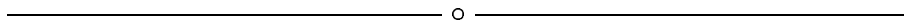


Traduciendo a Java

Asignación, y Expresiones Aritméticas y Booleanas

Este capítulo es dedicado a la instrucción de asignación. Como el operando derecho de una asignación es una expresión, se explica también cómo traducir expresiones aritméticas y expresiones booleanas del pseudo-lenguaje a JAVA.



0 Asignación

Una instrucción de asignación del pseudo-lenguaje se traduce a JAVA de la siguiente forma:

$$\langle \text{variable} \rangle := \langle \text{expresion PL} \rangle \rightsquigarrow \langle \text{variable} \rangle = \langle \text{expresion Java} \rangle ;$$

donde $\langle \text{expresion Java} \rangle$ es la expresión en JAVA correspondiente a la expresión $\langle \text{expresion PL} \rangle$ del pseudo-lenguaje. En la traducción resultante hay dos puntos importantes a comentar: primero, el uso del símbolo “=” para asignación; segundo, el uso de un “;” para finalizar la instrucción (note que en la instrucción original del pseudo-lenguaje que fue traducida no hay “;”).

La decisión de los diseñadores de JAVA, decisión heredada de los lenguajes C y C++, de utilizar el símbolo “=” para la instrucción de asignación no es, en opinión de muchos, incluyendo a quien suscribe, una decisión afortunada. Tradicionalmente, el símbolo matemático “=” denota la igualdad entre dos objetos y, por lo tanto, la decisión de utilizar en JAVA este símbolo para denotar un concepto totalmente diferente como lo es la asignación es comparable a, por ejemplo, decidir utilizar el símbolo “+” para denotar la operación de división. Es por esto que en el pseudo-lenguaje se utiliza para la asignación el símbolo “:=”, que es el símbolo que utilizan para esto otros lenguajes como PASCAL, MODULA y algunos de sus sucesores. Como en JAVA el símbolo “=” denota a la asignación, el operador de igualdad, requerido en expresiones booleanas, debe denotarse por un símbolo diferente; de nuevo siguiendo a los lenguajes C y C++, en JAVA se utiliza a “==” para denotar la igualdad.

Con respecto al “;” que finaliza la instrucción de asignación en JAVA, es importante recalcar que éste no tiene nada que ver con la operación de secuenciación del pseudo-lenguaje. La secuenciación en JAVA será explicada en su debido momento, en un próximo capítulo. Por lo pronto, el “;” que nos ocupa es simplemente un *terminador* de la instrucción de asignación; esto es, sintácticamente *toda* asignación en JAVA debe terminar en “;” (independientemente de que ésta se esté secuenciando con una siguiente instrucción o no). Por esta razón, en el esquema de traducción dado, en el lado derecho (JAVA) aparece un “;” que no estaba en el lado izquierdo (pseudo-lenguaje).

1 Expresiones Aritméticas

Las expresiones aritméticas en JAVA son muy parecidas a las del pseudo-lenguaje. Los operadores de suma “+”, resta “-” y multiplicación “*” se escriben igual en JAVA. La división sí tiene sus sutilezas. En el pseudo-lenguaje tenemos a “/” para división real y a “div” para división entera. En JAVA se utiliza el mismo símbolo “/” para ambas

divisiones, discriminando si se requiere la real o la entera de acuerdo al tipo de los operandos: si ambos operandos de “/” son enteros, se toma división entera; si alguno de los operandos es real, se toma división real. El operador “mod”, de resto de división entera, se denota “%” en JAVA.

Entre las librerías predefinidas de JAVA, se encuentra una de nombre `Math` que implementa varias operaciones útiles sobre números enteros y reales. Entre ellas tenemos a `abs`, `max` y `min`: “`Math.abs(a)`” da el valor absoluto de a , y “`Math.max(a,b)`” y “`Math.min(a,b)`” dan el máximo y el mínimo, respectivamente, entre a y b .

2 Expresiones Booleanas

Primero tenemos los operadores relacionales, que toman dos números enteros o reales y dan un resultado booleano. La tabla de traducción es la siguiente:

“=” \rightsquigarrow “==” ,
“ \neq ” \rightsquigarrow “!=” ,
“<” \rightsquigarrow “<” ,
“ \leq ” \rightsquigarrow “<=” ,
“>” \rightsquigarrow “>” ,
“ \geq ” \rightsquigarrow “>=” .

Luego tenemos los operadores lógicos o booleanos, que se traducen de la siguiente forma:

“ \wedge ” \rightsquigarrow “&&” ,
“ \vee ” \rightsquigarrow “||” ,
“ \neg ” \rightsquigarrow “!” .

Los operadores de implicación “ \Rightarrow ” y equivalencia “ \equiv ” no tienen contraparte directa en JAVA. Un detalle interesante de los operadores `&&` y `||` de JAVA es que cortan circuito: si la evaluación del primer operando permite determinar el resultado final (`false` en el caso de `&&` y `true` en el caso de `||`), el segundo operando no es evaluado. Esto es muy útil en expresiones como `(a != 0) && (b/a > 100)`, que daría error de división por cero cuando a vale 0 si el operador `&&` no hiciese corto-circuito. JAVA provee las variantes `&` y `|` para conjunción y disyunción, respectivamente, que NO hacen corto-circuito, pero muy rara vez es requerida la evaluación total, único caso en el que estas variantes deberían preferirse a las dadas arriba en la tabla de traducción.

Vale la pena comentar sobre expresiones, usadas muy frecuentemente en matemáticas, de la forma $x < y < z$. La traducción directa de esta expresión a JAVA como `x < y < z` daría error, pues JAVA interpretaría a esta expresión como `(x < y) < z` y no tendría sentido comparar a `(x < y)`, que es un valor booleano, con `z`, que sería un entero o un real. La traducción correcta se obtiene haciendo explícito el hecho que la expresión $x < y < z$ no es más que una abreviación de $(x < y) \wedge (y < z)$, con lo que obtenemos la formulación en JAVA adecuada: `(x < y) && (y < z)`.