Jonathan Chen (jc953)
&& Ishaan Jhaveri (iaj8)

Assignment 4: **Parsing and Fault-Injection**

Entry Class: **Assignment-4.a4.Main.java**

Summary

The most challenging aspect of the assignment was deciding on the appropriate class hierarchy for the Abstract Syntax Tree for Critter actions. This affected our Parser as well as our Fault-Injection implementation. The hierarchy we finally settled upon is interesting in the uses of BinaryOp, BinaryCondition and RelationCondition. A BinaryCondition extends Condition and it represents a situation in which there are 2 Conditions on either side of the logical AND and OR operators. A RelationCondition is also a subtype of Condition, representing the more low level Condition situation in which there are 2 Expressions on either side of a relation (!, =, <, > and so on). Since an Expression can be simply a number or a series of numbers (Expressions) separated by multiplier or addition operators, there is a class BinaryOp that handles that, extending Expression itself. The Expression class has another use, with actions. Using an Expression as an action allowed for some crucial code reuse.

Implementing Fault-Injection, it was a small issue to keep track of parent classes at first, but not anymore.

We do not know of any problems with this code.

Specification

The ParserImpl class handles Parsing, using the class hierarchy we defined to understand the rules of the grammar.

Fault-Injection and Pretty-Printing are handled for each class by the class itself, yet the hierarchy is important here too, because classes rely on their hierarchical definitions for functionality.

Design Implementation

Classes and Architecture:

Node. Any class that exists in the Abstract Syntax Tree is a Node. Though Node is an interface, it is important that a class on the Abstract Syntax Tree, as it has crucial methods in it for Pretty-Printing and Fault-Injection

Program. The first Node in the Abstract Syntax Tree.

Condition and Command. Children of Program. Condition is an Interface.

BinaryCondition and RelationCondition are the two types of Conditions, they have been described above.

Expression. All Conditions end in Expressions.

BinaryOp. A subclass of Expression. A type of Expression where it is an Expression followed by one or more mathematical operators

ExtendedExpression. A specific type of Expression which has a Token and another Expression.

Update. A Child class of Command.

Token. The final unit of the Tree.

ParserImpl. To Implement the Parser.

Jonathan Chen (jc953)
&& Ishaan Jhaveri (iaj8)

Code Design.

We approached Parsing recursively, always Parsing a smaller and smaller piece of the whole. The methods in ParserImpl show this off.

Our algorithm for Pretty-Printing relied on the class hierarchy and printed recursively too. We had flags in the individual classes when we needed to print parentheses, braces and brackets.

For Fault-Injection, it was handled locally in the classes using the hierarchy. We use Math for randomness to determine what mutation occurs. We also keep track of parent nodes.

Our algorithms were all original.

Programming.

We used a Top-Down approach starting with Program. Thus when parse() returns a Program this has access to the entire Abstract Syntax Tree.

Since we constantly worked side-by-side the code division was quite arbitrary. By and large, Jonathan Chen wrote most of Fault-Injection, while Ishaan Jhaveri wrote most of Pretty-Printing and Testing. Parsing was conceptualized and coded together.

Testing

Our plan for testing is inputting various combinations of the Critterworld grammar provided it meets the rules of the Critterworld language. Our test cases are mostly variations on the hierarchy while still using legal syntax. We input grammars that we know are false to make sure they are properly dealt with by the Parser. We use the Parser to test the Pretty-Printer.

Jonathan Chen (jc953)
&& Ishaan Jhaveri (iaj8)

Known Problems

Our Fault-Injection does not work perfectly with the Parser.

Comments

This is a great starting point to implementing Critterworld. It is exciting to see what kind of mutations Fault Injection will lead to later on.

Time: Roughly 20 hours.

Advice: the Class hierarchy is paramount.

Surprising: How tricky Parsing was.

Hard: Parsing.

Like: Learning about using Languages.