Paradigmas de la Programación
Laboratorio 2

Laura Alonso Alemany Franco Luque Ezequiel Orbe

24 de abril de 2014

Propiedades de los lenguajes de programación

El objetivo de este laboratorio es ejercitar y consolidar su capacidad para implementar pequeños programas que les ofrezcan un diagnóstico para determinar de qué forma un lenguaje particular implementa alguno de los conceptos vistos en el teórico. El objetivo de los programas de diagnóstico es que ofrezcan diferentes resultados para las diferentes opciones: tipado fuerte y débil, asignación simple o múltiple, etc.

Consignas

1. Qué tipado (estático o dinámico) tienen los siguientes lenguajes de programación: `python`, `java`, `scala` y `ruby`? Para cada uno de ellos cree un programa simple que lo demuestre.
2. Son los lenguajes `python`, `java`, `scala`, `ruby` y `javascript` de asignación única como `OZ`? Para cada uno de ellos cree un programa simple que lo demuestre.
3. Considere los siguientes lenguajes: `python`, `java`, `scala`, `ruby` y `javascript`. Qué tipo de alcance utilizan (estático o dinámico)? Para cada uno de ellos cree un programa simple que lo demuestre.
4. Elijan un lenguaje de tipado débil y otro de tipado fuerte. Cree un programa simple que haga evidente las diferencias entre uno y otro.
5. El compilador de Scala detecta recursiones a la cola y las reemplaza con un salto al principio de la función actualizando los parámetros correspondientes para evitar la sobrecarga de poner la misma función en la pila. Por ejemplo en el siguiente código, la función se llama a sí misma y al final levanta una excepción de manera que podamos ver la pila de llamadas.

```
def boom(x: Int) : Int =
   if (x==0) throw new Exception(‘‘boom’’)
   else boom(x-1) + 1
```

¿Es esta función recursiva a la cola? ¿por qué? Si la ejecutamos, vemos las sucesivas llamadas a la recursión.

```
scala>boom(3)
java.lang.Exception: boom!
        at .boom(<console>:5)
        at .boom(<console>:5)
        at .boom(<console>:5)
        at .boom(<console>:5)
        at .<init>(<console>:5)
```

Si modificamos la función `boom` de la siguiente manera:

```
def bang(x: Int) : Int =
   if (x==0) throw new Exception(``bang'')
   else bang(x-1)
```

tenemos:

```
scala>bang(5)
java.lang.Exception: bang!
        at .bang(<console>:5)
        at .<init>(<console>:6)
```

El uso de la recursión a la cola en Scala está bastante limitado. Scala sólo optimiza directamente llamadas recursivas a la misma función. Escriba código simple que muestre que dos funciones recursivas a la cola mútuamente no son optimizadas por el compilador.

6. En Scala se puende definir variables como `lazy`, la semántica dice que estas variables serán computadas la primera vez que se lean y que su valor no será recomputado en cada una de las veces que se lee de nuevo la variable. Si se quiere un comportamiento donde se recompute la variable, entonces hay que usar alto orden y definir a las variables como funciones que devuelven un valor. Investigue este tema y escriba código que muestre sus diferencias.

Características de la presentación

- El trabajo es **individual**.
- Formato de presentación:
  - La presentación del laboratorio se realizará via mail (paradigmas.famaf.2014@gmail.com).
  - El mail deberá contener lo siguiente:
    - Asunto: Lab 2 - $<$ *apellido* $>$, $<$ *nombre* $>$
    - Un informe (en formato PDF) donde se explique la resolución de cada ejercicio incluyendo el código utilizado y la salida en pantalla que se obtuvo al ejecutar el código.
    - Un archivo .tar.gz o .zip que contenga el código fuente utilizado en cada ejercicio.
- Fecha de Entrega: 07 de Mayo (Se recibiran mails hasta las 23:59:59).
- Fecha de Defensa: 09 de Mayo (de 9:00 a 13:00).
- Condición de Aprobación: tener 5 de los 6 ejercicios resueltos.