DIPLOMATURA EN

NUEVAS TECNOLOGÍAS

MÓDULO 1

Introducción a la Programación

Docente:

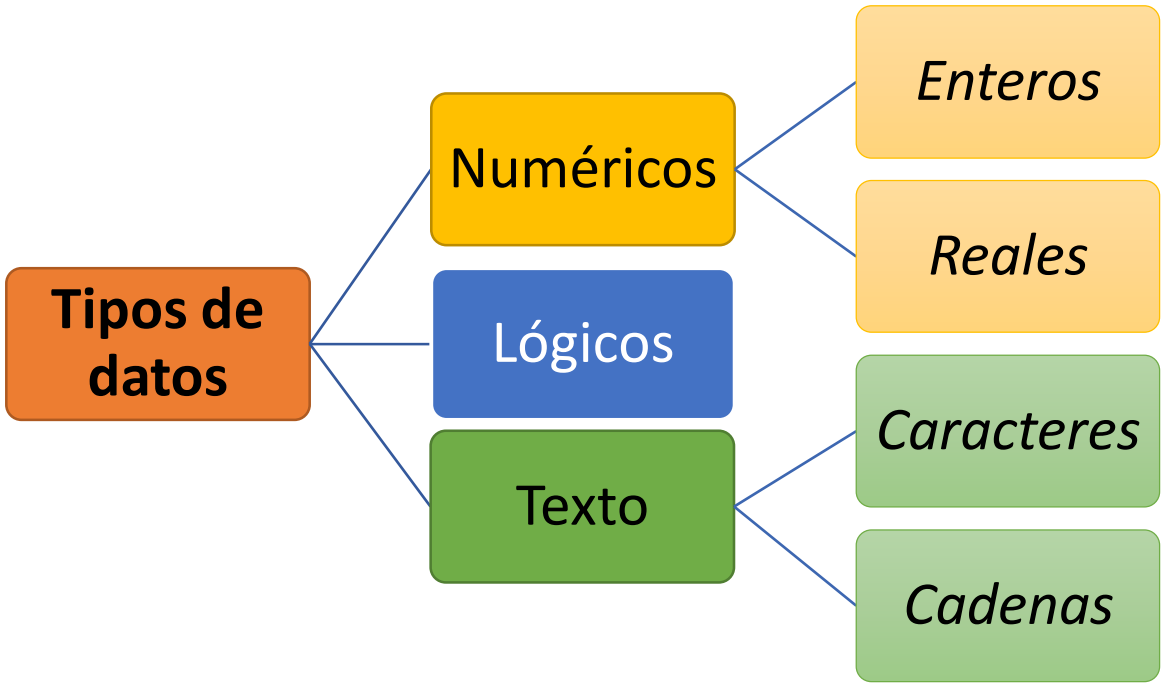
Esp. Ing. Martín Polliotto

Contenidos Semana 02:

- **2.1** Elementos básicos: **variables**
- **2.2 Operadores** aritméticos
- **2.3** Estructura de un **programa Java**
- **2.4 Entrada** y **salida** de datos
- **2.5** Estructuras **secuenciales**
- **2.6** Estructuras **condicionales**

2.1 Variables

*Una variable es **un grupo de bytes** asociado a un **nombre** o **identificador**, pero de tal forma que a través de dicho nombre se puede usar o modificar el **contenido** de los bytes asociados a esa variable.*



En **Java: byte, short, int, long, float, double, boolean, char y String**

2.1 Variables

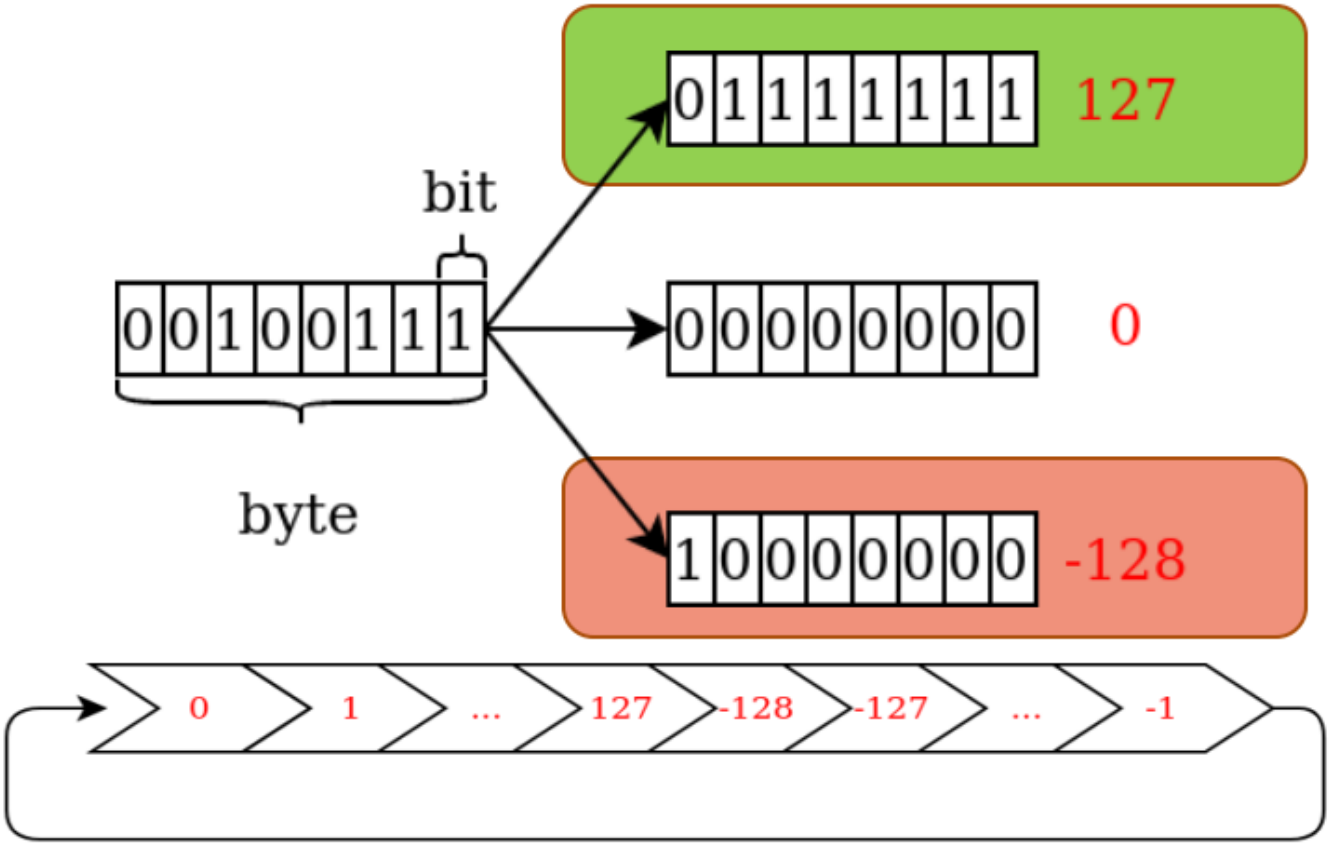
- ✓ Los nombres de variables deben empezar por letra (mayúscula o minúscula), un carácter de subrayado (_) o un signo pesos ($).
- ✓ No pueden empezar con un número
- ✓ No pueden contener puntuación, espacios o guiones (medios)
- ✓ No pueden coincidir con palabras reservadas (como class o static) del lenguaje
- ✓ Por convención se utiliza notación de camello. En general la primer letra de cada palabra se indica con mayúsculas excepto la primera. Por ejemplo: **boolean** estaVacio = true;

2.1 Variables

Nombre	Tipo	Byte	Rango de valores
boolean	Lógico	1	true ó false
char	Carácter simple	2	Un caracter
byte	Entero	1	-128 a 127
short	Entero	2	-32768 a 32767
int	Entero	4	-2.147.483.648 a 2.147.483.649
long	Entero	8	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.808
float	Numérico con coma flotante	4	$34x10^{-38}$ a $34x10^{38}$
double	Numérico con coma flotante	8	$1.8x10^{-308}$ a $1.8x10^{308}$

¿Cuánta información podemos guardar dentro de una variable?

2.1 Variables



¿Cuánta información podemos guardar dentro de una variable?

2.1 Variables

```
char c = 'A';
```

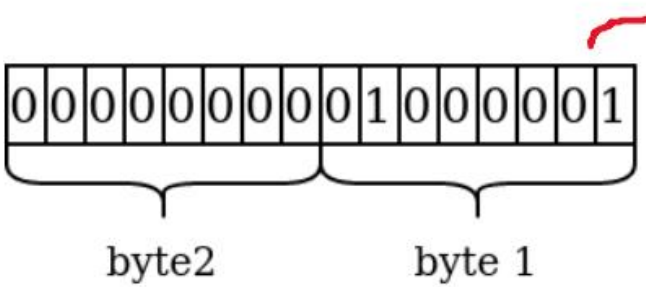


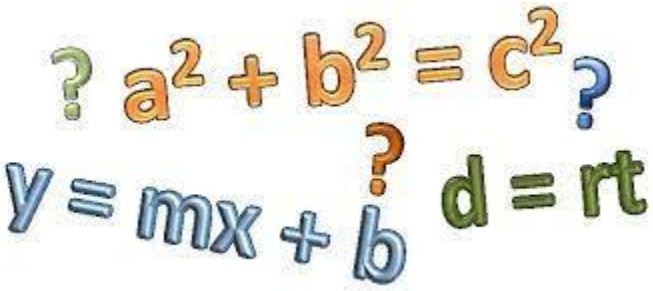
*Tabla de códigos **ASCII***

¿Y los caracteres?

¿Cuánta información podemos guardar dentro de una variable?

2.2 Operadores aritméticos

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	-4	-4
+	Suma	2.5 + 7.1	9.6
-	Resta	235.6 - 103.5	132.1
*	Producto	1.2 * 1.1	1.32
/	División (tanto entera como real)	0.050 / 0.2 7 / 2	0.25 3
%	Resto de la división entera	20 % 7	6



2.2 Operadores aritméticos

Operators	Precedence
postfix	*expr++ expr--*
unary	*++expr --expr +expr -expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

2.3 Programa Java

Por cada archivo java declaramos un **clase** mediante la palabra reservada **class**

```
public class ClasePrincipal {

  public static void main(String[] args) {

    sentencia_1;

    sentencia_2;

    // ...

    sentencia_N;
```

Para que nuestra clase pueda **ejecutarse**, necesitamos incluir el **método**:
public static void main(String args[]){}

Podemos incluir **comentarios** que serán ignorados por el compilador:
//
/**/
/**
*/

Dentro del juego de llaves del **main(), escribimos todas las sentencias de nuestro programa.**

2.4 Entrada y salida de datos

- Para mostrar **resultados** por **pantalla:**

  System.out.print().

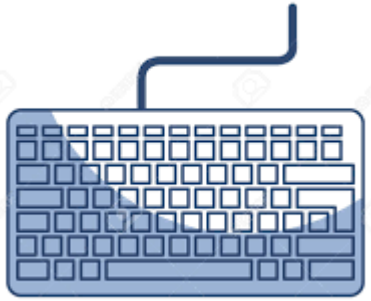
- Una variante **println():** muestra tanto una cadena de caracteres como el valor de las variables.
- Java también proporciona secuencias de escape, es decir, una combinación de la barra invertida \ . Por ejemplo:

  \n \t \' \"

- **Para mostrar una cadena junto con una variables:**

```
int a = 2;
System.out.println("Hola Java " + a);
```

2.4 Entrada y salida de datos



Usando clase de **java.io.*;**

```
BufferedReader lector = new BufferedReader(new
InputStreamReader(System.in));

int num = Integer.parseInt(lector.readLine());
```

Necesitamos agregar al método **main() throws IOException**

- Float.parseFloat();
- Double.parseDouble();
- Long.parseLong();
- Clases incluidas en el paquete **java.lang.*;**

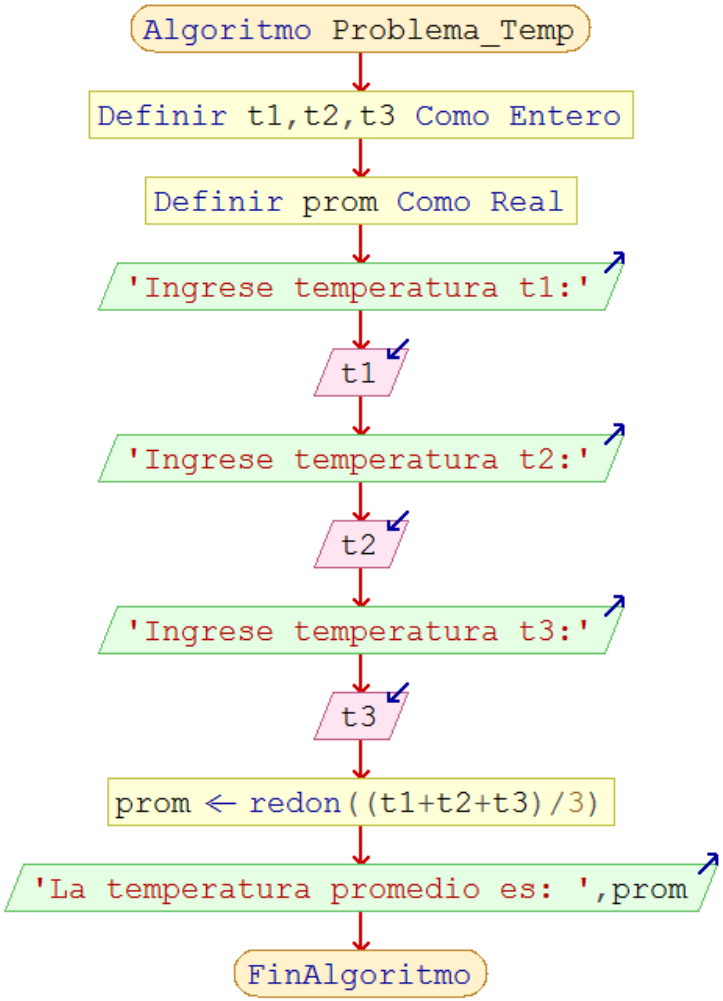
Usando **java.util.Scanner**;

```
Scanner lector = new Scanner(System.in);
int num = lector.nextInt();
```

NO Necesitamos agregar nada al método **main()**

- lector.nextFloat();
- lector.nextDouble();
- Lector.nextLong();

2.5 Estructuras **secuenciales**



En pseudocódigo:

Algoritmo Problema_Temp

Definir t1,t2,t3 Como Entero

Definir prom Como Real

Escribir 'Ingrese temperatura t1:'

Leer t1

Escribir 'Ingrese temperatura t2:'

Leer t2

Escribir 'Ingrese temperatura t3:'

Leer t3

prom <- redon((t1+t2+t3)/3)

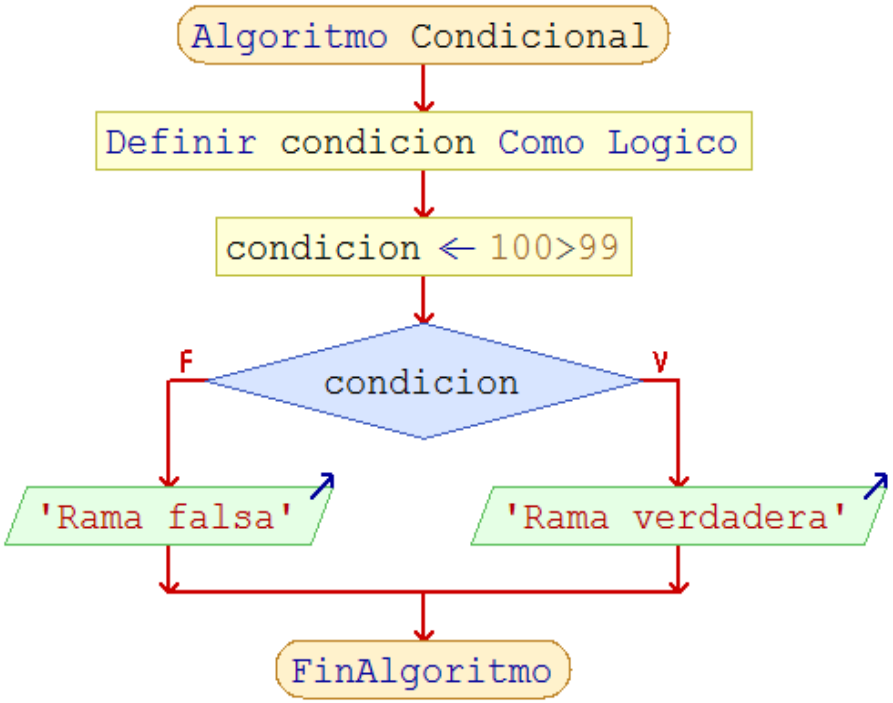
Escribir 'La temperatura promedio es: ',prom

FinAlgoritmo

2.5 Estructuras **secuenciales**


```
Vamos a NetBeans!
```

2.6 Estructuras **condicionales**



Sintaxis **Java:**

if (**expresión lógica**) {
    instrucciones de la rama **verdadera**
} else {
    instrucciones de la rama **falsa**
}

2.6 Estructuras **condicionales**

Operadores lógicos

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
!	Negación - NOT (unario)	`!false` `!(5==5)`	`true` `false`
||	Suma lógica con cortocircuito: si el primer operando es `true` entonces el segundo se salta y el resultado es `true`	`true || false` `(5==5)||(5<4)`	`true` `true`
&&	Producto lógico con cortocircuito: si el primer operando es `false` entonces el segundo se salta y el resultado es `false`	`false && true` `(5==5)&&(5<4)`	`false` `false`

Operadores relacionales

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	igual que	`7 == 38`	`false`
!=	distinto que	`'a' != 'k'`	`true`
<	menor que	`'G' < 'B'`	`false`
>	mayor que	`'b' > 'a'`	`true`
<=	menor o igual que	`7.5 <= 7.38`	`false`
>=	mayor o igual que	`38 >= 7`	`true`

2.6 Estructuras **condicionales**

Operador ternario ?:

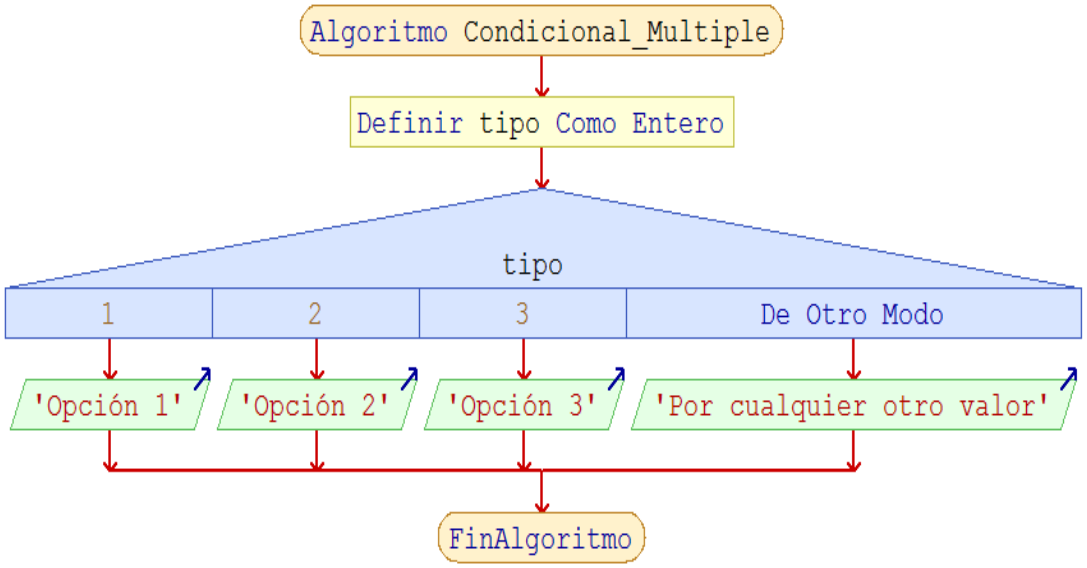
int entrada = -1;

int valor = entrada > 0 ? entrada : 0;

Sistem.out.println(valor);

Condicional múltiple:

switch (expresión_entera) {
case (valor1) : instrucciones_1; [**break**;]
case (valor2) : instrucciones_2; [break;]
...
case (valorN) : instrucciones_N; [break;]
default: instrucciones_por_defecto;
}



2.5 Estructuras **secuenciales**

Vamos a NetBeans de nuevo!

GRACIAS!