

CS1322 Homework 8

Due: Saturday, April 2 @ 6:00 pm

LOGO

(Or, how I learned that parsing is not a hybrid of parsley and ginseng)

For homework 8, we're going to do some more advanced string processing. We're going to be doing what is called parsing - analyzing input so that it can be easily processed later. We're going to use an object-oriented hierarchy of classes to store information about an input file.

The files that we will be reading are in LOGO, which is a very simple programming language that allows people to draw pictures. We'll be using a limited form of it that makes it easier for you to work with. After parsing the input, you will then be able to draw the picture that it describes in a JPanel.

The idea of logo is that you have a pen starting at the top left corner of the screen, and "facing" to the right. You can issue commands like "forward 10", or "turn 90" that will move the pen and change its orientation. If the pen is down as it moves, it will draw a line where it moved.

The LOGO programming language is a series of commands, each on a different line. So, for example, the following program draws a red box that is 100 by 100 pixels:

```
COLOR 255 0 0
PEN DOWN
FORWARD 100
TURN -90
FORWARD 100
TURN -90
FORWARD 100
TURN -90
FORWARD 100
```

We're supplying some skeleton files that will get you started on the design, and give you hints as to how to best go about it. You are free to come up with your own design if you wish, as long as it fulfills the requirements, but using our classes will give you a greater opportunity for partial credit, and make it easier on your TA to grade. We also do think that our design is a good way to go about it, so it would at least do you good to understand what the files we give you do.

Program Requirements

You should have a driver class called "Logo" that, when run, reads the input from System.in, and brings up a JFrame that draws the picture that was described in the input. It will need to be able to be run like so:

```
C:\cs1322\hw08\> java Logo < my_logo_program.txt
```

Logo has the following commands (which are NOT case-sensitive):

```
JUMP x y
```

This moves the Pen (without drawing) to the specified $x y$ location on the screen.

```
COLOR r g b
```

This changes the color of the Pen to a new Color (r, g, b).

```
FORWARD dist
```

This moves the Pen forward $dist$ pixels, in the direction that the pen is currently facing. If the pen is down, it should draw a line in the current color as it goes. If the pen is up, it will not draw, but it will still change position.

```
TURN angle
```

This rotates the pen by $angle$ degrees. (An angle of 0 is straight right)

```
PEN [up|down]
```

Picks the pen up or puts it down.

Error Detection

You are also required to report errors in the input. The errors need to be printed out to the user, and your program needs to exit without throwing an exception or anything else. You should report the following errors:

- A blank line (every line of the file must have a command on it)
- A command with too few or too many parameters
- An unrecognized command
- If the Pen command receives anything other than up or down

When an error is printed, it must contain an error message describing what the error is, the line number, and the actual line that contains the error. For example, if the input is

```
COLOR 255 0 0
PEN DOWN
TURN 90 90
```

The program should print out

```
Error on line 3: TURN 90 90
    TURN only takes one argument
```

You are allowed to assume that the user will input valid numbers for the turn, forward, color, and jump commands. (Note that everything but COLOR can take in decimal numbers. TURN 1.5 is a valid command.) You are also allowed to assume that the integers given to COLOR are between 0 and 255.

Extra credit possibilities

We can give up to 10 points extra credit if you do something above and beyond what we required of you (amount will be based on what is done). An example would be implementing extra commands (looping, or variables or something else). If you DO implement something extra, write up a short description of it in an extraCredit.txt file with your submission to let us know what you did so we don't miss it. Also maybe supply an example input file that demonstrates your extra functionality.

What to turn in

- Logo.java (and all other java files needed to make your program run)
- extraCredit.txt (optional)

Appendix A: Our suggested design

We've supplied a few classes for you as a model for how you should do the parsing. Since you've already dealt with the nitty-gritty of drawing things and doing trigonometry to figure out x/y locations, we thought we'd give you the code to do that so that you could concentrate on the new stuff. However, definitely look at the code we've supplied and make sure you understand what it does before trying to write anything.

Some of the classes are complete, and some of them require you to fill in methods for them to work properly. You will also need to create several classes in order for the program to work. Places where you need to supply code are commented with //TODO lines.

Make *SURE* that you read the javadocs for the methods so that you understand what they do.

Classes we supplied

- **Instruction.java** - Abstract base class for Instructions. You will need to create concrete classes for each of the different Instructions. (PenInstruction.java, ForwardInstruction.java, etc.)
- **Logo.java** - Creates a JFrame with a LogoPanel inside of it, with Instructions read from System.in
- **LogoPanel.java** - Executes an ArrayList of Instructions that draws a LOGO program.
- **Pen.java** - Keeps track of all of the state you need for the Pen (where it is, where it's facing, whether it's up or down, etc.)
- **Parse.java** - Deals with turning input from System.in (or any arbitrary Scanner) into an ArrayList of Instructions.

The general idea

In Logo, a Parser object is created on System.in. When it calls parse, the parser reads in all of the lines of text it can, and then calls parse(String) on each line of text. That method creates an instance of an object that extends from Instruction. You will need to create those classes. You should do one class per Instruction type - PenInstruction, ForwardInstruction, etc. Each returned Instruction is added to an ArrayList, and then returned. That ArrayList is then passed into a new LogoPanel, which can then display the program.

Make sure you check for error conditions inside of `Parse#parse(String)`, and return null if there is an error. In `Parse#parse()`, you should check to see if `parse(String)` returned null, and if it did, add the line number on to the error message, and then return null.

Appendix B: results from sample inputs

When you run Logo on the inputs `error[1-5].txt`, (files with problems in them), you should get the following outputs:

```
C:\cs1322\hw08\gte941n\test>java Logo < error1.txt
Error on line 30: color 220 0
    COLOR takes three arguments
```

```
C:\cs1322\hw08\gte941n\test>java Logo < error2.txt
Error on line 8: forward 300 200
    FORWARD only takes one argument
```

```
C:\cs1322\hw08\gte941n\test>java Logo < error3.txt
Error on line 20: pen sideways
    PEN needs to be either UP or DOWN.
```

```
C:\cs1322\hw08\gte941n\test>java Logo < error4.txt
Error on line 20:
    blank lines are not allowed!
```

```
C:\cs1322\hw08\gte941n\test>java Logo < error5.txt
Error on line 22: bobsyouruncle 42
    Unknown command: [bobsyouruncle]
```

When you run the command "java Logo <squares.txt", you should get the following image:

