Pre-Processing Phase

1. converting String → LinkedList
2. converting a String command into ICommand token

Two parse() functions splits up functionality, but necessary?

String command
Ex: fd sum 30 30

Model
void executeCommand(String command)

Parser
ICommand parse(String command)
Return parse(params)
//params is LinkedList representation of the command

Parser
ICommand parse(LinkedList<String> commandStream)
Return main;
//main is of type CommandList, which gets executed in model

Process Phase

The code is very modularized. No huge functions.

ProcessParameters and processParameter are simple functions

parseVariable, parseList
parseList, parseNestedFunction
Hard because you have pseudo-recursion.

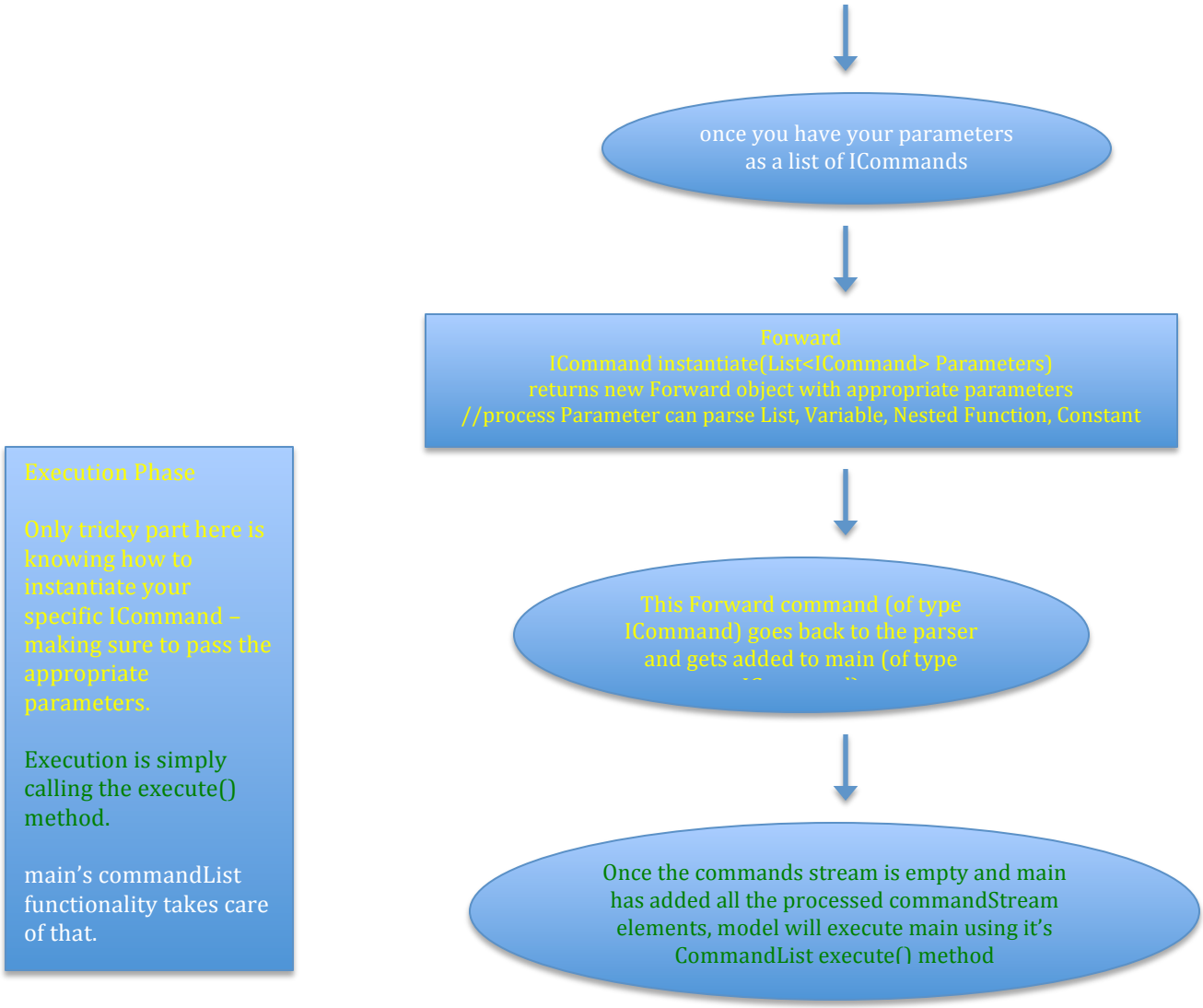
myParameters.add(myParser.parse(commandStream));

LinkedList implementation of commandStream adds elegance because you can easily take the head of a commandStream and have the rest of the commands

get the initializer of the keyword and call its build()

Abstract Initializer
ICommand build(LinkedList<String> commandStream)
convert commandStream into a LinkedList of ICommands called parameters
//done by calling processParameters on commandStream

Abstract Initializer
processParameters (LinkedList<String> commandStream)
iterates numArgs times, calling a processParameter on ComamndStream
//process Parameter can parse List, Variable, Nested Function, Constant


once you have your parameters
as a list of ICommands
Forward
ICommand instantiate(List<ICommand> Parameters)
returns new Forward object with appropriate parameters
//process Parameter can parse List, Variable, Nested Function, Constant
Execution Phase
Only tricky part here is knowing how to instantiate your specific ICommand – making sure to pass the appropriate parameters.
Execution is simply calling the execute() method.
main's commandList functionality takes care of that.
This Forward command (of type ICommand) goes back to the parser and gets added to main (of type
Once the commands stream is empty and main has added all the processed commandStream elements, model will execute main using it's CommandList execute() method