ASSIGNMENT 1: DESIGN

11/01/18

FALL 2018

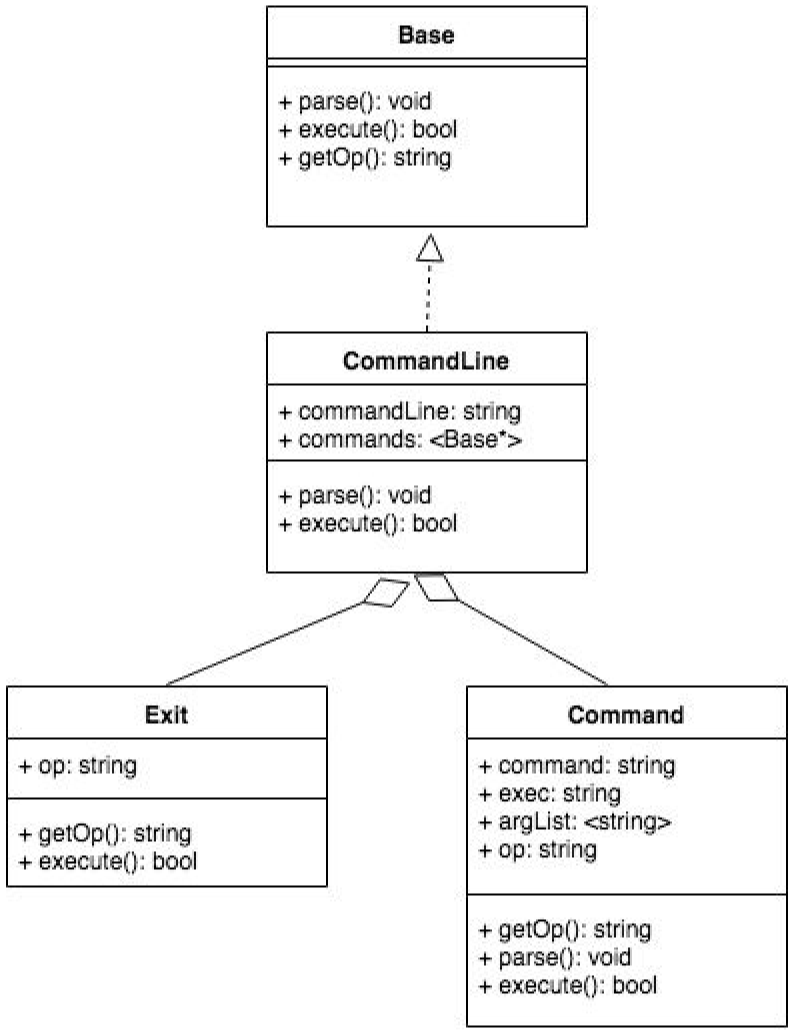
Spyridon Catechis

Rahul Nair

Introduction:

We plan to utilize a composite design pattern to build a command shell called rshell. We aim to be able to print a command prompt, read in a line of command(s) (and connector(s)) from standard input, and execute the appropriate commands using fork, execvp, and waitpid. Our solution involves using an abstract base class that represents the entire command line interface. As well as a composite class called CommandLine. CommandLine represents the user input string which may be an aggregation of smaller commands. The leaf's of our composite design pattern will be two types of concrete classes, Command and Exit. They represent the two types of commands that may be found within a CommandLine object.

UML



Coding Strategy:

1. Both partners write the abstract base class together, while discussing project outline.
2. Partner A writes one Leaf class while partner B writes the other.
3. Partner A writes the CommandLine composite class while partner B writes gtest functions for the individual functions in each class.
4. Partner B reviews the CommandLine composite class.
5. Both partners create unique tests for the whole structure aiming to account for edge cases.

Classes/Class Groups:

1. **Base** - Abstract Base class and user interface that acts as the parent for the rest of the classes
2. **Command Line** - Command Line will be the a class that inherits from the base class and acts as the composite class of the design pattern. It will be the class that takes in the input from the user. Its parse function will separate the entire string into a command that consists of the executable, argument, and operator. Its execute function will allow one command to interact with another based on its operator. It will return a bool value which will be true if the entire command inputted by the user worked.
3. **Command** - Command will be a leaf class that has instances of it inside Command Line. It will have instances of it inside the Command Line class, which will have a container that stores Commands. It has 3 important variables, exec, op, and arglist. Exec and op will hold the executable and operator portions of the string respectively. The arglist will be a vector that holds holds a string of words in each of its nodes. Its parse function will take the parsed string from Command Line and split it even further, storing the executable, argument, and operator into their respective variables. The execute function will actually execute the executable and argument stored in the object using syscalls.
4. **Exit -** will be a special subclass and a leaf of Base that will also be a leaf class. It will be used within the evaluate function of the Command class and will exit the program if the user inputs “exit” or if there is an error in executing the command.

Roadblocks:

- Problem: Issue with certain border cases and particular inputs
    - Solution: Create specific test cases to fit these cases and slowly close in to the actual problem
- Problem: Needing to change the structure of the classes and design of classes since current method does meet criteria
    - Redesign a new UML diagram and change class specification information so that project layout is clearly defined
- Problems with collaboration and Git clashes
    - Always make sure to define what work each member is doing so that clashes are reduce and always make sure to pull from upstream before pushing from local repo.