Assignment 1: Design
January 27, 2016
Winter 2016
Paul Rodriguez & Donald Weiss
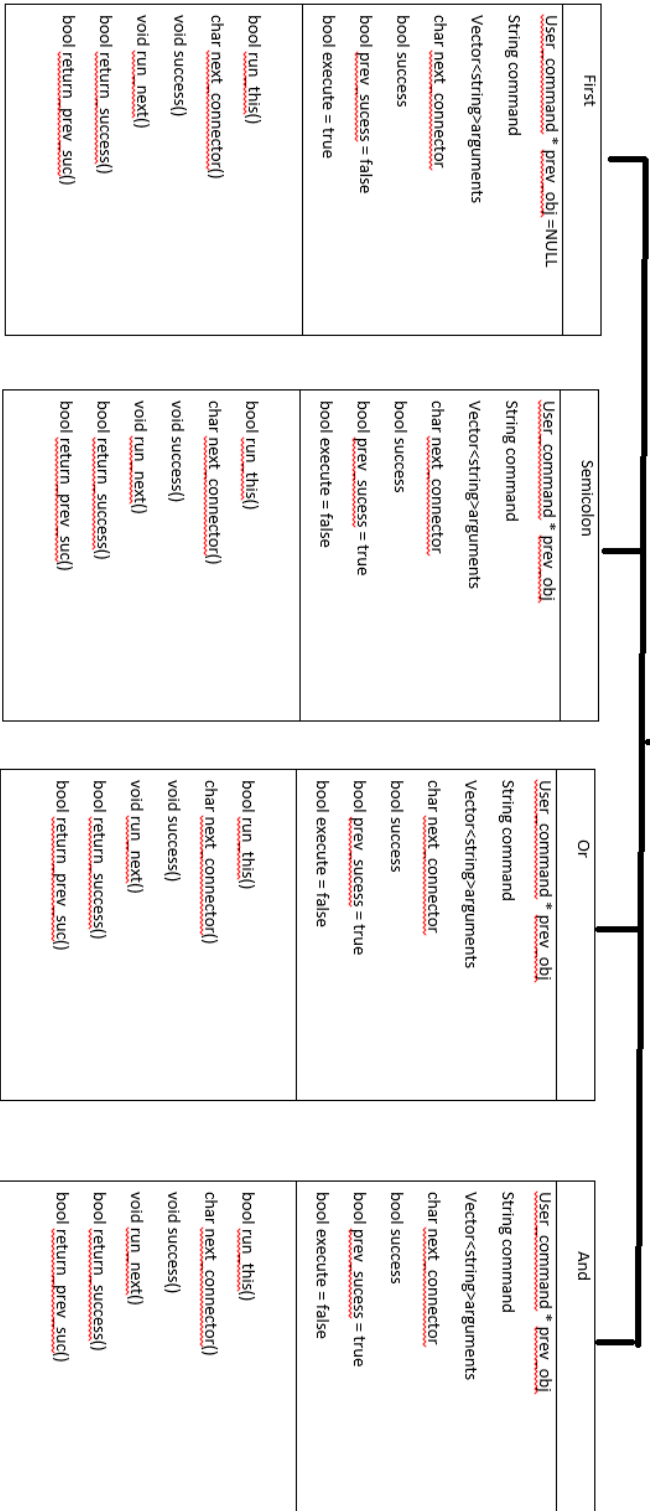
● Introduction:

- We will have a function called parse() that will read in the entire command and create tokens.
- Then we will have a function that will create and object based on the commands that are passed in.
- These objects will be held in a vector. Vector<Objects*> v
- The vector will then be passed in to our execution function. Our function will determine what kind of object it is (based on the command) and go into an appropriate branch for execution.
- This will be repeated until v.end()

● Diagram:

User

User_command

NO DATA

virtual bool run_this() = 0;
virtual char next_connector() = 0;
virtual void success() = 0;
virtual void run_next() = 0;
virtual bool return_success() = 0;
virtual bool return_prev_suc() = 0

First

User_command * prev_obj =NULL
String command
Vector<string>arguments
char next_connector
bool success
bool prev_sucess = false
bool execute = true

bool run_this()
char next_connector()
void success()
void run_next()
bool return_success()
bool return_prev_suc()

Semicolon

User_command * prev_obj
String command
Vector<string>arguments
char next_connector
bool success
bool prev_sucess = true
bool execute = false

bool run_this()
char next_connector()
void success()
void run_next()
bool return_success()
bool return_prev_suc()

Or

User_command * prev_obj
String command
Vector<string>arguments
char next_connector
bool success
bool prev_sucess = true
bool execute = false

bool run_this()
char next_connector()
void success()
void run_next()
bool return_success()
bool return_prev_suc()

And

User_command * prev_obj
String command
Vector<string>arguments
char next_connector
bool success
bool prev_sucess = true
bool execute = false

bool run_this()
char next_connector()
void success()
void run_next()
bool return_success()
bool return_prev_suc()

● Classes/Class Groups:

Object class will be the base class
　　　　(the following will be pure virtual functions)
　　　　- bool run_this() = whether the command will be run
　　　　- char next_connector() = sets the connector for the following command, if any
　　　　- void success() = sets whether the command executed
　　　　- void run_next() = depending on object and success, will set next object to true or false
　　　　- bool return_success() = will return T/F object executed
　　　　- bool return_prev_suc() = will return T/F if previous object executed

First object
　　　　- run_this() set to true
　　　　- return_prev_suc() = false

and object
　　　　- will only run if return_prev_success == true

or object
　　　　- will only run if return_prev_success == false

semicolon object
　　　　- run_this() = true

● Coding Strategy:

- Create objects classes
- create parser
- create object-creator function
- create execution function

The work will be done together at the same speeds. We will only continue when both codes are thoroughly revised and agreed upon. Then we will continue on with the rest of the implementation.

● Roadblocks:

The hardest part of this will be creating the execute function because of all the possible combinations that can arise. Another difficulty will be if any necessary new functionality that will be added will not work with the already created objects, extent modification may be required