Tipos de datos primitivos

Tokens

- Elementos léxicos de un lenguaje de programación
  - Identificadores
  - Palabras reservadas
  - Separadores
  - Constantes
  - Operadores

```
PUBLIC SUB Main()
i AS Integer
b AS Boolean
c AS Boolean

FOR i = 1 TO 10
   PRINT i, i*i, i^3.14
NEXT
FOR b = TRUE TO FALSE
  FOR c = TRUE TO FALSE
    PRINT b AND c, b OR c, b XOR c, NOT b
  NEXT
NEXT
END
```

Palabras reservadas

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Palabras literales reservadas: null, true y false

Identificadores

- Son los nombres para
  - Clases
  - Objetos
  - Métodos
  - Atributos
- Inician con
  - Letra unicode
  - Caracter $ o _
- Son sensibles a mayúsculas y minúsculas
- No tienen longitud máxima

```
public class Ejemplo {
   public static void main (String args[ ]) {
       System.out.println("Un simple programa Java");
   } // fin método main
}
```

Separadores

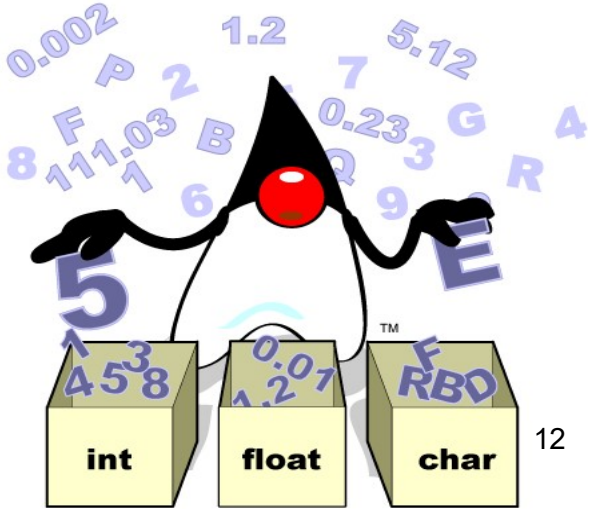
- ;
  - Una instrucción es una o más líneas de código terminadas con un ;
- { }
  - Un bloque, es una colección de instrucciones, entre llaves
- Espacio en blanco
  - Se permite cualquier cantidad de espacios en blanco

```
{int x; x=4*5;}
```

```
{
    int x;
    x=4*5;
}
```

Tipos de datos primitivos en Java

- El tipo de dato de un objeto, determina el espacio que éste ocupa en memoria y cómo usarlo
- Las constantes y los operadores dependen de los tipos de dato
- Java tiene 8 tipos de datos primitivos
  - Enteros: byte, short, int, long
  - Flotantes: float, double
  - Lógicos: boolean
  - Textuales: char



Enteros

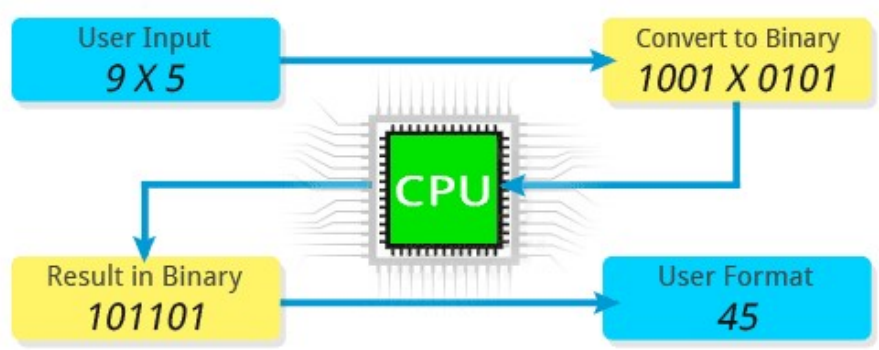
- Declaración
  - byte 8 bits -128 .. 127
  - short 16 bits -32,768 .. 32,767
  - int 32 bits -2,147,483,648 .. 2,147,483,647
  - long 64 bits -9,223,372,036,854,755,808 .. 9,223,372,036,854,755,807
- Ejemplos de Constantes
  - byte, short, int
    - 150 0150 0x150 0b1001 1_475
  - long
    - 150l 150L 123_456_789_000_000L

Java 7

- Java 8SE clase Integer permite int sin signo: $2^{32}-1$ = 4,294,967,296 - 1

Representación interna de enteros

- Números positivos
  - En cuántos bits se va a almacenar



- Números negativos
  - Complemento a 2

```
1 0 1 1 0 1 ← NUMERO BINARIO ORIGINAL
↓ ↓ ↓ ↓ ↓ ↓
0 1 0 0 1 0 ← NUMERO BINARIO EN COMPLEMENTO A 1
        + 1
-------------
0 1 0 0 1 1 ← NUMERO BINARIO EN COMPLEMENTO A 2
```

Cambio de base n a 10

- Aplicar la fórmula:

donde:

N = número resultante base 10

j = dígitos fraccionarios

k = dígitos enteros menos 1

x = dígito en la posición i del número

$$N_{10} = \sum_{i=k}^{-j} n^i x_i$$

- Ejemplo: 100011.1012

...	x_2	x_1	x_0	.	x_{-1}	x_{-2}	...
...	n^2	n^1	n^0		n^{-1}	n^{-2}	...

$$N_{10} = \sum_{i=5}^{-3} 2^i x_i = 2^5(1)+2^4(0)+2^3(0)+2^2(0)+2^1(1)+2^0(1)$$
$$+2^{-1}(1)+2^{-2}(0)+2^{-3}(1) = 32+2+1+0.5+.125 = 35.625$$

¡Cuidado!

- No existe un overflow para enteros
  - Cuando una variable entera se sale del rango, su aritmética no genera avisos
    - No hay mensaje de error o excepción
- En tipo byte del 127 sigue el -128

"Overflow can be silent killer of integer aritmetic"

Ronald Mak, autor del libro "Java Number Cruncher. The Java Programmer's Guide To Numerical Computing"

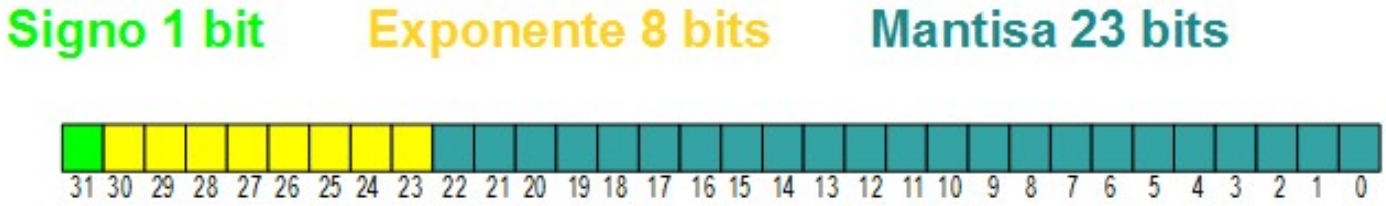
Números de punto flotante

- Declaración
  - float: 32 bits
    - Valor positivo más grande 3.4028234663852886E38f
    - Valor positivo más pequeño 1.401298464324817E-45f
  - double : 64 bits
    - Valor positivo más grande 1 .79769313486231 57E308
    - Valor positivo más pequeño 4.9E-324
- Constantes
  - float
    - 1.5f 1.5F
  - Double
    - 1.5 1.5d 1.5D 1.5e-7 1.5E4

Representación interna

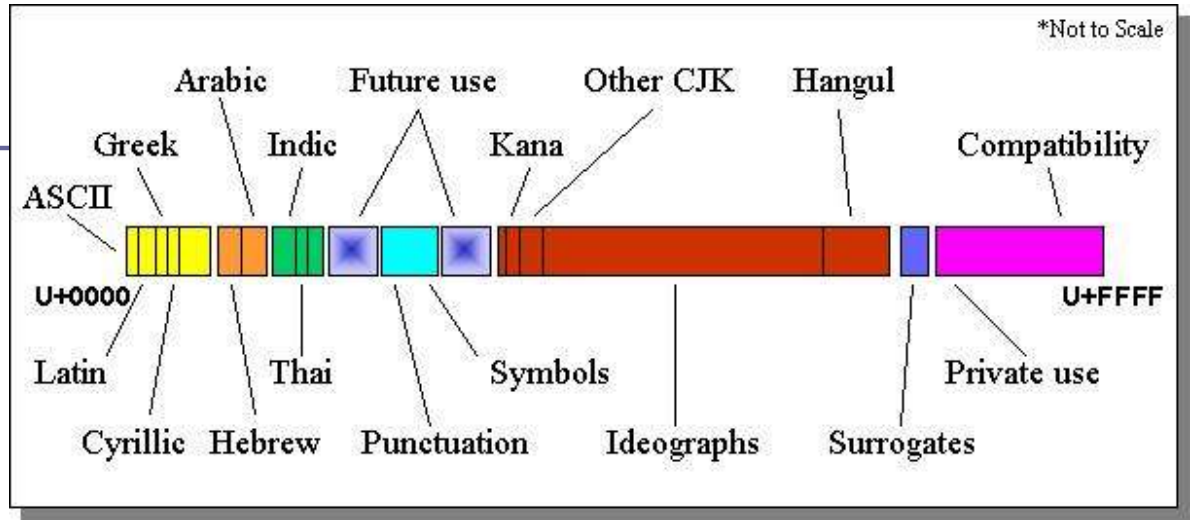
- Usa el estándar IEEE 754 aceptado en 1985
- Usa la notación científica base 2 Ax2N

float

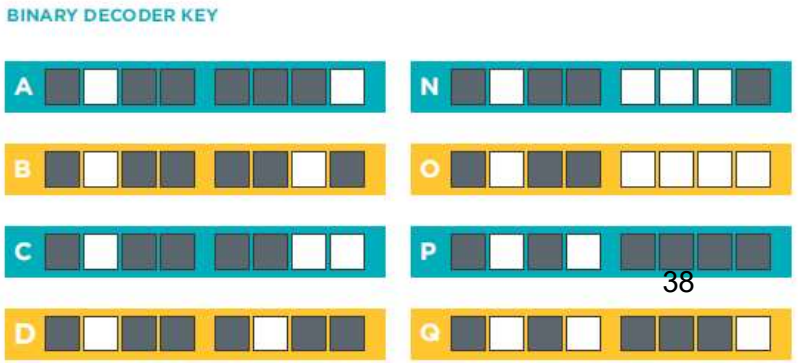


double

Textual



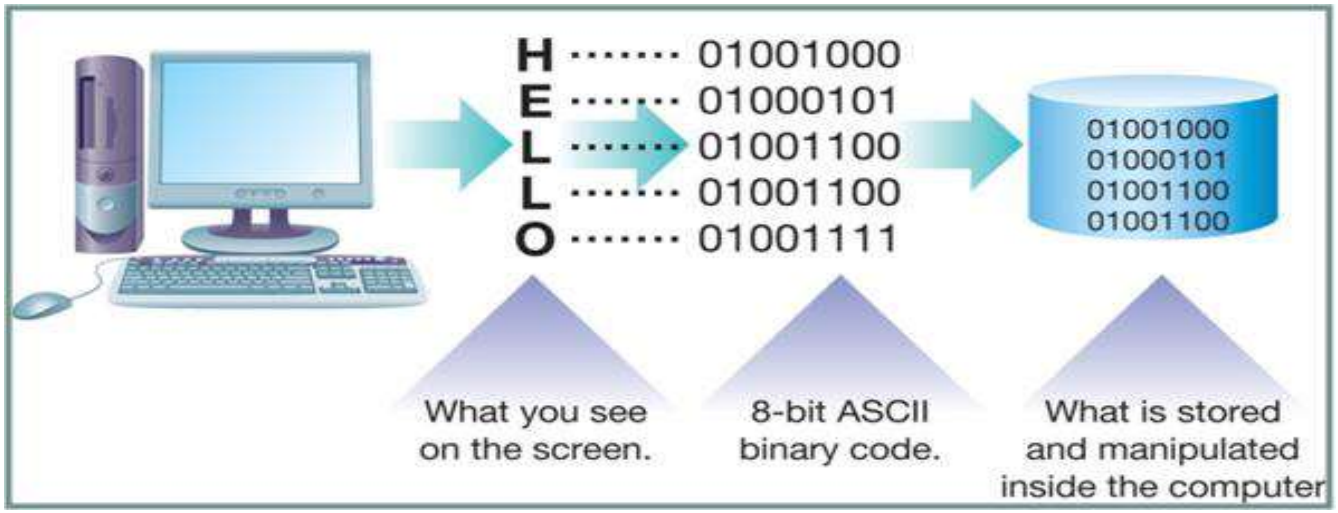
- Declaración
  - char 16 bits
- Constantes
  - Entre comilla simple
    - `'a'  '\t'  '\u3A21'`
- Representación
  - Unicode
    - Contiene más de 30,000 caracteres
    - El código ASCII es un subconjunto
      - 7 bits
      - 8 bits
        - Rango: \u0000 a \u00FF



Caracteres especiales

- Los símbolos y letras se almacenan como números
  - Se pueden sumar con aritmética entera

```
char caracter='0';
caracter = caracter + 5;
for(char c=97; c<=122; c++)
```



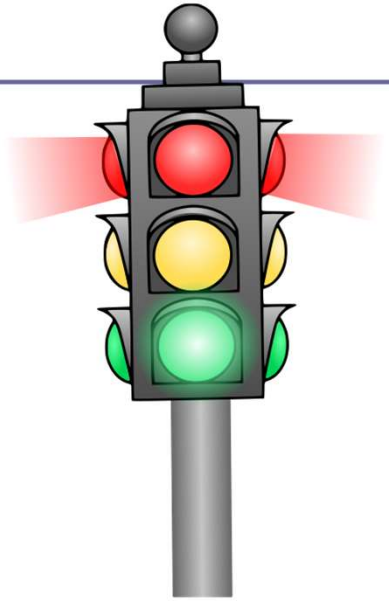
Caracteres especiales (cont)

- Emplea
  - Secuencias de escape para caracteres especiales
  - El unicode en hexadecimal '\uHHHH', para almacenar o imprimir cualquier caracter

Caracter	Escape	Dec	Hex
'\n'	nueva línea	13 10	0D 0A
'\r'	retorno	13	0D
'\t'	tabulador	9	09
'\b'	backspace	8	08
'\\'	diagonal inv.	92	5C
'\''	comilla simple	39	27
'\"'	comilla doble	34	22

Lógicos

- Declaración
  - boolean
  - 1 bit
- Constantes
  - true
  - false



```
boolean par;
par= numero%2 == 0;
```


```
boolean continuar;
if(continuar) ok();
else salir();
```

Expresión y operador de asignación

- Expresión
  - Es un elemento de un programa que toma un valor Consiste de uno o varios identificadores y/o constantes que pueden estar unidos por operadores
- Asignación =

```
variable = expresión;
```

```
var1 = var2 = ... = varN = expresión;
```

- Asignación simplificada

```
a=b=c=d=0;
```

  - Realiza una operación
    - *= /= %= += -= <<= >>= >>>= &= |= ^=

```
a+=b; => a=a+b;
```

Operadores aritméticos

- Enteros
  - -
  - * / %
  - + -
- Reales
  - -
  - * /
  - + -
- Asociatividad
  - Izquierda a derecha
  - Se puede romper la precedencia con paréntesis

```
7  / 2  = 3

7 % 2 = 1

7. / 2. =
3.5
```

Ejemplo

```
- (3 + 5) + 8 * 4 / 5 / 2 - 7
   - 8 + 8 * 4 / 5 / 2 - 7
    -8 + 8 * 4 / 5 / 2 - 7
      -8 + 32 / 5 / 2 - 7
         -8 + 6 / 2 - 7
           -8 + 3 - 7
              -5 - 7
               -12
```

$$\frac{1}{b+2}+b-5x-c\left(f-\frac{a}{z}\right)$$

1. / (b+2.) + b – 5*x -c*(f-a/z)

Acatlán
Programa de Matemáticas Aplicadas y Computación

Operadores incremento y decremento

- Incremento en 1
  - ++
- Decremento en 1
  - --
- Uso
  - Prefijo
    - ++var --var
    - Primero inc/dec y después asigna
  - Postfijo
    - var++ var--
    - Primero asigna y después inc/dec

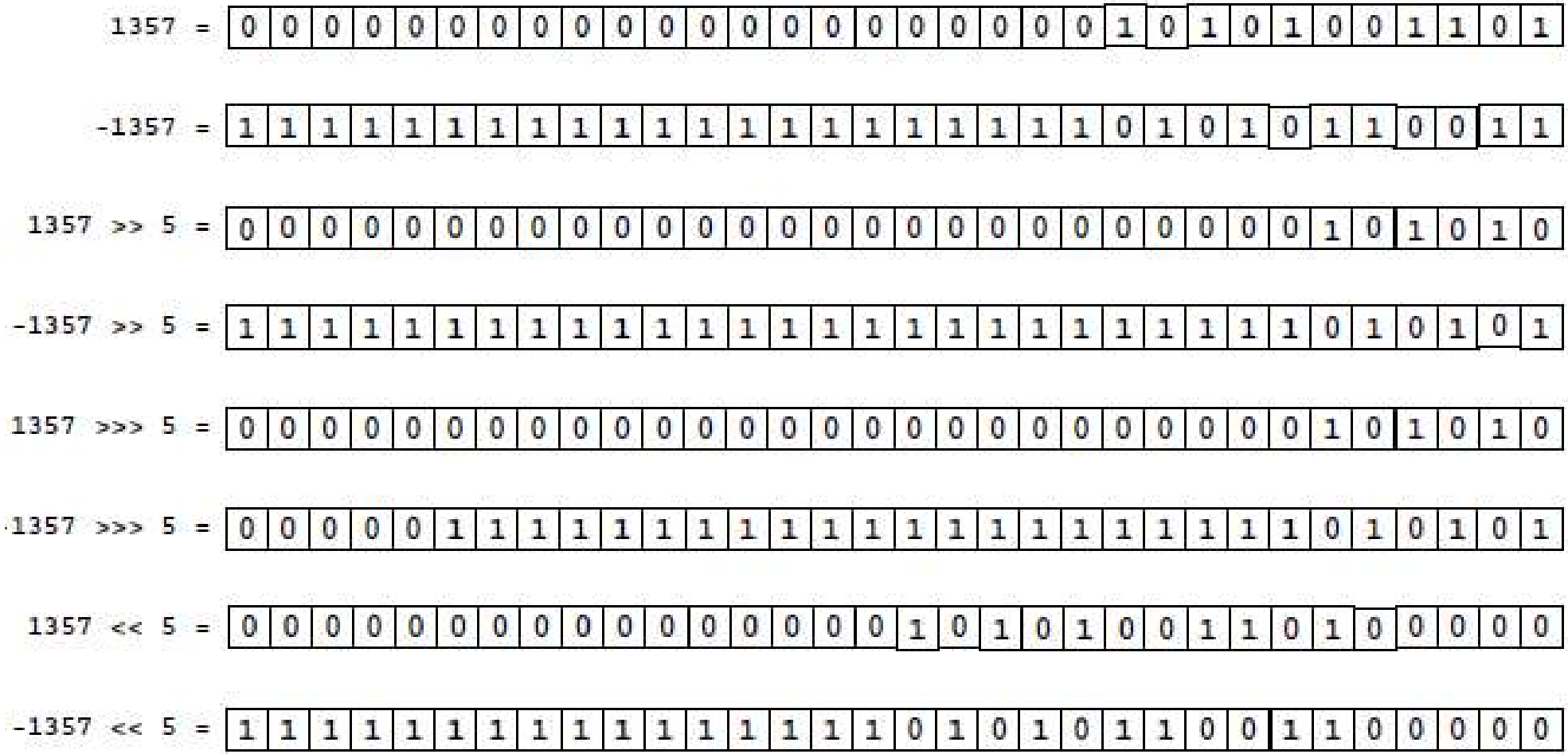
```
a++; => a=a+1;

a--; => a=a-1;
```

```
int a=1, b;
b = ++a;
=> a=2 y b=2
```

```
int a=1, b;
b = a++;
=> a=2  y  b=1
```

Ejemplo recorrimiento de bits



Operadores a nivel de bits (cont)

- Complemento
  - ~
- AND, XOR y OR
  - &
  - ^
  - |

```
~ 1 0 0 0 1 1 1 0
  0 1 1 1 0 0 0 1
```

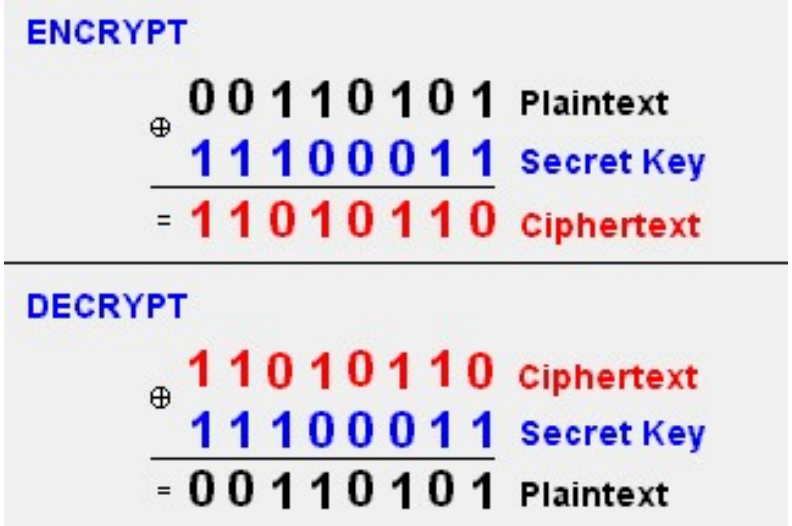
```
  1 0 0 0 1 1 1 0
& 0 0 1 1 1 0 0 0
  0 0 0 0 1 0 0 0
```

```
  1 0 0 0 1 1 1 0
^ 0 0 1 1 1 0 0 0
  1 0 1 1 0 1 1 0
```

```
  1 0 0 0 1 1 1 0
| 0 0 1 1 1 0 0 0
  1 0 1 1 1 1 1 0
```

Aplicaciones manejo bits

$$12 * 8 = 12<<3$$
$$12 / 8 = 12>>3$$



	Dot-decimal Address	Binary
IP address	192.168.5.10	11000000.10101000.00000101.00001010
Subnet Mask	255.255.255.0	11111111.11111111.11111111.00000000
Network Portion	192.168.5.0	11000000.10101000.00000101.00000000

Operadores relacionales y lógicos

- Arrojan valores Booleanos
  - 1 si es verdadero y 0 si es falso
  - Relacionales
    - ==
    - !=
    - >
    - >=
    - <
    - <=
  - Lógicos
    - !
    - &&
    - ||

¿Cuál es el resultado?

```
(25%4==3) && (4>=4) || !(6<2)
true
```

Otros operadores

- instanceof
  - Pertenencia de un objeto a una clase

```
instanceof(identificador)
```

- Condicional o ternario `? :`
  - Asigna el valor de la expresión1 si la expresión booleana es verdadera, sino asigna el valor de la expresión2

```
(expresión_booleana) ? expresión1 : expresión2
```

  - Ejemplo: `mayor = (a > b ) ? a : b ;`

Precedencia y asociatividad de operadores

	Asociatividad
()	izq a der
! ~ ++ -- +(unario) -(unario) (cast)	der a izq
* / %	izq a der
+ -	izq a der
<< >> >>>	izq a der
< <= > >= instanceof	izq a der
== !=	izq a der
&	izq a der
^	izq a der
|	izq a der
&&	izq a der
||	izq a der
<exp booleana> ? <exp1> : <exp2>	der a izq
= *= /= %= += -= <<= >>= >>>= &= |= ^=	der a izq