Assignment 1: Design

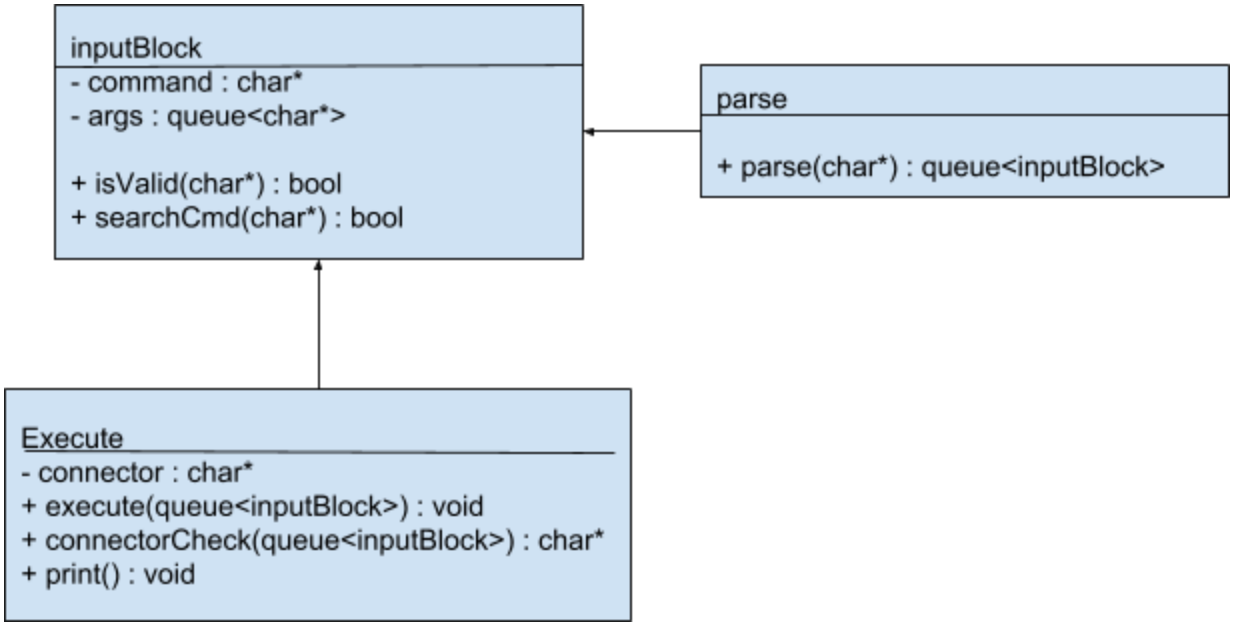
10/20/17 Fall Quarter

Kyle Bowen & Andrew Munoz

- **Introduction :**

To write a shell program in C++named “rshell” that loops indefinitely until the user exits. The shell will output “$ “ and await input. The shell will read in input one line at a time. Upon input, the program will parse the user input and will perform functions based on that input. Also, if the user input requires any command connectors, (such as “&&”, “||” , “;”) the shell will perform the multiple commands based on those connectors.

- **Diagram :**



- **Classes/Class Groups:**
    - Base Class
        - Class inputBlock: This class is going to check if the user input was valid or not. The parsed tokens from the parse command are going to be separated into either a command char* or a queue of arg char*. This will then be sent off to the execute class.
        - Class parse: This class is going to use strtok to parse the user input into tokens; creates queue of inputBlock objects. This class will also search for connectors, and confirm proper use of connectors. For example “;” must not be preceded by a space when the user inputs their command.
        - Class Execute: This class is going to be using the fork() and execvp() and waitpid() commands in order to execute the user input. If the execution of the command fails, then an error message will print.
- **Coding Strategy :**

We will compose a list of classes that need to be created, ordered by how important they are to make the shell seem functional. Kyle can be in charge of the parse class. Andrew can be in charge of implementing the execute class. Each class can be tested independently from each other to ensure the classes are working as intended. For example, some test cases can be made for the parse class in order to ensure that the desired commands are being split up. For the execute class it can be tested with different combinations of commands that are present in /bin/.

- **Roadblocks :**

  Issues that may arise during this assignment may occur when learning how to use functions such as strtok and execvp. Incorrect implementation of these functions will make coding for later assignments much more difficult. Another issue is how to add upon the shell in future assignments while still maintaining working code. Also virtual classes can be tricky.