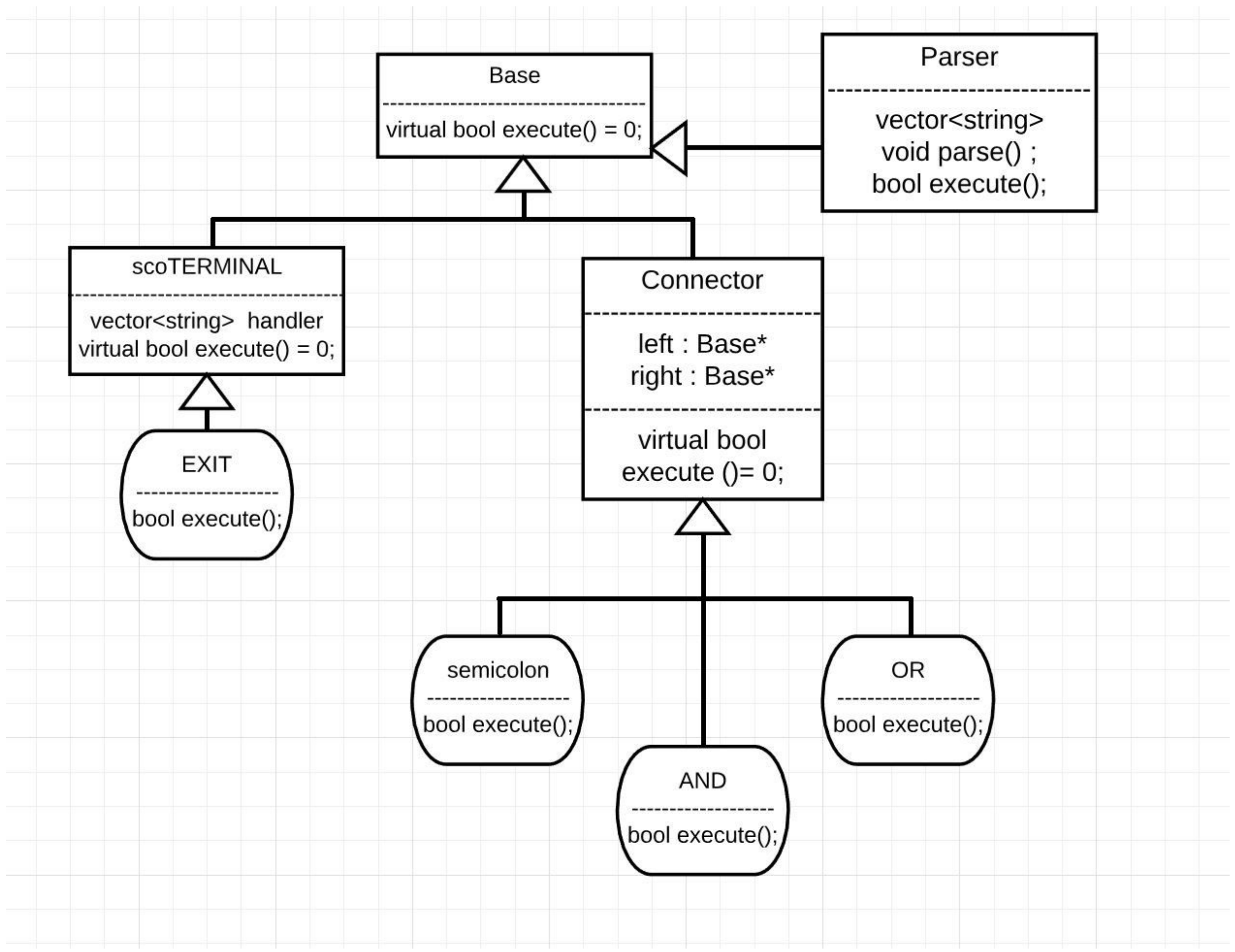
SCOTSHELL - A Better Alternative

TITLE: ASSIGNMENT 1: Design
DATE: 11/1/18
QUARTER: FALL 2018
YEAR: 2018
AUTHORS: Dylan Han, Alexander Yee

INTRODUCTION
Scottshell is the project of Dylan Han and Alex Yee for CS100 Fall 2018. Our primary objective is to make and develop a basic Linux shell that takes in a built-in commands along with some extras such as AND, OR, and semicolon. Using a composite pattern approach, we will create an inheritance hierarchy that will allow our composite classes functionality to be inherited in this manner.

DIAGRAM



***Quick Note: This is going to be our basic diagram based off of the composite pattern. We intend on extended this**

CLASSES/CLASS GROUPS
Super Class(Interface) : **Base** - Our abstract class and will contain :

- Virtual bool exec();
- This superclass will combine all of our objects

Derived Class(Composite) : **Parser** - Will parse command line for user inputted strings

- Bool exec();
- We will be using the built in strtok to tokenize the command line and break up our string

Derived Class(Composite) : **ScottTerminal** - Stores what the parser handles as strings

- Bool exec();
- This composite will have a child named exit which is responsible for ending the terminal's process when exit is inputted

Derived Class (Composite): **Connector** - This will be an abstract class that will be used for our || , && and ;

- Bool exec();
- This composite will have 3 derived classes/leaves
- One class for each of ||, && or ; and functionality will be defined in these classes according to specs

CODING STRATEGY
We will start the development process of this assignment together by doing research on the key concepts that we will need to actually build a shell. Our first task after having a good understand on how our shell will work, is setting up our header and .cpp files, adding function declarations and necessary inclusions and inclusion guards. Once we have our functions declared, we will continue researching and discussing functionality and the tasks that we need to complete in order to build momentum for our project. It is important for us to have a concrete understanding on how to actually build our shell and make it work before we start appointing tasks. We need to fully understand the overlying big picture of this assignment so that we assign tasks based off of the group members strengths and weaknesses while still maintain efficiency. We plan for our coding strategy to be a

composite pattern approach that is backed by using a Kanban style development process. We want to create a unary base function that all base classes inherit. Using github, we will create different branches and work on each piece by piece meanwhile systematically pushing and pulling what we need. In terms of the division of the work, we want to focus on creating the terminal with total working external command functionality. Then we will create the respective connectors and parsing algorithms.

ROADBLOCKS:
There is no doubt that one of the bigger problems that we will encounter while doing this collaborative project is the collaboration aspect itself. Not in the sense that working with each other would be difficult but rather the syncing of progress during development. It is crucial that our group codes this project incrementally, while writing code that would coincide with each others' work. Because of this, we intend to work on branches and merge to our master branch, only when we have both looked over the code and agree that whatever we have is ready to be pushed. Also, suppose that the work is divided by the base class being written by one member and the derived classes being written by the other member, one does not complete a fully function portion of their part, the other member would have to bear the burden of waiting for the other person to finish, hindering the progress and rate in which the project is being completed. Maintaining consistent flow is intrical to completing this project, which is why we are adopting the Kanban approach to maintain this productive workflow. What makes Kanban a good fit for our design is being able to limit work in progress,which can help us avoid scenarios in which a certain partner pushes more than what the other partner can handle. This systematic pulling of work only when group members are ready for it will overcome many of our sync problems and also seek to progressively answer the collaboration part of our project.