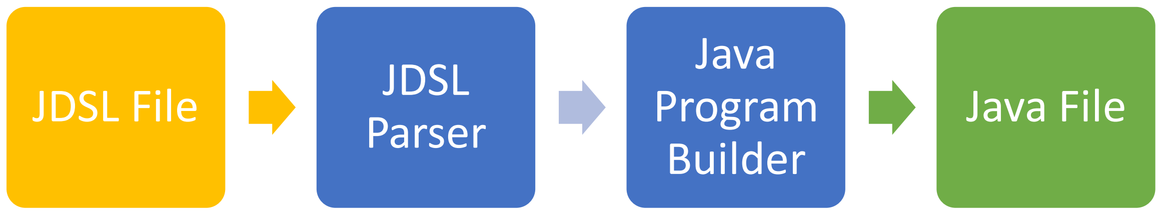
Christina Trotter **5/27/18**

Simple Java Program Builder Python DSL

Overview

Grading roughly eight-hundred CSCI 111 projects over the last two semesters gave birth to the idea of creating a program that can speed up the grading process and lessen the effort. This assignment is the first part of the overall project. For this assignment, an external domain specific language was defined for writing simple very simple java programs.



This program takes a jdsl file as input, parses it, then converts it to a java equivalent, and finally outputs a .java file.

This program is a very simple prototype. It can only generate simple syntactically correct java code. It is only useful for programs as intricate as those given as an assignment in a CSCI 111 course.

Program Execution

The entire program was written in python 3.6.2. It uses outside modules **copy**, **os**, **re**, **sys**, and **traceback**. All necessary outside modules should already be installed on a machine that has python 3 installed.

Before running the program make sure all the source code files and jdsl files are in the same directory. To run the program, use the command 'python3 jdsl.py' in a terminal window inside of the directory all the pessary files are in.

Future Work

The overall goal is to extend the program to be used to help check the validity of java files. The java code produced from the input jdsl file will be used as an answer key. Java code from an input java file will be parse and matched against the jdsl java code to estimate the validity/correctness of the input java file. Yes, there are easier, more efficient ways to streamline program grading. However, finding a solution that incorporates a domain specific language was an interesting challenge.

JDSL Keywords

assignment
instantiate already declared objects and variables with instances/values
usage: assignment [object/variable] [instance/value]

class - start a new class
usage: class [class name]

comment - add a new comment
usage: comment [contents of comment]

end - end code block
usage: end

fragment - add a miscellaneous line a code (e.g. void method call)
usage: fragment [line of code (sans semicolon)]

header - start a new program header
usage: header
note: see *Header Content Words* below

import - add a new import
usage: import [name of object class]
note: currently only works for objects from java.util since the Scanner and Random classes are the main imports needed in CSCI 111 programs.

method – start a new method
usage: method [name of method] [return type] [parameters]

object – declare a new object
usage: object [object type] [name of object]

print – add a new print statement
usage: print [contents of print statement]
print [contents of print statement] | [values/variables]

program – start a new program
usage: program [name of program]

return – add a new return statement
usage: return, return [values/variables]

statement – start a new if/else if/else statement or for/while loop
usage: statement [statement type]
statement [statement type] [condition]

variable – declare a new variable
usage: variable [variable type] [name of variable]

Header Content Words

In CSCI 111, the program header is worth a decent percentage the the overall project grade. Therefore, it was included in this program to allow for grade checking implementation in the future.

course_id - course code and section number
student_name – the student's first and last name
student_id – the student's ID
program_id – the program ID (e.g. 5)
due_date - the date the program is due
honor_code – honor code (e.g. UM's school of engineering honor code)
program_description – short description of the program

Source Code File Overview

blocks.py - blocks of code
this file contains all the classes of keyword objects that contain more than one line of code

lines.py - lines of code
this file contains the code for keyword objects that only take up one line of code

ijpb.py - Incremental Java Program Builder
this file contains the IJPB (and RecognitionError) class that is used to construct java code out of input, parsed jdsl code.

lp.py – line parser
this file contains the LineParser class that ever custom parser inherits from

cps.py – custom parsers
this file contains the custom line parsers that cater to the specifications of each keyword

jdsl.py – main program file
this file contains the "main method" and the top parsing objects

Sample Test File Overview

ly.jdsl – LeapYear.java JDSL representation

hw.jdsl – HelloWorld.java JDSL representation

mc.jdsl – MultiClasses.java JDSL representation