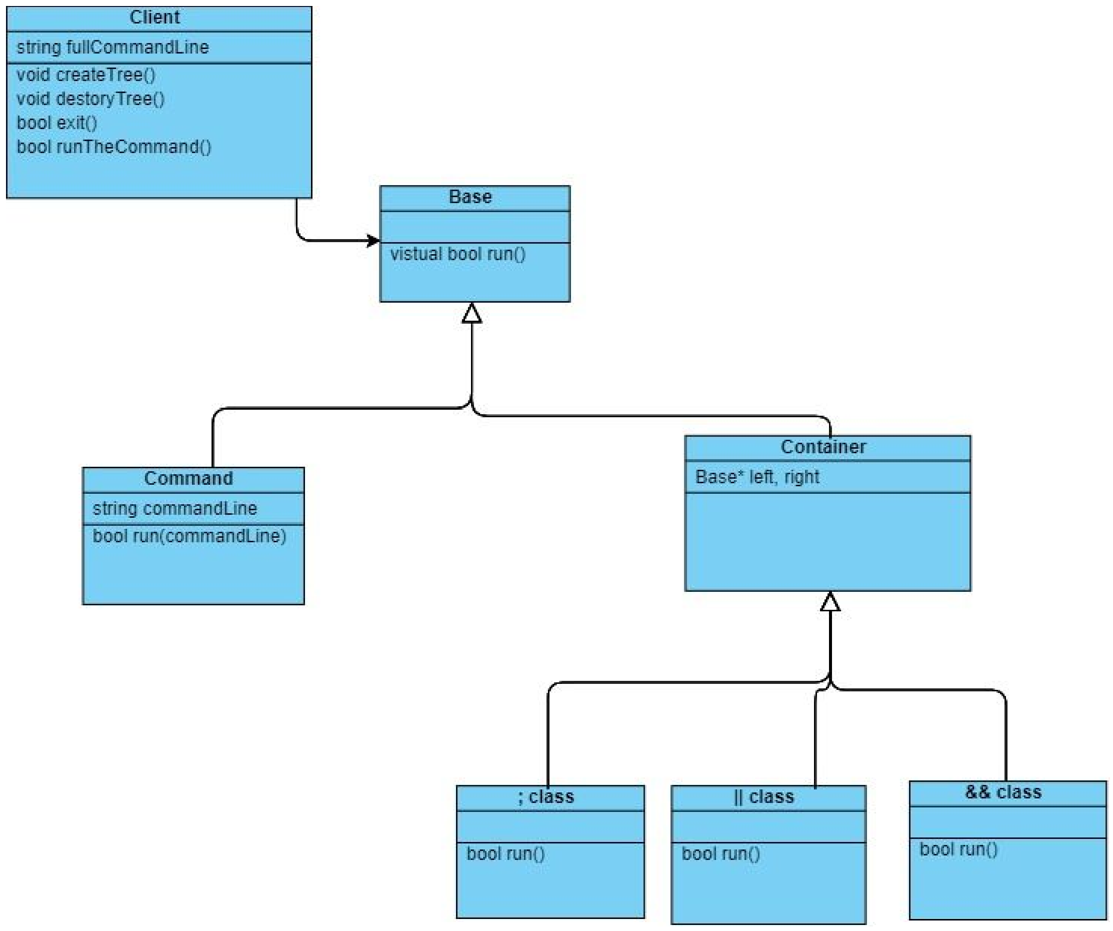
Assignment 1: Design

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 has 4 major function. They are createTree(), destroyTree(), checkExit(), and runTheCommand().
    - createTree(): It will take the full command line that pass in to the class and cut it according to the connector. Then it will build a tree structure that connect all the command. All the tree node is command and connector is parent of either command or another connector.
    - destroyTree(): It will be call after all the command has ran. It was use to clear memory that was declare when building the tree.
    - checkExit(): Use to check is the full command line is "exit". If it is, return true, else, return false. It is use for response to the outer function to indicate when to end the program.
    - runTheCommand(): It was used to run at the top of the tree after the tree has created. When run the top of the node, it will start to run the command at the

bottom of the tree and travel to the top. The connector would make the decision to either run the command of their children or not.

Extra function:

- runShell(): It was used to call all the function in the client class. It will ask the user to enter a line of command that can be connected with connector. Then it will use the line of command to build a client object. It will first check using the client function is the command is exit. If the command is exit and client would return a boolean and the runShell() function would receive it and exit the function. If it is a normal command, the runShell() function would call the client to build a tree and run the command. After all the command has ran, it will call destroy tree to prevent memory leak. The function would not exit unless the command is exit.

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.