Assignment 3: Design Document

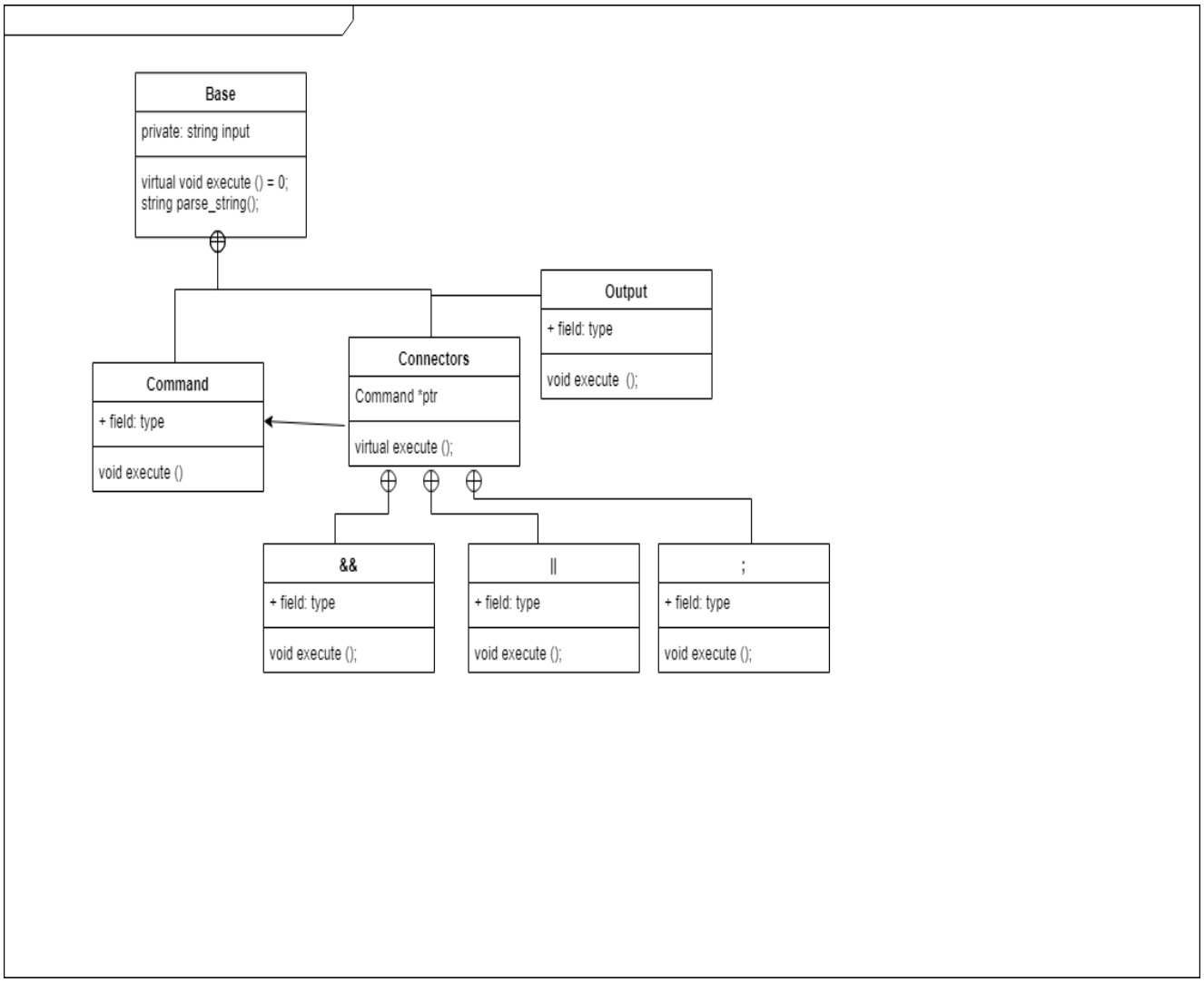
November 29th 2018
Fall Quarter
Authors:

David Huecker
Andres Rodriguez

Introduction:

In this assignment our team built a basic command shell in C++. The name of the program is rshell and it will start by printing the command prompt, '$'. Next it will read in a line of command(s) (and connector(s)) from the user). Finally the program will execute the command(s) using fork, execvp, and wait output. To accomplish these tasks our team will used the composite strategy, along with other coding techniques. Now rshell is will take in any old command along with the keyword test and []. This will signal rshell to test the code following the keyword or inside the brackets. The results will be stored as a boolean value and shown to the user as either TRUE or FALSE.

UML Diagram:


Base
private: string input
virtual void execute () = 0;
string parse_string();
Output
+ field: type
void execute ();
Command
+ field: type
void execute ()
Connectors
Command *ptr
virtual execute ();
&&
+ field: type
void execute ();
||
+ field: type
void execute ();
;
+ field: type
void execute ();

Classes/Class Groups:

1) Base class will have a private data member called user_string and will have public functions Base(), String Parse_User_String(), and virtual void execute()=0; The function execute is pure virtual which means every child class must include the same function.

2) The first group of classes is Command, Connectors, and Output. The Command and Output classes implements the void execute() function directly. However the Connectors class that function to its children.

3) The subclasses of Connector are the second group of classes. These classes will implement the void execute() function. Depending on which connector the user inputed and in which order, will determine the version of execute() to call in this group.

4) The Class parser was add to the project, so we could correctly store the correct input from the user. This class also receives the token associated with a connector. Parser is crucial to this project, since improper input will lead to many bugs.

5) The test command responsibilities were split under the Command class and Pasers class. The main of the project will also be used to help handle some of the data. The function test() will return true or false to the user depending on the data entered. The test command is now its own class which includes exe() function.

Coding Strategy:

For this project the work was divided between my partner and I. David was incharge of creating the Base, Command, and Parser classes. My partner will be in charge of creating the Connector class, and all of its sub children, and main. Since the project is ongoing we have both are in charge of testing the entire project, and making corrections to the test command. Currently google test directory is working and more tests are being added.

Roadblocks:

The major roadblock for this assignment was finding a way to properly use the connectors class. A way to determine the logical value of the command is still needed as well as the comparison function for each connector. Part of the problem is that the child process are erased after running, which means their isn't a way to change any data in the child. Continued research is needed to fix this problem.

*update for roadblocks 11/21/18

The project's connectors now work after altering the code. Next step to overcome is how to use the connector's class instead of main for handling multiple cmds. Continued research and testing is still needed.

*update for roadblocks 11/29/18

The project is currently trying to figure out how to correctly handle the test command properly. The usage of stat() and the macros associated with it, need more research before being added into the project. Rshell still runs with a few errors.

*update for roadblocks 11/30/18

The project now currently runs the test command and [] operation correctly. Also the flags are turned on and are running as well. The precedence operator still needs work and research before implementing it to rshell.