Assignment 1: Design

11/01
Quarter 1
2018

Alex Cui
Jacob Halvorson

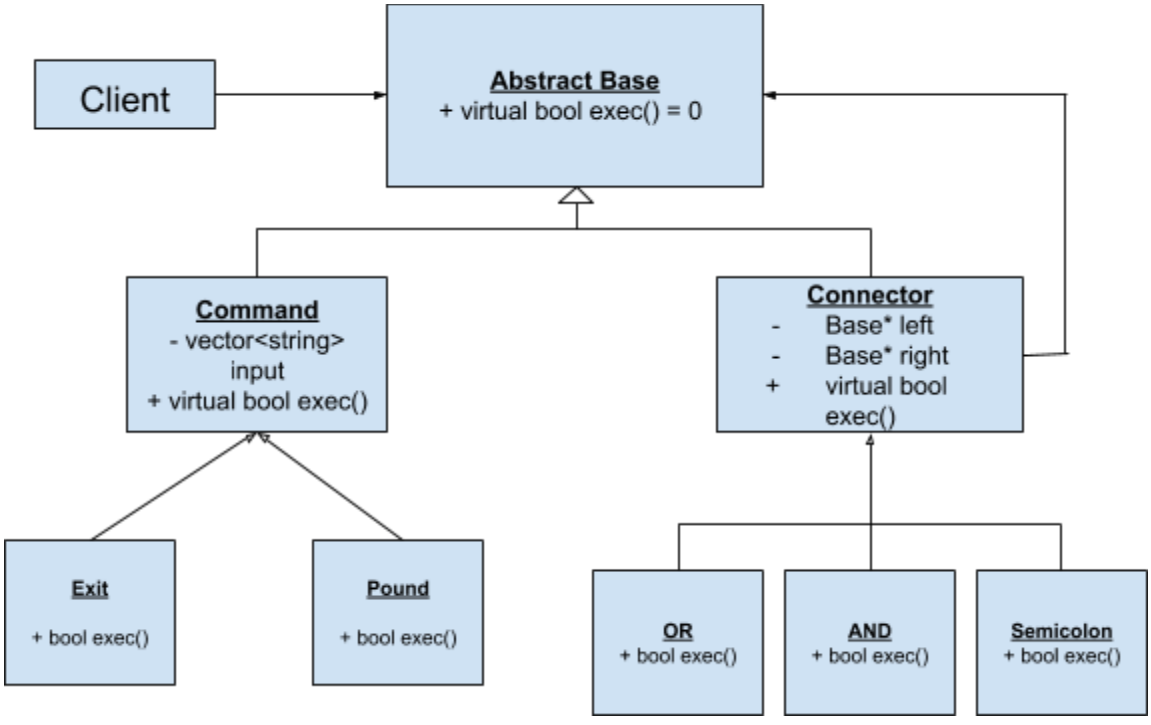
Introduction:

In the command shell, the user can type only two types of inputs: commands and connectors. The command will do an actual task, while the connectors are, in a sense, conditional statements that will decide whether the next command is executed. So, we create two classes: the Command class and the Connector class that each inherit from an abstract base class simply called Base.

The Command class will execute all the built-in commands through execvp, but we will also make two additional classes that inherit from Command: Exit and Pound. Exit will simply exit the program and Pound make it so that everything after a # is considered a comment.

The Connector class consists of three valid connectors: "||" and "&&" and ";". So we will respectively make three classes called OR, AND, and Semicolon, all of which inherit from the Connector class. The OR connector makes it so that the next command executes only if the first fails; the AND connectors makes it so that the next command executes only if the first succeeds; and the Semicolon connector makes it so that the next command is simply executed normally.

Diagram:



Classes/Class Groups

Base:

Has a pure virtual function exec() that will return a bool. We made an exec() function pure virtual because everything the user types must execute some sort of action, so everything class that inherits from this base must accomplish something. We chose to return a bool value because with connectors, the next command's execution may depend on whether the previous command was completed, or in terms of our diagram, returns true/false.

Class Group- Command:

We chose to have a vector of string so that we have a container to hold the user input. The command composite class will read in the user input one by one and push it back into the vector, and then execute them by utilizing the fork, execvp, and waitpid system commands (all while paying attention to connectors).

Exit:

A special built-in command that simply exits the program when executed

Pound:

A command that will cause the rest of the input after the # symbol to be considered a comment when executed.

Class group- Connector:

The connector class serves as the composite class for the OR, AND, and Semicolon classes. We chose to have a left and right Base pointer variable so that we can easily check the left command, see if it returned true/false, and then execute the right command depending on the result. The children of this class only differ in the logic used to execute commands.

OR:

Executes the next command only if the left hand side fails to execute.

AND:

Executes the next command only if the left hand side succeeds.

Semicolon:

Executes the next command regardless of success or failure.

Coding Strategy

After making the abstract base class, my partner will attempt the Connector class and its subtree, and I will attempt the Command class and its subtree. Due to the complexity of the Command class, as it involves bash command executables which we currently do not understand, we will likely work on it together in person. We will unit test each operation as we go along using gtest. If either of us have issues with coding, we will refer to one another for help. Afterwards, we will make sure it all compiles, and ensure its functionality by writing multiple integration test cases.

Roadblocks

- Scheduling conflicts:

  We will have to communicate through text and figure out the best times we can meet throughout the week.

- Other homework requirements:

  Due to a heavy course load with multiple assignments and labs due for other classes we will need to be sure to complete our assignments early and not procrastinate.

- Knowledge of system commands:

  To learn more about the Linux environment and system commands we will read more about it online and watch the provided tutorials (commands such as execvp, syscall fork, waitpid).