Objectives

- Team Final Project: SLogo Code Exploration

Common Refactoring Thought

- In Factory

```
private static final Random ourGenerator = new Random();

public int nextIntInRange(int min, int max) {
      return ourGenerator.nextInt(max - min + 1) + min;
}
```

Review

- What is the final project?
- What do you think you will learn from this project?

Team Project

- Benefits
  - **Gaining experience**
  - Writing a larger code base
  - Learn how to learn new things
  - Dealing with ambiguity
  - Working with a team on code
  - Depth on part of the project
- Limitations
  - Hard to get breadth on project

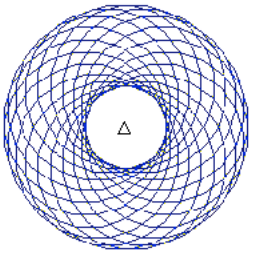
Review: Project Deliverables Timeline

Worth 20% of your course grade

Deliverable	Who	Weight	Due Date
Preparation	Individual	8%	Fri, 11/18
Preliminary Implementation, Demo	Team	20%	Wed, 11/30
Intermediate Implementation, Demo	Team	22%	Wed, 12/6
Final Implementation	Team	35%	You decide →latest Thursday 12/15, midnight
Analysis	Individual	15%	12/16, 5 p.m.

Review: SLogo Project Overview

- Goal: Create an IDE for simplified version of Logo
- Logo: programming language designed to teach children to program
  - Low floor, high ceiling



Code Base Exploration

5 minutes

- Create a picture of the given code base
  - What are the main classes? How are they related?
  - Any processes that you can enumerate?

Exploring Code Guidelines

- Try to get the big picture
  - Skip over parts that you don't understand—mark with a black box
- Iterate over the code
  - May not be immediately
  - Fill in the black box boxes, as necessary/appropriate
- Draw pictures if you can't keep the mental model

Deal with ambiguity

Review: Programming Language Syntax

- What does an identifier look like in Java?
- What does an assignment statement look like in Java?
- What can be on the left hand side?
- What can be on the right hand side?

- What does a multiplication look like?
- How do we evaluate arithmetic expressions?

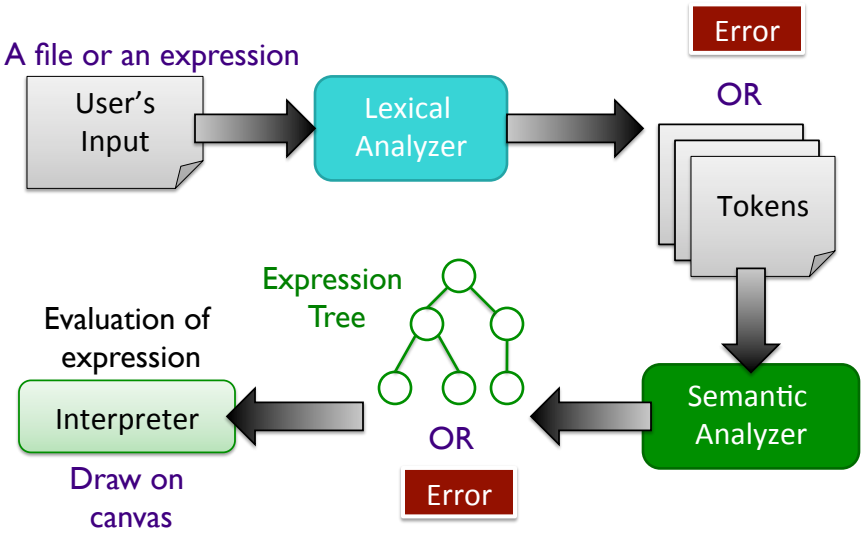
Programming Language Design

- Must be unambiguous
  - Programming Language defines a syntax and semantics
- Interpreting programming languages
  - Parse program into tokens
    - Example: x = 4*3; →

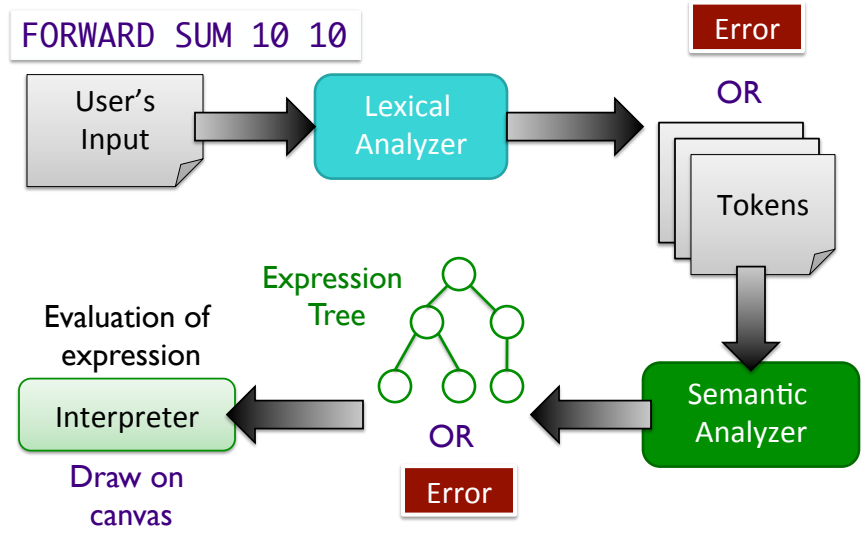
```
<id> <assignment> <num> <mult> <num> <endofstmt>
```

  - Verify that tokens are in a valid form
  - Generate executable code

Interpreting a Language



Interpreting a Language



What We Need to Do/Represent

- Lexical Analysis
- Semantic Analysis
- Evaluation

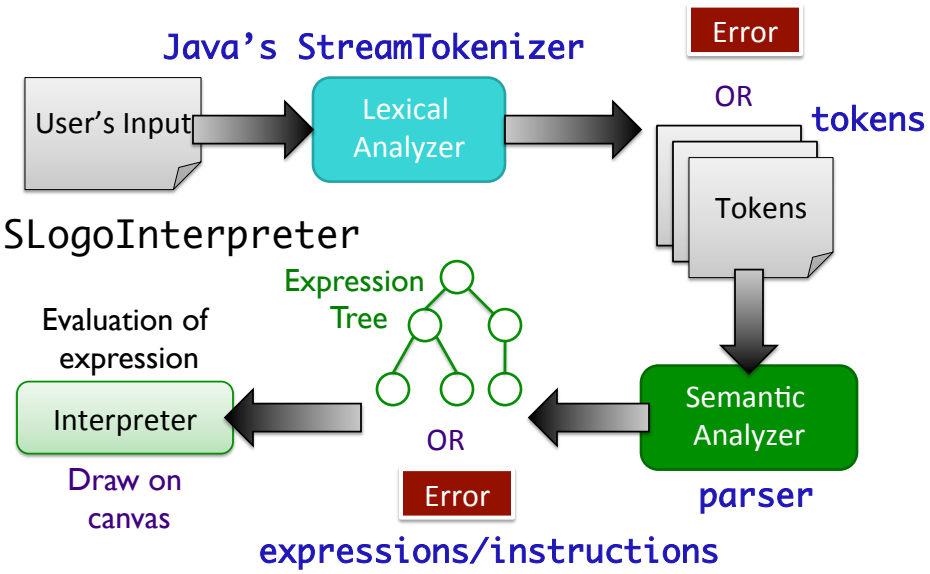
What We Need to Do/Represent

- Lexical Analysis
  - Recognize/create tokens
  - Report errors in creating tokens
- Semantic Analysis
  - Parse tokens into *expressions*
  - Report errors
- Evaluation
  - Evaluate expressions with respect to turtle/model

Understanding the Code

- How does the given code map to lexical analysis, semantic analysis, and evaluation components?

Interpreting SLogo Language



Understanding the Code: Lexical Analysis

- Important classes
  - `slogo.SLogoInterpreter`
  - `slogo.parser.tokens.TokenFactory`
- Output: `slogo.parser.token.*`

`PrintToken`

Understanding the Code: Semantic Analysis

- Important Classes
  - Common interface: `slogo.parser.Parser`
  - `slogo.parser.*Parser`
    - Ex: `slogo.parser.ExpressionParser`
  - `slogo.parser.InstructionParser`
    - Decides which instruction parser to call
- Output: `slogo.expression.*` or `slogo.instruction.*`

`PrintParser`

Understanding the Code: Evaluation

- Important Classes
  - Base class: `slogo.GrammarElement`
  - Subclasses:
    - `slogo.instruction.Instruction`
    - `slogo.expression.ArithmeticBase`
    - `slogo.instruction.Assignment`
- Key method: `evaluate(Context c)`
  - returns `Object`

`Print`

Bringing it together

- `slogo.*`
  - Breaks classes into appropriate packages: Tokens, Expressions, Instructions, Parsers
- `slogo.parser`
  - Parse Tokens to create Instructions
- `slogo.instructions`
  - Represent instructions
  - `evaluate` method

Bringing it together

- Mapping between `Token`, `Instruction`, `Parser`
  - Knows which `Parser` to call based on `instructions.prop` and mapping from `Token` to `Parser`
- Run `SLogoInterpreter`

Trace through Logo programs

- What happens when `print.logo` is interpreted?
  - Assign parts to team members
- One team member should add the file `repeat.logo` to `slogo_programs`
- Contents of file:

```
x = 4
REPEAT x [ PRINT 7 PRINT x ]
```

Practice Adding Instructions

1. Create a token for instruction
   - Likely a subclass of `token.ReservedToken`
   - Same prefix as new instruction, e.g., `IfToken.java`
2. Create a parser for the instruction with same prefix as instruction, e.g., `IfParser.java`
   - Parsing class (presumably implementing `Parser`) returns an instance of parsed Instruction
3. Create an instruction with prefix name, e.g., `If.java`
4. Add instruction name to file `instructions.prop`, e.g., add a single line to file containing string `If`

Brainstorming

- What do you need to do to complete the project?
- What do you "see" for the final project?
- What's going to *change*?
- Where do you think you'll run into problems?
- To focus your thinking, consider this use case: "The user starts the program, types 'fd 50' in the command window, and sees the turtle move in the display window leaving a trail."
  - What are other use cases?

Due Friday

Preparation Analysis

- What are the main parts/steps that need to be completed to complete the project?
  - How much work does each part require?
    - Approximate in terms of time or relative to the other steps.
  - How many people should work on each part?
- How will your program handle the following use case: "The user starts the program, types 'fd 50' in the command window, and sees the turtle move in the display window leaving a trail."?
  - From your description, it should be clear which classes/objects are responsible for completing each part of the task.
- What 3 extensions would you like to have in the final application?
- A plan for how you would tackle implementing the project
  - What parts can be completed independently of the other parts?
  - What parts need to be completed before other parts?
- The parts of the project you're most interested in working on, in ranked order.
- Any questions about the given specification.

TODO

- Project Analysis due Friday