Assignment 1: Design

11/1/18

Fall 2018

CS 100, Section 001

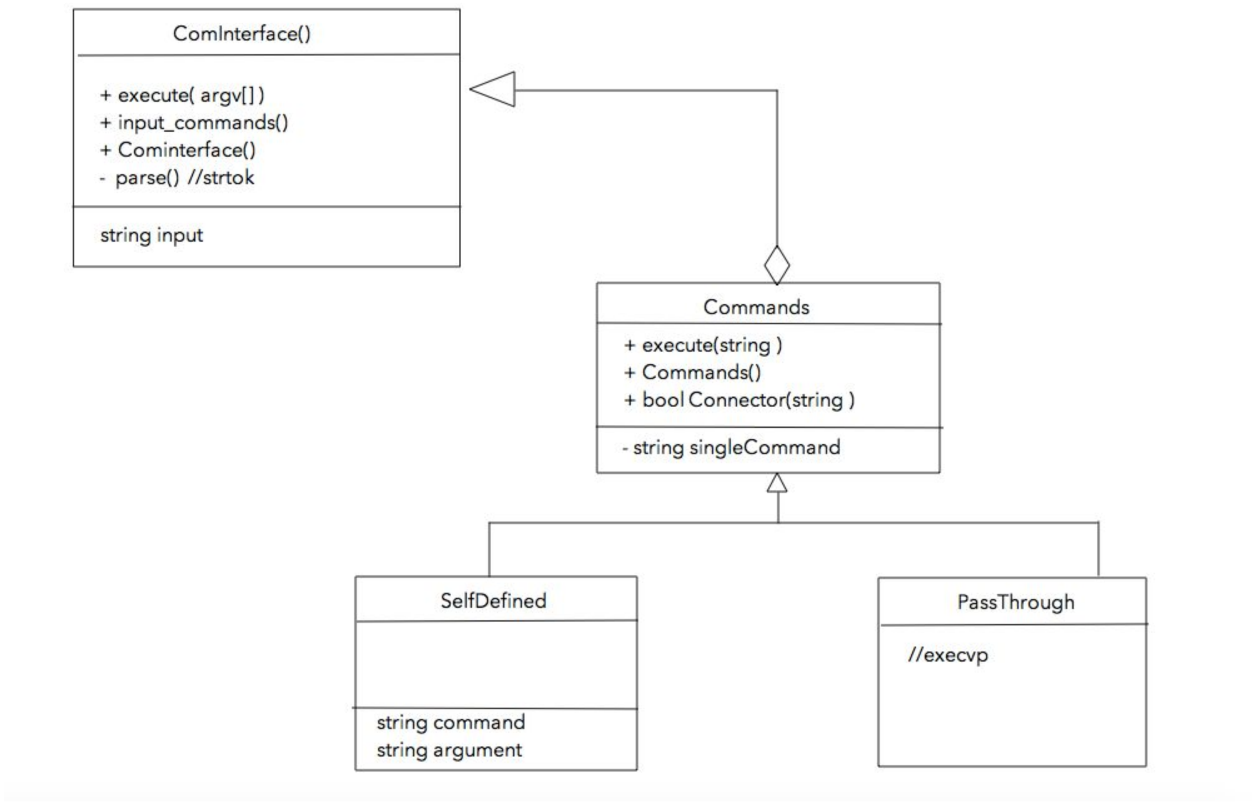
Nivedha Karthikeyan, nkart001

Erin Wong, ewong012

Introduction

We identified three main things that the specs are requiring of us. We need to create a basic command shell that prints out a command prompt, read in a line of command while taking into consideration any connectors that might be present, and execute the commands we take in.
We implemented a composite design pattern, initially, in order to implement the required parameters.

Diagram



Classes/Class Groups

ComInterface - We needed three things from the Command Interface component. We needed it to take the input of commands, parse through while again taking into consideration any connectors or errors, and then execute the list of commands after that. To do this we created three functions to do each of these things, and included a constructor and a public variable *input* to hold the string upon receiving it from the command line.

Commands - This composite was created to handle a few more details deriving from the ComInterface component class. We created a pure virtual function called execute() which takes in a command-argument pair from ComInterface and executes it; it returns a bool value depending on the success of the execution. We then call the boolean connector() function which takes in the connector string following the command and returns true if the command chain should continue or false if execution of the commands should stop.

SelfDefined : public Commands - Our understanding is that we will need to write functions for only some of the commands. SelfDefined will hold theses self-written commands.

PassThrough : public Commands - From what we understand so far, commands which we will not have to self-define are passed through using execvp/fork/waitpid. These commands will be called from within PassThrough. Our understanding is that these pass-through commands include ls, cd, etc.

Coding Strategy

At this time, our plan is to work together on all of the preliminary programming and divide up the work if/when we feel it is more advantageous to work separately. We have found in lab that we do our best work when we can discuss and bounce ideas off of each other as we are working on one piece of code together.

If we feel very confident after a few team meetings and group coding sessions, we will divide the work as we see fit at that time. Again, at this time we expect to do most, if not all, of the coding together.

Roadblocks

We were advised to keep our design scheme and UML simple. Our expectation is that a simple, elegant design will be better than an overly-detailed, convoluted plan. We are concerned, however, that our design could potentially be overly simple. If we need to create more classes, we will do so as the need arises. If anything, we would expect to use decorators rather than restructuring the entire design.

We are also somewhat worried about our C++ knowledge, but we will come to office hours as needed.