Java to Python

Elizabeth Mahon and Brendan Ritter

Problem Statement

We were curious about methods of translating one language into another, and wanted to apply what we learned in this class to languages that are commonly used in the real world.

Proposed Solution

We were familiar with both Java and Python, and both are commonly used in industry.

We decided to translate from Java to Python because Python has is an object oriented imperative language like Java which would make translation smoother and more in scope with the project. While it clearly is possible to translate Java into another language, that would be much too large a project for this class.

Structure of Solution

We split the translation into three parts:

- Parser
  - Takes Java and turns it into tokens, then to an internal representation
- Internal representation
  - Holds the data needed to get working code
  - Three main types of data, statements, expressions and scope (keywords)
- Translator
  - Takes the internal representation and generates Python

Internal Representation

```
(*things that can return a value*)
datatype limExpr =  ECall of string*(expr list)
          | Var of string
          | Paren of expr
          | Not of expr
          | Neg of expr
          | ArrLit of expr list

and expr=
          LExpr of limExpr
          | EInfix of limExpr*string*expr
```

```
(*things that can't return a value*)
and stmt =   ClassDef of scope*string*stmt
          | MethDef of scope*string*string*(string*string) list*stmt
          | Initial of scope*string*string*expr
          | SmInitial of scope*string*string
          | Assign of string*expr
          | SCall of string*(expr list)
          | If of expr*stmt
          | IfElse of expr*stmt*stmt
          | While of expr*stmt
          | Return of expr
          | Block of stmt list
          | Comment of string*stmt
          | SInfix of expr*string*stmt
          | For3 of stmt*expr*stmt*stmt
          | CheatExpr of expr
```

Sample Input

```
/*Simple test program that prints out Hello There!
followed by four verses of Make New Friends.
Made for Software Engineering, Spring 2013 at Olin College.
@author Elizabeth Mahon*/
public class MyFirstApp {
	public static void main(String[] args) {
		System.out.println("Hello there!");
		int x = 4;
		String[] song = {"Make new friends, but keep the old", "One is silver and the other's gold",
"A circle's round, it has no end", "That's how long I want to be your friend", "Across the ocean, across
the sea", "Friends forever we will always be", "The grass is green, the sky is blue", "Friends forever,
me and you"};
		System.out.println("Make New Friends");
		while (x > 0) {
			System.out.println("Verse: " + (5 - x));
			System.out.println(song[2*(4 - x)]);
			System.out.println(song[2*(4 - x) + 1]);
			x--;}}}
```

Sample Output

```
Simple test program that prints out Hello There!
followed by four verses of Make New Friends.
Made for Software Engineering, Spring 2013 at Olin College.
@author Elizabeth Mahon
"""
class MyFirstApp:
   def main(args):
        System.out.println("Hello there!")
        x=4
        song=["Make new friends, but keep the old","One is silver and the other's gold","A circle's round, it has no end","That's how long I want to be your friend","Across the ocean, across the sea","Friends forever we will always be","The grass is green, the sky is blue","Friends forever, me and you"]
        System.out.println("Make New Friends")
        while(x > 0):
             System.out.println("Verse: " + (5 - x))
             System.out.println(song[)x-4(*2])
             System.out.println(song[1+)x-4(*2])
            x -= 1
```

Difficulties Encountered

- Initially lacked structure
  - Wanted to convert directly from tokens to python
  - Cognates galore!
- Then wanted to ignore difference between statements and expressions
- Infix operators are annoying
  - Tried to not differentiate between statement infix and expr infix
  - Tried to make semi-colons optional
  - Finally gave up at did it the right way

Difficulties encountered

- Left recursion Infix continued
  - resolved through adding more parse rules
- Expression/statement ambiguity
  - E.g. a call could be an expression or statement
  - Resolved through… brute force?
- Could actually look at java parser documentation
  - Not as fun