**SLogo Design Document**

**John Godbey, Kevin Jian, Carlos Reyes, Vincent Wang**

**Design Goals**

Describe the goals of your project's design (i.e., what you want to make flexible and what assumptions you may need to make) by breaking it into *modules*. A module is a concept in your program that may be represented by a single class or a group of classes related in some standard way, such as set of subclasses, that has a purpose with regards to the program's functionality and collaborates with other modules.

The goal of our project design is to have the entire codebase as flexible and extensible as possible- especially the command classes. We plan on having two main components, the model and the view. In order to keep the workflow organized and streamlined and extensible, we think that this separation will be most effective. The model will take care of all the states of the turtle and coordinates of the trails, and the view will take care of actually drawing the representation onto the GUI/screen. This set up does not preclude the possibility of swapping out either the model or the view. If someone were to follow the constraints of our API, then they would be able to do just that.

**Primary Classes and Methods**

Describe the program's core architecture (focus on behavior not state), including pictures of a UML diagram to describe the Model and "screen shots" of your intended View interface

Syntax hierarchy: Every command possible will be a class of its own (e.g. FORWARD, BACK, SET, FOR, IF). These classes will be subclasses of classes that represent types of syntax (e.g. Boolean, Loops, Math, etc.)

Our UML file is in the same folder as this file.

**Example code (this is really important in helping others understand how to use your API)**

Show actual code that implements the following use case: "The user types 'fd 50' in the command window, and sees the turtle move in the display window leaving a trail." It should be clear from this code which objects are responsible for completing each part of the task, but you do not have to implement the called functions.

When the program is initialized, it will create a Turtle and Parser object. To the user will be displayed the command window and display window. When the user types in the following command:

fd 50, the View passes the string “fd 50” into the parser. The parser then parses the commands and generates the according command objects. These commands alter the Model values which calls the View class to draw the image of the turtle 50 pixels forward. If the Model sees that the pen is down then it also tells the View to draw the line of the trail as well. The Model stores the information of where the turtle is and has been.

**Explain some alternatives to your design, and why you choose the one you did**

One alternative to writing a class hierarchy to deal with all of the possible syntax is to write a switch case where every case is the syntax that is being processed. Two alternatives to using an abstract syntax tree are: one, using an concrete syntax tree, the other is simply looping through each command issued.

One alternative to having the Model pass information directly to the View and vice versa is to have a Turtle class in between that both the Model and the View can access. This could potentially be easier to implement because both could simply change and edit the Turtle and both would able to access its information to update the image/position. However, this is not ideal because then too much relies upon the turtle and there are more interactions than are necessary.

**List of each team member's role in the project and a breakdown of what each person is expected to work on.**

Carlos and Kevin will be working on creating the abstract syntax tree. He will also work on making a class hierarchy for the syntax that is accepted in logo.

Vincent and John will be working on creating the View part. This will be the drawing of the images and GUI aspect as well as dealing with receiving the input.

We chose these roles by using a random number generator.