Paradigmas de la Programación
Laboratorio 4: Propiedades de los lenguajes de programación

Laura Alonso Alemany Cristian Cardellino Ezequiel Orbe

En este laboratorio trabajaremos con los conceptos aprendidos en el teórico y su implementación en un conjunto de lenguajes de uso práctico.

Para ello, deberán desarrollar pequeños programas en `Scala`, `Ruby`, `Python`, `Haskell`, `Java`. `C` y `Javascript`, que permitan determinar de que forma, cada lenguaje implementa los conceptos vistos en el teórico.

Características de la presentación

- El trabajo es en grupo: máximo: **dos integrantes**, mínimo: **dos integrantes**.
- Fecha de Entrega: Hasta las 23:59:59 del 14/05/2015.
- Formato de entrega:
  1. Se deberá realizar 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.
  2. Empaquetar el directorio `lang-tests` y el informe en un archivo llamado `<dni>-lab-4.tar.gz`, donde `<dni>` es el DNI de alguno de los integrantes del grupo.
  3. Enviar el archivo `<dni>-lab-4.tar.gz` por mail a la dirección:

     `paradigmas@famaf.unc.edu.ar`

     El título del mail debe decir:`lab-4`. En el cuerpo del mail se deben indicar los integrantes (nombre completo y DNI).
  4. Los trabajos enviados fuera de término serán evaluados de acuerdo a la siguiente Pólitica de Entrega Tardía:
     - Hasta 1 día despues del deadline: -50% de la nota.
     - Hasta 2 días despues del deadline: -60% de la nota.
     - Hasta 3 días despues del deadline: -70% de la nota.
     - Hasta 4 días despues del deadline: -80% de la nota.
     - Hasta 5 días despues del deadline: -90% de la nota.
     - Más de 5 días despues del deadline: el trabajo se considera NO entregado y se debe recuperar.
  5. Los trabajos enviados que no cumplan con las condiciones de presentación no serán considerados.

Software Requerido

Para realizar este laboratorio necesitaremos el siguiente software:

- Compilador de `Haskell`.
- Compilador de `Python`.
- Compilador de `C`.
- Compilador de `Java`.
- Compilador de `Scala`.
- Compilador de `Javascript`.
- Compilador de `Ruby`.

Pueden utilizar algún compilador online que les guste. Algunas opciones son: Ideone o Compile Online

Para `Javascript` pueden utilizar la consola de Google Chrome.

Código Inicial

Para realizar este laboratorio, utilizaremos el código disponible en:

`http://cs.famaf.unc.edu.ar/materias/paradigmas/node/192`

Al descomprimir el archivo, encontrarán un directorio llamado `lang-tests`, y dentro del mismo 9 directorios, llamados `test-n`, uno para cada una de las 9 consignas.

Los programas desarrollados para cada consigna deberán ser colocados en el directorio correspondiente (ej.: los programas realizados para la consigna 1 deben ser colocados en el directorio `test-1`).

Consignas

1. [**Tipado**] Determine si los siguientes lenguajes son de tipado estático o tipado dinámico: `Python`, `Haskell`, `Java`, `Scala`, `C`, `Javascript` y `Ruby`. Para cada uno de estos lenguajes, cree un programa (simple) que lo demuestre.
2. [**Tipado**] Determine si los siguientes lenguajes son de tipado débil o tipado fuerte: `Python`, `Haskell`, `Java`, `Scala`, `C`, `Javascript` y `Ruby`. Para cada uno de estos lenguajes, cree un programa (simple) que lo demuestre.
3. [**Asignación Única**] Determine si los siguientes lenguajes son de asignación única: `Python`, `Haskell`, `Java`, `Scala`, `C`, `Javascript` y `Ruby`. Si no son de asignación única, indique si tienen algún mecanismo para obtener un comportamiento similar. Para cada uno de estos lenguajes, cree un programa (simple) que lo demuestre.
4. [**Alcance**] Determine si los siguientes lenguajes tienen alcance estático o alcance dinámico: `Python`, `Haskell`, `Java`, `Scala`, `C`, `Javascript` y `Ruby`. Para cada uno de estos lenguajes, cree un programa (simple) que lo demuestre.
5. [**Recursion a la Cola**] Determine si los siguientes lenguajes realizan la llamada optimización de llamada a la cola (tail call optimization): `Python`, `Haskell`, `Java`, `Scala`, `C`, `Javascript` y `Ruby`. Para cada uno de estos lenguajes, cree un programa (simple) que lo demuestre.
6. [**Alto Orden**] En `Scala` se pueden definir variables como `lazy`, la semántica de `Scala` dice que estas variables serán computadas la primera vez que se necesiten y que su valor no será recomputado si la variable es reutilizada luego en el programa. Si se quiere un comportamiento donde se recompute la variable cada vez que se la necesite, entonces hay que usar alto orden y definir a las variables como funciones que devuelven un valor. Cree un programa que muestre estas diferencias.

7. [**Pasaje de Parámetros**] ¿Qué diferencia hay en `Scala` entre pasar una variable por valor, pero *lazy*, a pasar a una variable por nombre? Cree un programa que muestre estas diferencias.

8. [**Pasaje de Parámetros**] En los siguientes fragmentos de programa, introduzcan los elementos necesarios para determinar si `C` y `Perl` funcionan por *call-by-value* o *call-by-reference*:

   Fragmento en `C`:

```c
int main(void) {
    int x=50, y=70;
    ...
    printf("x=%d y=%d", x, y);
    return 0;
}

void interchange(int x1, int y1) {
    int z1;
    ...
    ...
    ...
    printf("x1=%d y1=%d", x1, y1);
}
```

   Fragmento en `Perl`:

```perl
$x=50;
$y=70;
&interchange ($x, $y);
print "x:$x y:$y\n";

sub interchange {
  ($x1, $y1) = @_;
  ...
  ...
  ...
  print "x1:$x1 y1:$y\n";
}
```

9. [**Pasaje de Parámetros**] Describa qué sucede en el siguiente programa en `Java`, teniendo en cuenta el mecanismo de pasaje de parámetros que utiliza el lenguaje.

```
public class Point {
	public int x;
	public int y;

	public Point(int x, int y){
		this.x = x;
		this.y = y;
	}

	public static void tricky1(Point arg1, Point arg2)
	{
	  arg1.x = 100;
	  arg1.y = 100;
	  Point temp = arg1;
	  arg1 = arg2;
	  arg2 = temp;
	}

	public static void tricky2(Point arg1, Point arg2)
	{
	 arg1 = null;
	 arg2 = null;
	}

	public static void main(String [] args)
	{
	  Point pnt1 = new Point(0,0);
	  Point pnt2 = new Point(0,0);
	  System.out.println("pnt1 X: " + pnt1.x + " pnt1 Y: " +pnt1.y);
	  System.out.println("pnt2 X: " + pnt2.x + " pnt2 Y: " +pnt2.y);
	  System.out.println("triki1\n");
	  tricky1(pnt1,pnt2);
	  System.out.println("pnt1 X: " + pnt1.x + " pnt1 Y:" + pnt1.y);
	  System.out.println("pnt2 X: " + pnt2.x + " pnt2 Y: " +pnt2.y);
	  System.out.println("\ntriki2");
	  tricky2(pnt1,pnt2);
	  System.out.println("pnt1 X: " + pnt1.x + " pnt1 Y:" + pnt1.y);
	  System.out.println("pnt2 X: " + pnt2.x + " pnt2 Y: " +pnt2.y);
	}
}
```