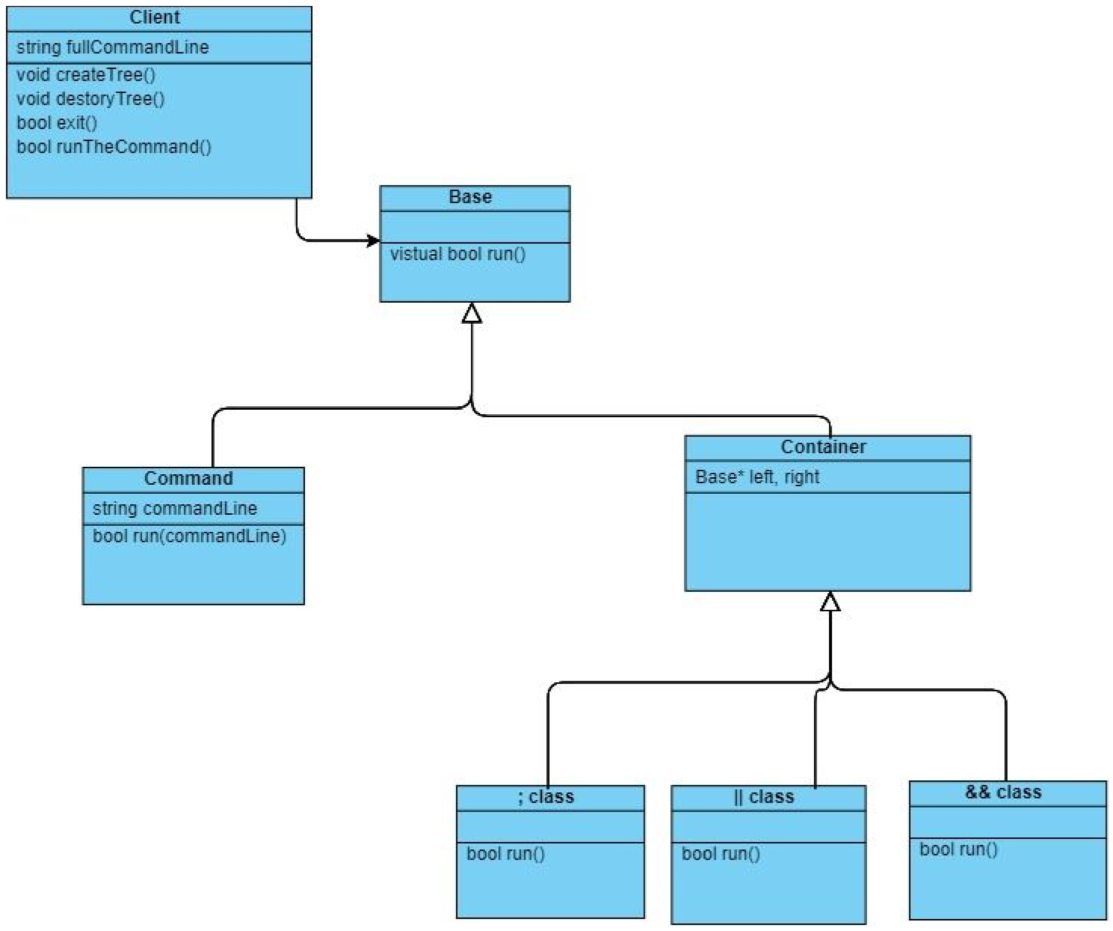
Assignment 1: Design

Fall, 2018, 10/31/2018

Chi Chiu Tsang
Ka Keung Cheung

Introduction: In this assignment, we are building a command shell that can read the user commend and run it according to the connecter. One major difference between a normal command shell and the one we are building is we can have connector between command and the shell will decide to run the command or not depend on the connector. We will implement 3 connector that can be used to connect command together. The first connector is ';', it will run the command on the left and the right. The second connector is "||", it will run the command on the right if and only if the left command fail to run. The last connector is "&&", it will run the command on the right if and only if the left command success to run.

Diagram:



Client
string fullCommandLine
void createTree()
void destoryTree()
bool exit()
bool runTheCommand()
Base
vistual bool run()
Command
string commandLine
bool run(commandLine)
Container
Base* left, right
; class
bool run()
|| class
bool run()
&& class
bool run()

Class:

- Base: visual function, provide the base for all the other function.

- Command: it store a string of command, it has a function to run the command. It will use fork, execvp, and waitpid to run the command. After running the command, it will return boolean for does the run success or not.

- connector: parent of the 3 connectors classes, which is the parent of the decorator class. Not vistual so that can save some time when building the decorator class.

- ALL(;) class: use to examine both left and right command. No matter if left child is success or fail the second child will always run. There can be only left child. Return right child as boolean (0 or 1 depend if right child run fail or success).

- OR(||) class: use to examine the left child command first, if it fail then we continue to examine the right child command and return right child as boolean (0 or 1 depend if right child fail or success). But if the left child success, then we skip the right child command and return boolean (0).

- AND(&&) class: use to examine the left child command first, if it success then we continue to examine the right child command and return boolean (0 or 1 depend if right child fail or success). But if the left child fail, then we skip the right child command and return boolean (0).

- client class: Constructor take in a string of full command line by getline in the main function. It only has one function that can be run from the outside and that function will run the whole line of command. This function return bool to indicate is the user input exit. In the run function, it first check is the string pass in is "exit". If it is, return false for the function and the caller will know to exit the shell. If it is not, it will read the command line string and cut the string according to the connector. Then it will build tree with using the node "Command" and "connector". After the tree was build, it will run the command at top. After all the command is run, it will delete the tree to prevent memory leak.

Coding Strategy:

Chi Chiu Tsang: Client, Base, Command, Function that call the Client

Ka Keung Cheung: Container, Connectors, Command

We will integrate the assignment by meeting regularly outside of class, and we can also use github to assemble the segments together online.

Roadblocks:

- Never use folk, execvp, and waitpid before and that could take long time to understand how to use them. It can be overcome with continue testing with it.
- Reading the command from the user input need to be careful or the tree would not built correctly. Can be prevent problem by unit testing the tree.
- Problem in the parent class could lead to failure of the children class. Can caught the problem using unit test.