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 so far was deciding on the appropriate class hierarchy for the Abstract Syntax Tree for Critter actions. We are a fluctuating between a few different designs, so we are not positive what our class design will finally be. We haven't run into any other major issues. Since we haven't yet implemented the Parser completely we cannot tell what problems we might run into.

Specification

We haven't deviated very far from the specification that was provided. We are considering adding a class to deal with Critter actions but haven't yet decided on its full functionality.

Design Implementation

We started off with a Node superclass. This is extended by Condition, Expression, Rule, and Program. BinaryCondition extends Condition. BinaryOp and Num extend Expression. The tokens of the grammar are all fields of class Token. We are considering defining BinaryToken that extends Token to represent binary operations.

The Parser class reads through a given file and uses Tokens to determine the meaning of the file. It does this by constructing an Abstract Syntax Tree using the class hierarchy we are currently working on.

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

Our Parser class has methods that read through the given file, and construct the corresponding Abstract Syntax Tree. Then using this Tree, a program is generated which instructs Critters on how to act. The program is printed out using the pretty-printing algorithm. The class Token handles pretty-printing, converting commands into something readable.

We used a top-down implementation where the fact that all commands extend Node is paramount.

Jonathan Chen is working on Parsing. Ishaan Jhaveri is working on Pretty-Printing. We have not decided on the division for Fault-Injection as yet.

Testing

Our plan for testing is inputting various combinations of the Critterworld grammar provided it meets the rules of the Critterworld language. We will design further test cases as needed after executing these test cases. We will also input grammars that we know are false to make sure they are properly dealt with by the Parser. We will use the Parser to test the Pretty-Printer.

We haven't tested yet, since we haven't finished implementing all of our code.

Known Problems

We have not encountered any as yet.

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.

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

We will write a more thorough overview when we have completed the assignment.