Assignment 2: Basic Command Shell

November 15, 2018

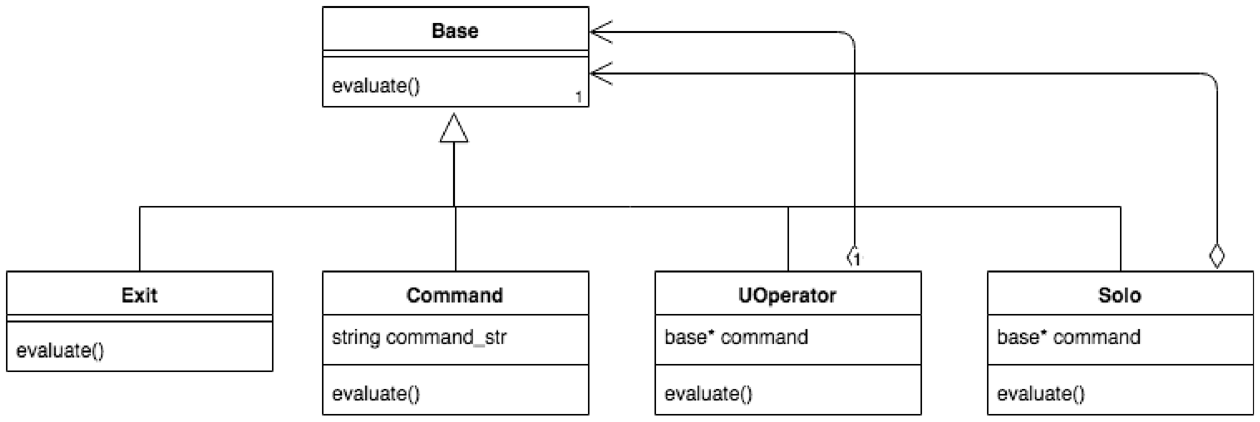
Fall 2018

By: Christopher Gentibano & Jason Jewell

Introduction

Our project design is composed of 5 different classes. We have the Base class, and two composites which would be the UOperator and the Solo class. While we have two leaves, the Exit class, and Command class. The composites will be taking in commands in the correct order so that the user will be able to run the requested commands as if they are using bash. While the two leaves will be handling the requests of the user looking to exit the program when requested. Lastly in our main.cpp we were able to parse through the line/lines typed by the user so that the correct commands will run.

Diagram:



Class Groups:

Base:

Our Base class will be composed of two leaves and one composite. The first leaf will be “Exit”, which will be created once a user requests to exit the program. Next leaf will be the Command which will be used to create Command objects for UOperator and Solo. UOperator will hold the base features to implement functionaility for Solo to run the commands. Solo class will take in the vector of string commands for execvp to run.

UOperator:

Our UOperator class plays as a composite. This composite will be used to implement any of the commands that are inputted by the user, which are then sent to Solo to be parsed through. This composite holds the virtual evaluate function for the Solo class to fulfill. The UOperator also has a constructor that takes in one Command object to assign a command.

Solo:

The solo class will handle executing the command, and takes in each of the commands for execvp to run. In our main we will have the command entered by the user parsed, so that when Solo object is created, the correct commands will be ran.

Command:

The command class plays as a composite for the Exit class. In order for the user to exit the program, they would need to enter in an exit command. The command class also takes in a string in the constructor to assign a specific string command to the object. This class also contains an evaluate function which returns a string.

Exit:

The exit class is created when the user wants to exit from the RShell. Once they type in Exit, we will call the Exit class to create the object, and runs “exit(0)” to end.

Coding Strategy:

The code has been broken up in a few different parts. We worked together to design the program as a whole, planning how we are going to organize our composite pattern, and what each of the classes will contain. Jason will be working on the parsing of each of the commands provided, storing these into Stacks & Queue’s to have a vector of strings provided to the Solo class, creating the Base & plans for composites. While Chris will be working on creating the subclasses, implementing the composites, creating the exit class, and creating the execvp system to run each of the commands. Lastly we needed to work together to create the design for main on how we wanted to organize the program.

<u>Roadblocks:</u>

The roadblocks that we foresee will be found during the coding phase of this project. We believe that we will see errors or gaps in logic in our current design while we are building this project. The solution to this is going back to the drawing board and reworking the class design to fit the needs that we have. The other roadblock that we believe we will run into is disagreement between each other for how the class design should be and/or how to code the solution. The solution to this would be compromising and discussing the pros/cons to each solution and coming to an agreement that both of us can work with.