Tema 4:
Operadores en Java

Operadores

- Expresión = operandos +operadores

Associative	Operators
R to L	++ -- + - ~ ! (data type)
L to R	* / %
L to R	+ -
L to R	<< >> >>>
L to R	< > <= >= instanceof
L to R	== !=
L to R	&
L to R	^
L to R	|
L to R	&&
L to R	||
R to L	?:
R to L	= *= /= %= += -= <<= >>= >>>= &= ^= |=

O. Aritméticos

Operador	Significado
+	Suma
-	Resta
/	División
*	Multiplicación
%	Residuo (div. Entera)

O. de Asignación

- A

Operador	Utilización	Expresión equivalente
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2

O. de Incremento y decremento

- ++
- - -

Expression	Initial Value of *x*
`y = x++`	5
`y = ++x`	5
`y = x--`	5
`y = --x`	5

Final Value of *y*	Final Value of *x*
5	6
6	6
5	4
4	4

O. Relacionales

- R

Operador	Utilización	El resultado es true
>	op1 > op2	si op1 es mayor que op2
>=	op1 >= op2	si op1 es mayor o igual que op2
<	op1 < op2	si op1 es menor que op2
<=	op1 <= op2	si op1 es menor o igual que op2
==	op1 == op2	si op1 y op2 son iguales
!=	op1 != op2	si op1 y op2 son diferentes

O. Lógicos

- L

Operador	Nombre
&&	AND
||	OR
!	negación
&	AND
|	OR

Utilización	Resultado
op1 && op2	true si op1 y op2 son true. Si op1 es false ya no se evalúa op2
op1 || op2	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2
! op	true si op es false y false si op es true
op1 & op2	true si op1 y op2 son true. Siempre se evalúa op2
op1 | op2	true si op1 u op2 son true. Siempre se evalúa op2

O. Bitwise y O. Corrimiento

- B

Operador
>>
<<
>>>
&
|
^
~

Utilización	Resultado
op1 >> op2	Desplaza los bits de op1 a la derecha una distancia op2
op1 << op2	Desplaza los bits de op1 a la izquierda una distancia op2
op1 >>> op2	Desplaza los bits de op1 a la derecha una distancia op2 (positiva)
op1 & op2	Operador AND a nivel de bits
op1 | op2	Operador OR a nivel de bits
op1 ^ op2	Operador XOR a nivel de bits
~op2	Operador complemento

```
        00110011
        11110000
XOR     --------
        11000011
```

Ejemplos

- Ejemplo 1

```
int i = 13;
// i es 00000000 00000000 00000000 0000 1101
     i = i << 2;
// i es 00000000 00000000 00000000 0011 0100
```

- Ejemplo 2 :

```
int b = -11;
 //11111111 11111111 11111111 1111 0101
b = b >>> 2;
// b ahora se vuelve
//00111111 11111111 11111111 1111 1101
```

- Ejemplo 3:

```
int b = -11; // 11111111 11111111 11111111 1111 0101
b = b >> 2; // 11111111 11111111 11111111 1111 1101
```

O. Bit-wise

- Asignación a nivel de bits

Operador	Utilización	Equivalente a
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2
^=	op1 ^= op2	op1 = op1 ^ op2
<<=	op1 <<= op2	op1 = op1 << op2
>>=	op1 >>= op2	op1 = op1 >> op2
>>>=	op1 >>>= op2	op1 = op1 >>> op2

O. instanceof

- Sirve para verificar si una variable de tipo Object* es de un tipo de Clase determinada.**
- Ejemplo:

```
if (x instanceof String) {
  // do nothing
}
else {
```

- Devuelve *true* o *false*

O. Condicional ? :

- Es un operador ternario porque ocupa 3 operandos.

- Su funcionalidad es como la del if-else.

- Ejemplo:
  Si **a**,**b** y c son tipo int , **x** boolean

  La sentencia **a**= **x** ? **b** : **c**;
  es equivalente a

```
if (x) {
  a = b;
}

else {
  a = c;
}
```

Casting

- Casting es el proceso de tomar una variable declarada de un cierto tipo y “decirle” al compilador que se quiere tratar como otro tipo.
- El casting no modifica el ítem original.
- Válido para tipos primitivos como int o float y para tipos de objetos como java.lang.Object y java.lang.String.

- Sintaxis:

(datatype) valor/expression

donde datatype= nuevo tipo.

Casting

- Tipos de casting:
  - ✓ A Primitivos
  - ✓ A referencias de objetos*
  - ✓ De expresiones

- Ejemplos:

```
long bigValue = 99L;
int squashed = (int)(bigValue);
```

- short a,b,c;
  a=1;
  b=2;
  c=a+b; //error

- Solucion 1:
  short a,b; int c;
  a=1;
  b=2;
  c=a+b;

  Solucion 2:
  short a,b,c;
  a=1;
  b=2;
  c= (int) a+b;

Promoción (widening)

- Las variables pueden ser promovidas automáticamente, es decir hace una conversión automática a un tipo de dato “mas grande” siempre y cuando sea “compatible”.

- Las promociones toleradas por Java son:

✓ byte→short→int→long→float→double

✓ char→int→long→float→double

- La operación inversa de la promoción es llamada reduccción.

Promoción

- Ejemplos:

```
long bigval = 6;     // 6 is an int type, OK
int smallval = 99L; // 99L is a long, illegal

double z = 12.414F; // 12.414F is float, OK
float z1 = 12.414;  // 12.414 is double, illegal
```

Operadores de concatenación

- Los operadores **+** y **+=** también se pueden aplicar a cadenas. Ambos realizan una concatenación y están implementados con objetos StringBuffer **.

- Ejemplo:
- String s = "¿Qué" + " haces ?";

// es interpretada por el compilador como:

String s= new StringBuffer().append("¿Qué").append(" haces ?").toString();

//y se marcaría el StringBuffer para borrarlo ya que el contenido pasa al objeto String.