Assignment 1: Design

April 26, 2017

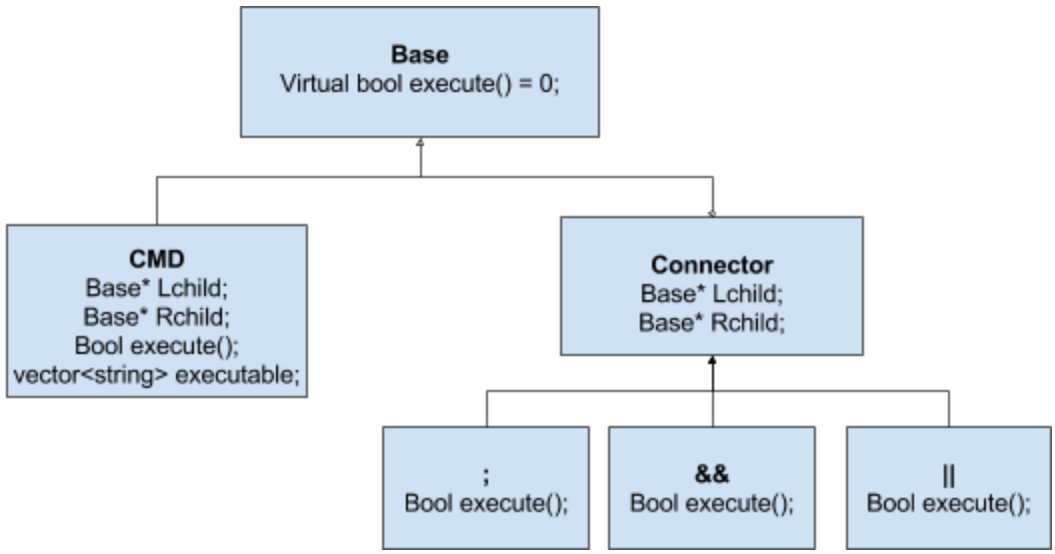
Spring 2017

Jerry Lee

Natasha Orie

Introduction:
In this assignment, we will be writing a command shell called rshell using C++.
Our shell will print a command prompt and read in a command on a line.
It should be able to take in multiple commands at once using connectors.
The program is structured to parse a string of user input and create a binary tree of command (CMD) and connector objects, which is then used to execute the user's commands.

Diagram:



Classes/Class groups:
Base:
The composite abstract class for all objects in the bash.
All classes inherit from this base class and each inheriting class must implement its own version of execute function.

CMD:
Subclass of class Base with the function and instance of inputs (strings).
Includes a left and right child pointer for tree implementation. The pointers are of Base class so that the tree can be composed of CMD and Connector objects.
The execute function in the CMD class with utilize syscalls (fork, execvp, waitpid) to perform the specified commands.

Connector:
Subclass of Base; it is the composite class that defines the left and right child Base pointers for its children, which are the Semicolon, And, and Or classes.

The pointers are derived in the Composite class so they do not have to be redefined in each child class.
Like the pointers in CMD, leftChild and rightChild are of class Base so as to make the construction of a tree possible.

;, &&, ||
Semicolon, And, and Or are the derived classes of the Connector class.
Each connector has its own version of the execute() function; for example the Or class will execute its right child only if the left child fails. And's execute function will execute its right child only if the left child succeeds. And so on.

Coding Strategy:
With the UML diagram, all classes and their implementations have been mapped out. We can both implement the base class together. From there, we work down separate branches of our diagram, i.e. one person works on the CMD class while the other implements the Connector class and its children. Once finished, we can much more easily integrate the completed segments together.

Roadblocks:
Potential issues may include incorrect implementations due to details not considered in the initial planning phase with the UML diagram. Integrating segments together may also prove to be an issue and may require some adjusting when done.