

Objectives

- SLogo Design
- Discussion of Preparation Analyses

Nov 28, 2011

Sprenkle - CSCI209

1

Development Issues/Discussion

- How can you identify issues in the screensavers project?
- Design principles still hold
 - Use parent classes
 - Don't duplicate code
 - Ex: Location of random number generation
 - Readability: naming, no magic numbers, organization

Nov 28, 2011

Sprenkle - CSCI209

2

Review

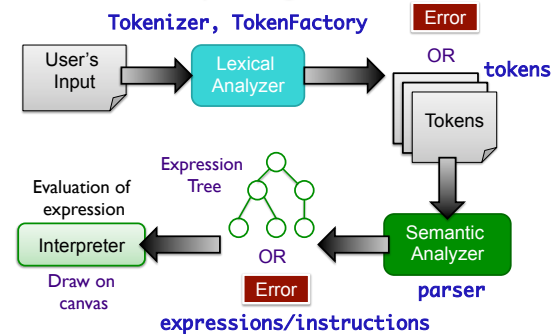
- How is a programming language processed?
 - What are the different phases?
- Start up Eclipse

Nov 28, 2011

Sprenkle - CSCI209

3

Review: Interpreting User's Input



Nov 28, 2011

Sprenkle - CSCI209

4

Review: 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`

Consult examples

Nov 28, 2011

Sprenkle - CSCI209

5

PLANNING THE PROJECT

Nov 28, 2011

Sprenkle - CSCI209

6

Definition of Use Case?

- Description of steps or actions between a user and a software system towards some goal

Nov 28, 2011

Sprenkle - CSCI209

7

What Steps Need To Be Completed?

- By the Team
- Process to figure out what needs to be completed

Nov 28, 2011

Sprenkle - CSCI209

8

What Steps Need To Be Completed?

- Model: Turtle
 - API
 - State
- GUI
 - Canvas – displays turtle
 - Command interface
 - Listeners
 - Multiple workspaces
 - Turtle info displayed (toggable)
 - More options/buttons (optional)
- **TESTING!**
- Parsing SLogo language
 - Handle instructions
 - **Handle errors** appropriately
- Evaluating expressions
 - Manipulate turtle appropriately
- File handling
 - Read file of SLogo files
 - Save SLogo commands

Nov 28, 2011

Sprenkle - CSCI209

9

Dependencies?

Nov 28, 2011

Sprenkle - CSCI209

10

Dependencies

- Interpreter classes (tokens, analyzer, expression) are very dependent on each other
- Need to hook GUI to Interpreter
- Need to hook Turtle to GUI and Interpreter
- Can test without other pieces but easier and more satisfying to see results displayed

Nov 28, 2011

Sprenkle - CSCI209

11

Effect of Extensions

- Extensions could affect your code design
 - Where could change → abstraction
- Decision?
 - May change your minds after start working on the code
 - Top vote getters

Nov 28, 2011

Sprenkle - CSCI209

12

Plan

- Tasks/Steps
 - Testing
 - Think about iterative development
 - Monday deadline: FD 50 working at least
- Division of tasks
 - # of people per part
- Deadlines

Nov 28, 2011

Sprenkle - CSCI209

13

Goals

- Implement one instruction completely
 - Involves a lot of different pieces
- Don't go too far in breadth, more depth
 - See design issues sooner
 - "We need method/functionality X in class Y"

Nov 28, 2011

Sprenkle - CSCI209

14

Secondary Goals

- You're going to figure out that your design isn't perfect--maybe not even good!
 - Fix smaller and/or more critical things
 - Refactoring!
 - Note larger things
 - analysis/post-mortem due at end of finals week

Good judgment comes from experience.
How do you get experience?
Bad judgment works every time.

Nov 28, 2011

Sprenkle - CSCI209

15

SLogo Timeline

- Monday, Dec 5: demo preliminary functionality of application (group)
- ??: final implementation due (group)
 - Latest Fri, Dec 16
- Fri, Dec 16: "Post-mortem" (individual)

Nov 28, 2011

Sprenkle - CSCI209

16

Exam Review

- Discuss the design principles applied and tradeoffs in how Sun designed the Java IO classes.
- Specifically, Java provides classes that handle reading from/writing to sources (e.g., files, Strings, network), classes that take as input other streams and filter those streams, and convenience classes that combine commonly used source and filter streams together.

Nov 28, 2011

Sprenkle - CSCI209

17

Exam Feedback

- Good:
 - Design for *change*
 - Comparing Java and Python
- Not so good:
 - JUnit properties
 - Change → Abstraction
 - Code smells → poor *design*
 - When to stop testing

| Grade | Score |
|-------|---------|
| A | 85.5-95 |
| B | 76-85.5 |
| C | 66.5-76 |
| D | 57-66.5 |

82% median,
average

Nov 28, 2011

Sprenkle - CSCI209

18