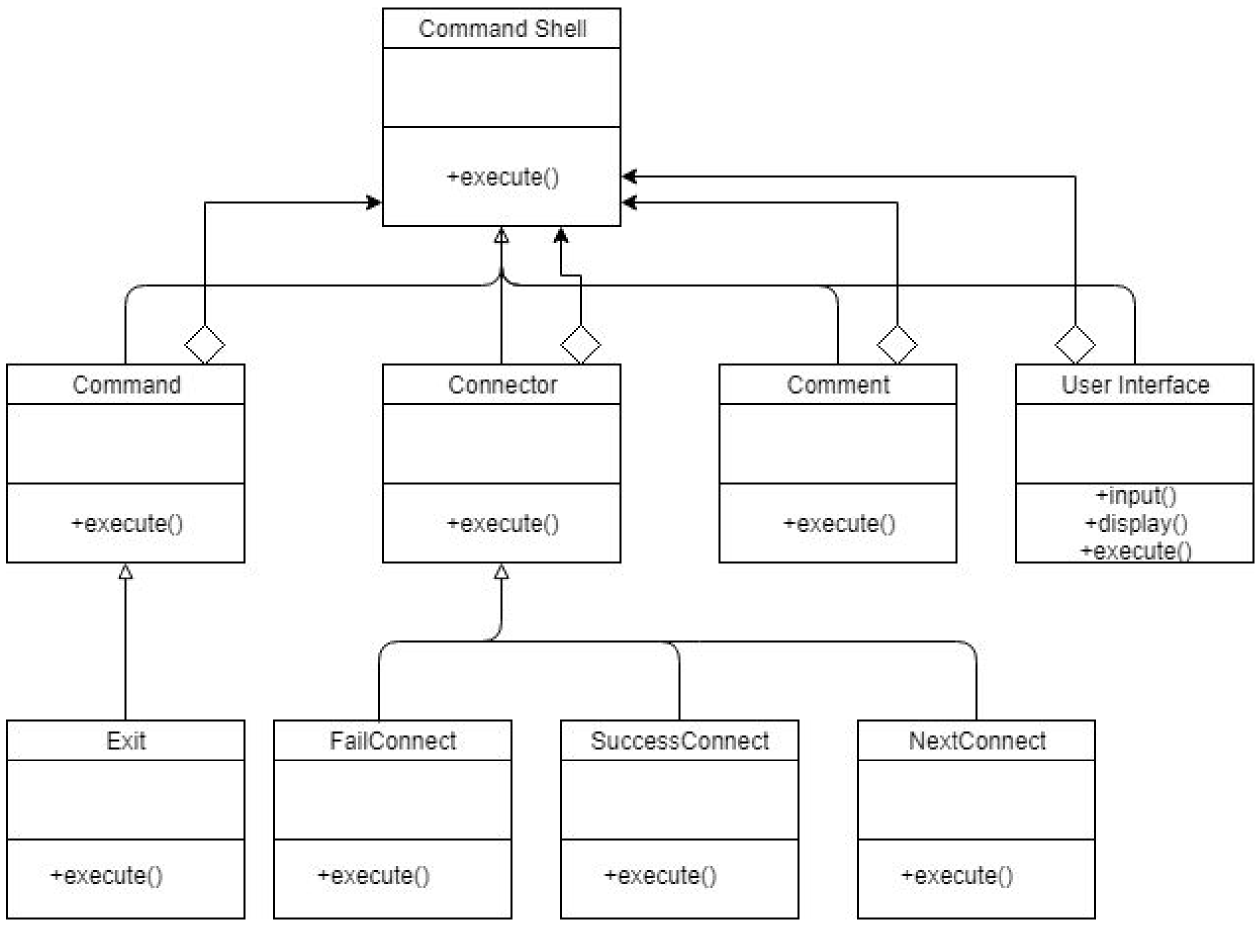
Assn 3 CommandShell

12/2/18

Tyler Ortiz
Hung Phan

Introduction: We are using a composite pattern to create a command shell in C++. It will print a command prompt , then read in a line of commands from standard input then execute those commands. Functionality has been added to include the test command

Design:



Classes/Class Groups

- Base Class: CommandShell
    - Pure virtual class to act as pointer for the rest of the classes
- Composite Classes: all children of CommandShell base class
    - Command: will be used to execute commands input by the user
    - Connector: will be used to distinguish between connectors and help execute commands
    - Comment: will be used to distinguish what is a comment in the user input and skip execution of the rest of the line
- Leaf Classes:
    - Exit: child of command, execute will exit the command prompt
    - TestCommand: child of command, executes the test command to see if the designated path exists or not
    - NextConnect: child of Connector, executes next command on the line
    - SuccessConnect: child of Connector, executes next command on the line if previous command was successful
    - FailConnect: child of Connector, executes next command on the line if previous command failed
- UserInterface: output the command prompt, take in user input to be parsed and executed as commands later
- Parser: parses the input string, seperates it by spaces, creates commands and executes them

Coding Strategy:

- Tyler: Connector and its subclasses, CommandShell class, Parser class
- Hung: Command and its subclasses, Comment, User Interface
- Integrating assignment together: using github's push/pull to sync local repositories

Roadblocks:

- Parsing the input string: Potentially parse through the string by looping through it and everytime there is a space character, store the preceding string in a dynamic array like a Vector and determine from there what kind of string it represents
- Unfamiliarity with C standard library and command prompt programming: extensive googling and spending time working with the prompts and commands. Starting work on the assignment early
- Testing: coming up with good and extensive tests throughout the process of coding to minimize the amount of bugs in the project
- Implementing precedence operators: figuring out a way to implement the precedence operators within the current scope of the program

Implementing comment functionality

Current known bugs:

- exit must be called an extra time for every failed command
- Precedence Operators have not been implemented
- Comment functionality has not been implemented