La programación orientada a objetos ha encontrado desde la década de los 90s un nicho en el área de la ingeniería de software debido a que permite abstraer problemas de administración de información de una manera muy estructurada.

Este proyecto pretende aplicar la programación orientada a objetos al estudio de lenguajes de programación. Este ejercicio nos permitirá pensar en las características estructurales de los elementos básicos de un lenguaje de programación funcional.

El objetivo es construir un intérprete de un subconjunto del lenguaje de programación funcional Scheme utilizando Groovy.

Especificación

Construir un intérprete interactivo de línea de comando que lea una expresión en Scheme y la evalúe correctamente, imprimiendo el valor resultante en pantalla. A continuación se especifica el conjunto mínimo de funcionalidad que debe implementar:

- Valores literales para números naturales (`0`, `1`, `3234`) y booleanos (`#t`, `#f`)
- Funciones incorporadas para aritmética entera (`+`, `-`, `*`, `modulo`, `quotient`)
- Función condicional incorporada (`if`)
- Funciones incorporadas para relaciones (`=`, `<`, `<=`, `>`, `>=`)
- Funciones incorporadas para lógica (`and`, `or`, `not`)
- Funciones definidas por el programador (`define`). El cuerpo de una función siempre será una única expresión.
- Expresiones entendidas como invocación de funciones. Las expresiones pueden ser anidadas, es decir pueden trabajar sobre valores literales o sobre otras expresiones.

Ejemplo de uso:

```
$> groovy MiniScheme.groovy

scheme> 1
1
scheme> (+ 1 2)
3
scheme> (if #t 1 2)
1
scheme> (not (= 1 2))
#t
scheme> (+ (* 2 2) (* 3 3))
13
```

```
scheme> (define (fact n) (if (= n 0) 1 (* n (fact (- n 1)))))
Function: fact
scheme> (fact 5)
120
scheme> (+ 1 (fact 5))
121
```

- Puede suponer que las entradas del usuario siempre van a ser correctas (no es necesario manejar errores de programación).
- La definición de nuevas funciones con `define` no es una expresión.
- No existe el concepto de variables locales, sólo de parámetros.
- Debe manejar una pila de activación para la invocación de funciones.
- Debe manejar una tabla de símbolos para asociar el nombre de una función con la expresión que se debe ejecutar cuando se invoque la función.

Recursos

Parser

```
/**
 * El código Scheme corresponde a listas de símbolos demarcadas por
 * paréntesis ( ).
 * Este parser lee el código parentizado y produce las listas de
 * de símbolos correspondientes en Groovy.
 */
def parse(String code) {
    def stack = []
    def current = null
    def tokens = code.replace('(','( ').replace(')',' )').split(' ')
    tokens.each { token ->
        switch(token) {
            case '(':
                stack.push(current)
                current = []
                break
            case ')':
                def top = stack.pop()
                current = top ? top << current : current
                break
            default:
                current << token
        }
    }
    current
}

parse('(define (fact n) (if (= n 0) 1 (* n (fact (- n 1)))))')
// produce:
// [define, [fact, n], [if, [=, n, 0], 1, [*, n, [fact, [-, n, 1]]]]]
```

```
parse('(if (not (= 1 2)) (quotient 3 2) 5)')
// produce:
// [if, [not, [=, 1, 2]], [quotient, 3, 2], 5]
```

Logística

El proyecto debe ser trabajado en grupos de máximo 3 personas. El proyecto debe ser entregado el día **martes 16 de junio, 2015** a más tardar a las 11:59pm. Enviar un correo electrónico a la dirección dmm.itcr@gmail.com, el asunto debe ser los apellidos de los integrantes del grupo ordenados alfabéticamente y el sufijo "-P4", ej: RamírezSotoZamora-P4. Se debe adjuntar al correo un zip con el mismo nombre del asunto que contenga los archivos de código fuente con la solución del proyecto.

Adicionalmente se deben enviar avances semanales que contaran como un rubro más a calificar, los avances corresponden al código que se tenga al momento, no necesariamente tiene que funcionar/compilar. **El primer avance debe incluir el modelo de objetos planteado para solucionar el problema**. Los avances deben enviarse a la dirección de correo anteriormente mencionada, utilizando el mismo formato de nombre para el asunto y el archivo comprimido. Las fechas para entrega de avances son: **jueves 4 de junio**, **jueves 11 de junio**.