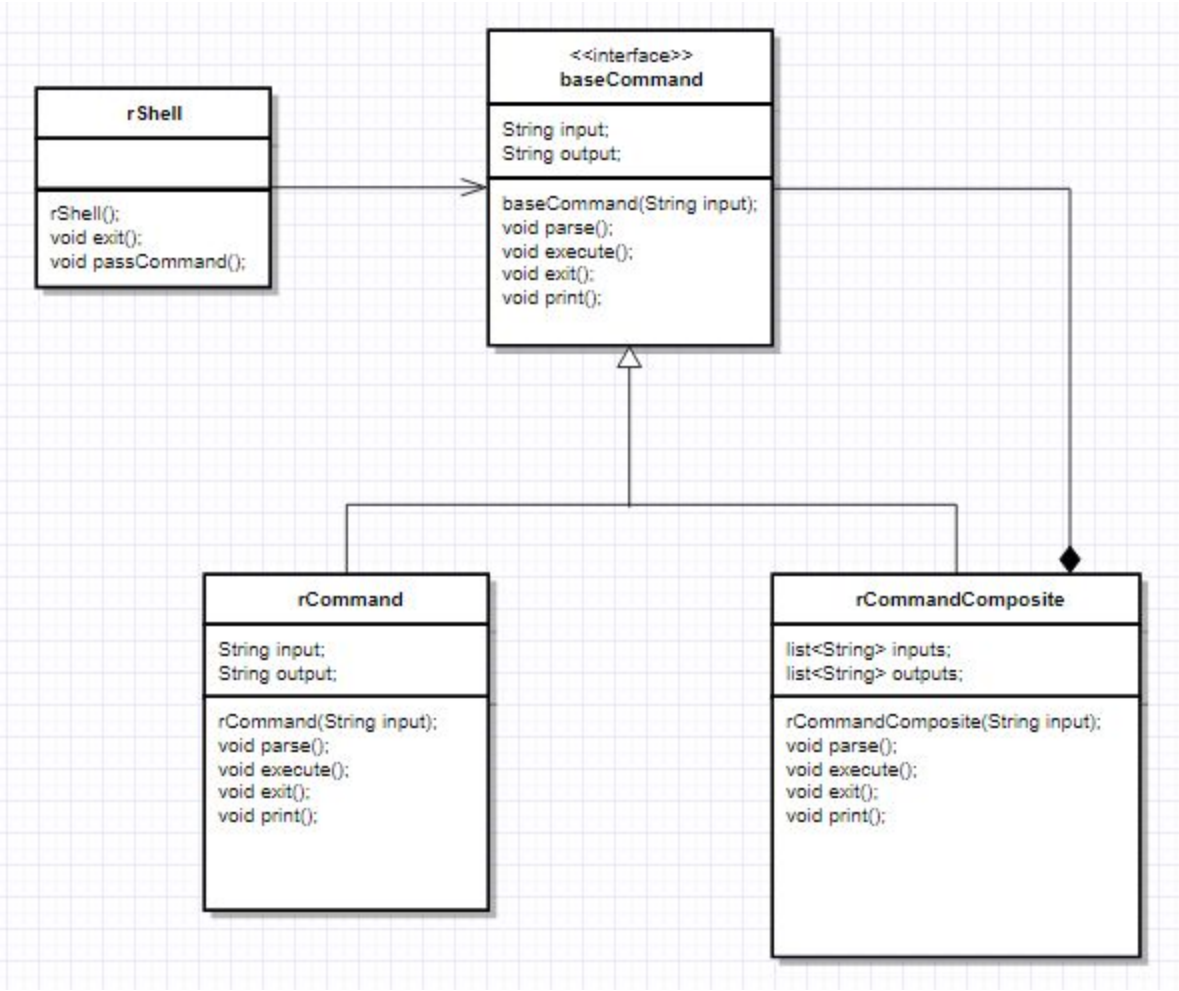
AUTHORS: ELIJAH NICASIO, LIAM HAN

OCTOBER 20, 2017
FALL QUARTER

Introduction:

We will be creating a class called ‘rShell’ which will be handling most of the user interaction. User input commands will be checked to see if classifies as a singular or multi-part command, and then passed to the relevant command class, either ‘rCommand’ or ‘rCommandComposite’, respectively. These command classes will then execute code passed by the user.

UML DIAGRAM



Classes/Class Groups:

- rShell - class that will run shell instances. Will continuously loop to get input until user inputs exit command, in which case exit() will be called. In every other case, passCommand() will be called, get and parse input, then pass to relevant command classes, rCommand and rCommandComposite, depending on whether input is a single or multi-part command.
- Command Group -
    - baseCommand - base class / interface that rest of command group will be derived from. Will have variables that store commands that are input, and also will store output that will be displayed as a result of input commands. Constructor will take in string, which will be parsed according to parse(). Will execute command, exit if exit is input, and print to display anything if needed
    - rCommand - primitive command, for single command, mostly same as base class
    - rCommandComposite - composite of command, parse will separate commands according to connectors and run their function. Stores inputs and outputs as list of strings

Coding Strategy:

Elijah will be responsible for setting up the github, breaking down the user stories we drafted in 'assignment 1', and logging each user story as an **issue** labeled as enhancements.

For the coding segment, we will break down our goals and assign each other a task from user stories:

Liam will be in charge of getting the rShell up and running and implementing the ability to take both single and multi commands on a single line by checking the input String for a connector followed by using the parse() method. Elijah will handle creating class groups listed above, building on our command group, and integrating the pieces together to have a working shell.

RoadBlocks:

- Possible bug when reading user input and not properly executing commands. We will use many test cases to try to get around this.
- Parse() method not catching a connector (i.e && ||) due to human error and not properly parsing command(s). We will again try to use many test cases to get around this.
- There may be conflicts when merging code, as we will each be working on different sections. We will communicate and keep in contact in order to clear up any conflicts that arise.