Assignment 1: Design

October 14th, 2017

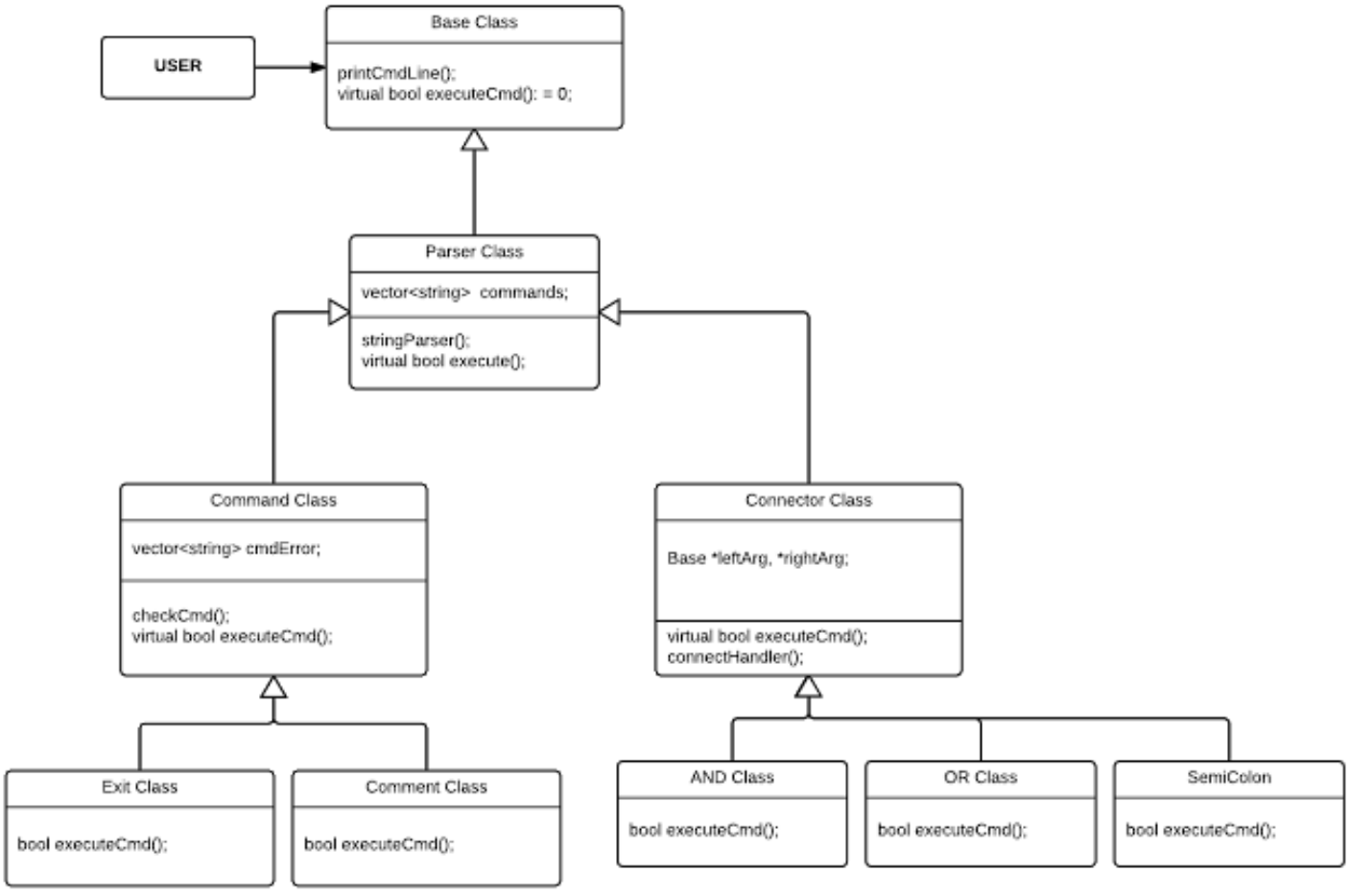
Fall Quarter 2017

By Mark Zemlany and Nate Dang

Introduction

Our rshell UML uses the composition design pattern in C++. A user inputs commands via strings and the user can use various symbols to indicate if they want to chain commands in various ways. The program will exit upon the user's command. We used the Composite strategy in order to break down our handling of input commands into classes.

Diagram



Classes

- Base Class
    - Contains the function that outputs the $
    - Contains the pure virtual bool executeCmd() command to pass to children
- Parser Class
    - Goes through the user input and looks for strings and ignores spaces.
    - A vector contains all commands that are recognized.
    - Contains the virtual bool executeCmd() command to pass to children
- Command Class
    - Vector cmdError contains the error responses depending on the error
    - checkCmd() determines if there's an error and returns the error
    - Contains the virtual bool executeCmd() command to pass to children
- Exit Class
    - Executes the exit command after user inputs exit command. Bool value is returned to indicate command
- Comment Class
    - Executes the comment code and passes in the comment to be written
- And Class
    - Calls the left and right arguments for itself and its parent. Makes sure that it is successful before executing using the return bool.
- Or Class
    - Calls the left and right arguments for itself and its parent. Makes sure that parent argument was not successful before executing using the return bool
- SemiColon Class
    - Automatically executes the following command regardless of successful execution.
- Connector Class
    - Contains Base pointers for each argument (left and right) given by the user called leftArg and rightArg. leftArg is the argument and rightArg is the connector symbol

Coding Strategy

Due to our plan being split into two major components, one of us will work on the Connector and its children while the other will work on the Command class and its children. As for the Base and the Parser class, we will work on them together to establish them as they are vital to our program and the other classes.

Roadblocks

Interfacing our own commands with the existing commands in the command prompt. Working asymmetrically due to time conflicts will also be a challenge. We are also worried about dealing with all the possible bugs that may arise working in the command prompt. Learning syscalls is also a large concern of ours as it's something that we don't have any experience with.

We are also concerned with the potential complications we'll have with github when it comes to branches and how to deal with errors such as merge conflicts and diagnosing those problems. Continuing on the idea of merging branches, being responsible for coding our assigned classes and having them work correctly together when combining them could pose some issues if we're not careful.

Most importantly, we are most concerned about creating a solid base of code so that it doesn't hurt is in the future