Resolución de Problemas y Algoritmos

Tipos Primitivos y Expresiones

Facultad de Informática
Universidad Nacional del Comahue

2022

Temario

Temario

Tipos de Datos

¿Qué es un tipo de dato?

El tipo de dato de una variable determina los valores admisibles que esta puede asumir, además de las operaciones que se puedan realizar con ella.

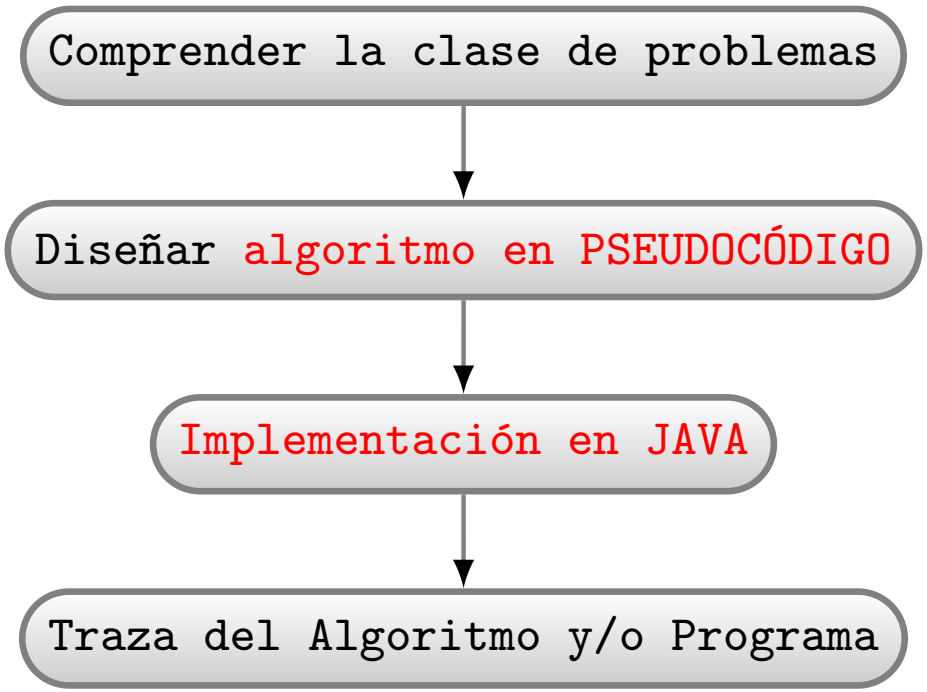
- Tipos Primitivos
- Tipos definidos por el Usuario

Tipos de Datos

Diseño de Algoritmos. Tipos de Datos Primitivos en Pseudocódigo

Hemos explicado en la clase anterior que trabajaremos con 4 tipos de datos primitivos en pseudocódigo: ENTERO, REAL, CARACTER, LóGICO.

```
ALGORITMO nombreAlgoritmo() RETORNA ∅
        | ENTERO edad
        | REAL altura
        | CARACTER inicial
        | LÓGICO aprobo
        | edad ← 5
        | altura ← 1.89
        | inicial ← 'A'
        | aprobo ← true
        | ....
FIN ALGORITMO nombreAlgoritmo
```


Comprender la clase de problemas
Diseñar algoritmo en PSEUDOCÓDIGO
Implementación en JAVA
Traza del Algoritmo y/o Programa

Primitivas e Instrucciones

¡Importante Pseudocódigo!

- Todas las primitivas o directivas en Pseudocódigo las escribiremos en mayúscula.
- Las primitivas que vamos a utilizar son: ALGORITMO, FIN ALGORITMO, RETORNA, ENTERO, REAL, CARACTER, LÓGICO, LEER, ESCRIBIR, SI-ENTONCES-SINO-FIN SI, SEGUN-FIN SEGUN, MIENTRAS-HACER-FIN MIENTRAS, REPETIR-HASTA, PARA-DESDE-HASTA-FIN PARA.

¡Importante JAVA!

- Toda instrucción en Java finaliza con un punto y coma.
- Todas las instrucciones en java se escriben por lo general en minúscula.
- Las instrucciones que vamos a utilizar son: int, double, char, boolean, if-else, switch, while, for, procedure, main, etc.

```
ALGORITMO nombreAlgoritmo() RETORNA ∅
        | ENTERO edad
        | REAL altura
        | CARACTER inicial
        | LÓGICO aprobo
        | edad ← 5
        | altura ← 1.89
        | inicial ← 'A'
        | aprobo ← true
        | ESCRIBIR("La edad es:" + edad)
        | ESCRIBIR("La inicial es:" + inicial)
        | ESCRIBIR("La altura es:" + altura)
        | ESCRIBIR("La valor de aprobo es:" + aprobo)
        | ....
FIN ALGORITMO nombreAlgoritmo
```

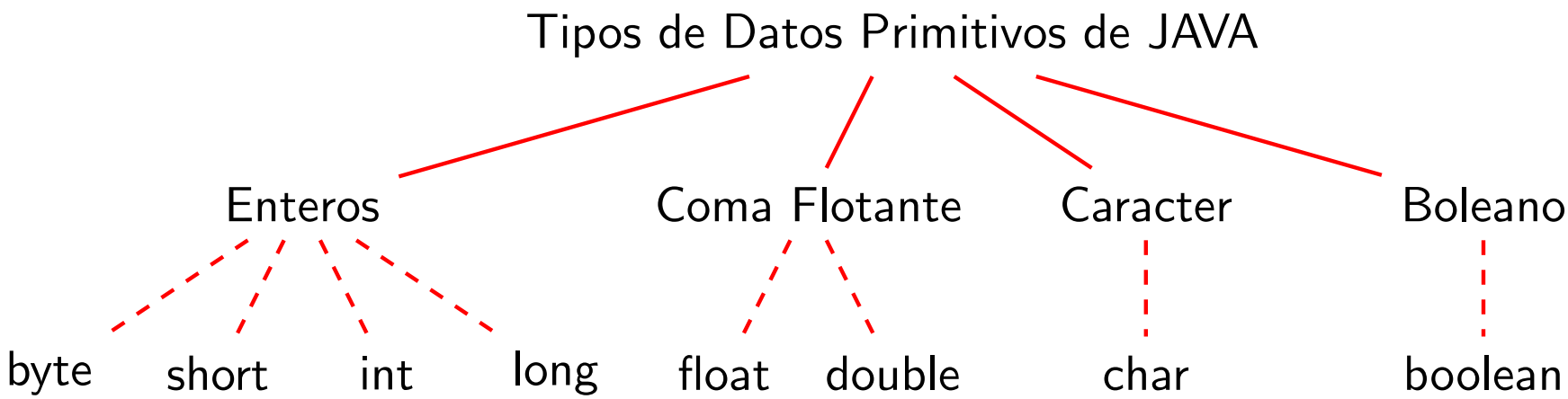
Tipos de Datos

Diseño de Algoritmos. Tipos de Datos Primitivos en Pseudocódigo

```
public class PrimerPrograma
{
// Parte principal de programa
public static void main( String[] args )
{
// Declaración de variables
 int edad;
 double altura;
 char inicial;
 boolean aprobo;
 edad = 5;
 altura = 1.89;
 inicial = 'A';
 aprobo = true;
 System.out.println("Su edad es: "+ edad);
 System.out.println("Su inicial es: "+ inicial);
 System.out.println("Su altura es: "+ altura);
 System.out.println("El valor de aprobó es: "+ aprobo);
}
}
```

Tipos de Datos

Tipos de Datos Primitivos de Java



Tipos de Datos

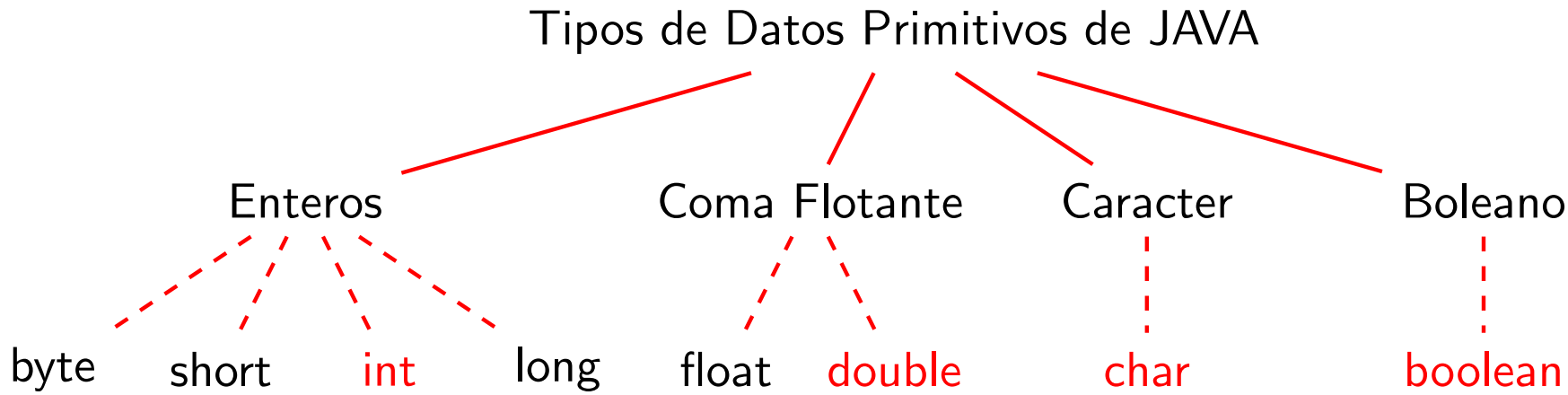
Tipos de Datos Primitivos de Java

Tipo	Tipo de Valores	Memoria Usada	Rango
byte	integer	1 byte	[-128, 127]
short	integer	2 bytes	[-32.768, 32.767]
int	integer	4 bytes	[-2.147.483.648, 2.147.483.647]
long	integer	8 bytes	[-9.223.372.036.854.775.808, 0.223.372.036.854.775.807]
float	coma flotante IEEE 754	4 bytes	[+-3.40282347 x 10^{+38}, +-1.40239846 x 10^{-45}]
double	coma flotante IEEE 754	8 bytes	[+-1.76769313486231570 x 10^{+308}, +-4.94065645841246544 x 10^{-312}]
char	un caracter Unicode	2 bytes	todos los caracteres Unicode
boolean	true ó false	1 bit	no se aplica

Tipos de Datos

Tipos de Datos Primitivos de Java

Un tipo primitivo es un tipo predefinido por el lenguaje y se nombra con una palabra reservada. La palabra reservada para indicar el tipo de dato entero asociado a edad es int.
Sin embargo en la práctica de RPA utilizaremos solo un caso de cada uno de los cuatro tipos de datos primitivos: int, double, char y boolean.



Temario

Variables

¿Qué es una Variable?

Nombre o etiqueta usado para identificar una ubicación en donde puede almacenarse un dato.

```
ALGORITMO nombreAlgoritmo() RETORNA ∅
        | ENTERO edad
        | REAL altura
        | CARACTER inicial
        | LÓGICO aprobó
        | .....
FIN ALGORITMO nombreAlgoritmo
```

Constantes

¿Qué es una Constante?

Una constante representa un valor que no puede ser modificado en el curso de la ejecución del programa. Cualquier intento de modificarla da lugar a un error. Normalmente, las constantes de un programa se suelen escribir en letras mayúsculas, para distinguirlas de las que no son constantes.

¡Importante! ¿Cuándo es útil definir una constante?

- Cuando sabemos que su valor lo necesitaremos en distintos lugares del programa.
- Cuando queremos evitar un error de tipeo al escribir su valor.
- Cuando su valor se mantiene inmutable en el transcurso del programa.

Constantes

```
ALGORITMO nombreAlgoritmo() RETORNA ∅
        | ENTERO PI
        | REAL radio, area
        | PI ← 3.14
        | radio ← 5.7
        | area ← PI * potencia(radio, 2)
        | ESCRIBIR(" El area del circulo es: "+ area)
FIN ALGORITMO nombreAlgoritmo
```

En JAVA una constante se declara con la palabra reservada final y se inicializa.

```
public class PrimerPrograma
{
public static void main( String[] args )
{// Declaración de variables y constantes
 final double PI=3.141592653589793;
 double area, radio;
 radio = 5.7;
 area = PI * MATH.power(radio, 2);
 System.out.println("El area del circulo es:"+area);
}
}
```

- Para definir una constante utilizamos datos o también llamadas expresiones literales como 2, 3.75, 'y', etc.
- Las constantes enteras pueden estar precedidas por un signo + ó -, pero no pueden contener decimales.
- Las constantes float con decimales necesitan estar seguidas de la letra f para indicar al compilador que no es una constante double. Ej: float num= 12.3f;
- Las constantes de punto flotante pueden ser escritas en notación científica.

Temario

Que es una asignación?

- Es la instrucción que nos permite asociar un valor a una variable, i.e., nos permite almacenar un dato en una variable.
- Todo algoritmo puede verse como una composición de asignaciones. Su sintáxis es:
- En Pseudocódigo una asignación se representa con el símbolo $\leftarrow$.
- En JAVA con el signo igual ($=$).

X $\leftarrow$ E donde X es una variable y E puede ser:

- Un valor;
- Una expresión;
- Otra variable, la cual cuenta con un valor.

```
ALGORITMO nombreAlgoritmo() RETORNA ∅
        | ENTERO edadPablo, edadJuan, edadMarisa
        | edadPablo ← 3
        | edadPablo ← edadPablo + 1
        | edadJuan ← edadPablo
        | edadMarisa ← 3 * edadJuan
        | ESCRIBIR("La edad de Marisa es: "+ edadMarisa)
FIN ALGORITMO nombreAlgoritmo
```

EdadPablo	**EdadJuan**	**Marisa**	**Salida**
3	4	12	La edad de Marisa es: 12
4			

Expresiones

Sentencias de Asignación

- En cambio en Java la asignación se representa con un signo igual (=).
- Es utilizada para asignar un valor a una variable. Ej:

$$\text{contador} = 42;$$

- El signo igual es un operador de asignación
- Sintaxis

$$\text{variable} = \text{expresión};$$

Que es una expresión?

Una expresión puede ser:

- una variable,
- un literal o una constante (ej: un número),
- Una combinación de variables y literales, utilizando operadores(ej + y -)

```
cantidad = 3.99;
inicial = 'W';
x = x + 1;
huevosPorCanasta= huevosPorCanasta -2;
puntaje = cartasBasto + cartasEspada + cartasOro + cartasCopa;
```

¿Como se evalua la expresión?

Se evalúa primero la expresión del lado derecho del operador de asignación (=). Luego, el resultado se usa para inicializar el valor de la variable a la izquierda del operador de asignación.

Los operadores de asignación pueden combinarse con operadores aritméticos (+,-, *, /, %).

Temario

- Una variable declarada, que no se le ha dado un valor se dice no-inicializada.
- Las variables primitivas no-inicializadas pueden tener un valor por defecto.
- Es una buena práctica no dejar las variables sin inicializar.
- Para protegernos contra una variable no-inicializada (y mantener al compilador feliz), le asignamos valor en el momento de declararla.
- Ejemplos:

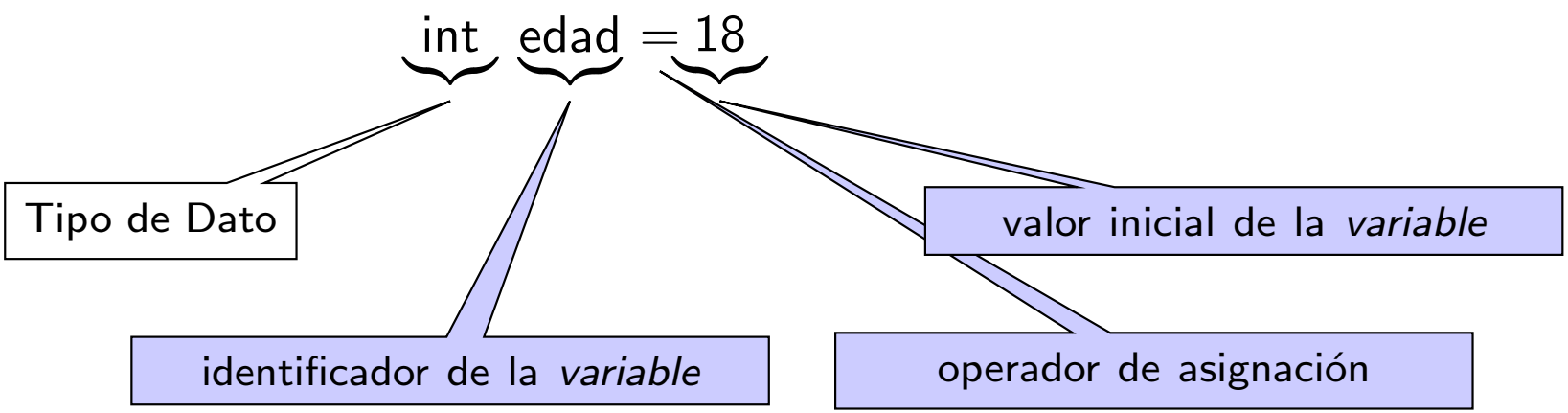
```
int contador = 0;
char grado = 'A'; // default es una A
```

- Sintáxis

```
type variable_1 = expression_1, variable_2 = expression_2,...;
```

Expresiones

Compatibilidad de Asignaciones



Temario

Java Coding Style Guide[3] recomienda

Los nombres de identificadores que no son constantes deberian utilizar el estilo *infixCaps*. Comenzar con una letra minúscula, y luego para cualquier palabra subsecuente la inicial en mayúscula, asi como cualquier letra que sea parte de un acrónimo. Todos los otros caracteres del nombre deben estar en minúscula. No utilice underscores para separar palabras. Los nombres deben ser sustantivos y no frases. Ejemplos:

```
boolean resultadoParcial;
int edadPersona;
```

Temario

- Muchos lenguajes de programación utilizan el conjunto de caracteres ASCII.
- Java utiliza el conjunto de caracteres Unicode[1] que incluye al conjunto de caracteres ASCII;
- Unicode agrega caracteres de diferentes alfabetos distintos al Inglés (es probable que nosotros no los necesitemos).

[1] es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes. El término Unicode proviene de los tres objetivos perseguidos: Unique, Universal, and Uniform character enCoding.

Expresiones
ASCII/UNICODE

32	33	34	35	36	37	38	39	40	41
	!	"	#	$	%	&	'	(	)

48	...	57	...	65	...	90	...	97	...	122
0		0		A		Z		a		z

```
char respuesta = 'y';
System.out.println(respuesta);
System.out.println((int)respuesta);
```

```
y
121
```

Temario

- Las expresiones aritméticas pueden estar formadas por los operadores +, -, *,/ junto con variables o números referenciados como operandos.
- Cuando ambos operandos son del mismo tipo, el resultado es de ese tipo.
- Cuando uno de los operandos es de tipo punto flotante y el otro es un entero, el resultado es de tipo punto flotante.

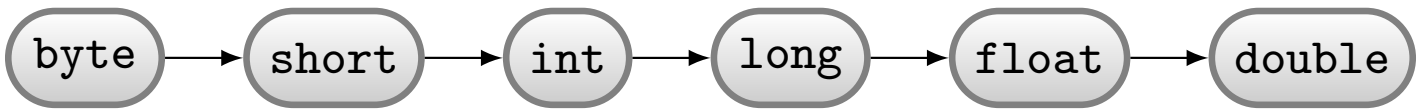
```
int horasTrabajadas= 40;
double pagoPromedio= 8.25;
```

- Si calculamos horasTrabajadas * pagoPromedio devolverá tipo double con un valor de 500.0.

- Las expresiones con dos o más operandos pueden verse como una serie de pasos, cada uno involucrando sólo dos operandos.
- El resultado de un paso produce uno de los operandos a ser utilizado en el paso siguiente.

```
balance + (balance * tasa)
```

- Si al menos uno de los operandos es un tipo punto-flotante y el resto son enteros, el resultado será de tipo punto-flotante.
- El resultado es del tipo más a la derecha de la siguiente lista según ocurra en la expresión:



- El operador de división (/) devuelve un punto-flotante cuando uno de los operandos es tipo punto-flotante.
- Cuando ambos operandos son tipo entero, el resultado es el cociente entero de la división.

25 / 10	devuelve 2
25.0 / 10	devuelve 2.5
25 / 10.0	devuelve 2.5
25.0 / 10.0	devuelve 2.5

Temario

- El operador mod (%) se utiliza con *operandos de tipo entero* para obtener el resto de la división entera.
- Ejemplo: 14 dividido 4 es 3 con un resto de 2, i.e. `14 % 4` es igual a 2.
- El operador `mod` tiene muchos usos, por ejemplo:
  - Determinar si un entero es par o impar.
  - Determinar si un entero es divisible por otro entero.

Temario

Notación Matemática	**Nombre**	**Notación Java**	**Ejemplo Java**
$=$	igual a	$==$	balance $==$ 0 respuesta $==$ 'y'
$\neq$	no igual a	!=	ingreso != impuesto respuesta != 'y'
$>$	mayor que	$>$	gastos $>$ ingreso
$>=$	mayor igual a	$>=$	puntos $>=$ 60
$<$	menor que	$<$	presion $<$ max
$<=$	menor igual a	$<=$	gastos $<=$ ingreso

Temario

- El tipo boolean es un tipo primitivo con sólo dos valores: true y false.
- Las variables booleanas pueden hacer los programas más legibles:

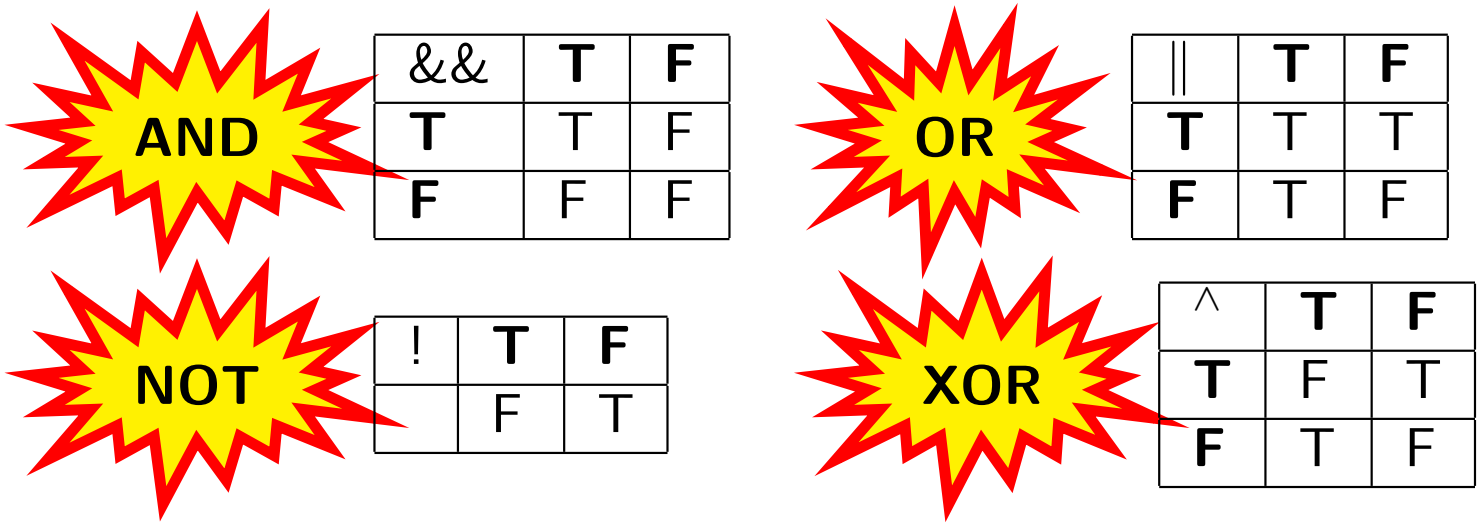
```
if(sistemasEstanOK)
```

En lugar de

```
if((temperatura <= 100) && (aceleracion >= 12000) &&
(presionCabina > 30) &&...)
```

Operación Lógica	**Operador Java**	**Sintaxis**
and	&&	$(subExpresion_1)$ && $(subExpresion_2)$
or	||	$(subExpresion_1)$ ||(subExpresion$_2$)
negación	!$(expresion\_booleana)$	!$(Expresion_1)$
or-exclusivo	^	$(subExpresión_1)$ ^ $(subExpresión_2)$

A	B	A and B
T	T	T
T	F	F
F	T	F
F	F	F



- Una variable booleana puede obtener su valor de una expresión booleana. Ejemplo:

```
booleanesPositivo= (numero > 0);
...
if(esPositivo) ...;
```

- Elegir nombres significativos como esPositivo, sistemasEstanOk, etc.

- Una variable booleana puede obtener su valor de una expresión booleana. Ejemplo:

```
booleanesPositivo= (numero > 0);
...
if(esPositivo) ...;
```

- Elegir nombres significativos como esPositivo, sistemasEstanOk, etc.

Temario

Expresiones Booleanas

Evaluación con Corto Circuito

- Generalmente sólo necesita ser evaluada una parte de una expresión booleana para determinar el valor de la expresión entera:
  - Si el primer operando asociado con un $\|$ es *true*, la expresión es *true*.
    - T $\|$ p1 $\|$ p2... $\|$ .... $\|$ .... $\Longrightarrow$ T
  - Si el primer operando asociado con un && es *falso*, la expresión es *false*.
    - F && p1 && p2... && .... && .... $\Longrightarrow$ F
- Esto se llama evaluación con corto circuito (short-circuit) o lenta (lazy).

¡Importante!

La evaluación con corto circuito no sólo es más eficiente, sino que muchas veces es esencial

- Un error en tiempo de ejecución puede producirse, por ejemplo intentando dividir por cero. Al aplicar las sentencias de la siguiente manera se evita el error de ejecución:

```
if ((number!= 0) && (sum/number> 5))
```

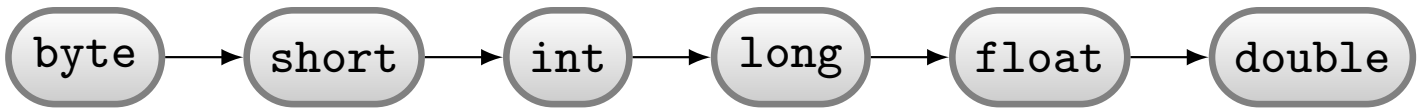
- La evaluación completa se logra sustituyendo:
  - & en lugar de &&
  - | en lugar de ||

Temario

- Java es fuertemente tipado (strongly typed).
- Ej: no puede asignarse un valor real a una variable declarada como entera.
- Sin embargo, algunas conversiones entre números son posibles.

```
double numero = 7; // es posible porque numero es de tipo double
```

- Un valor puede ser asignado a una variable de otro tipo que se encuentra hacia la derecha en la figura siguiente:



- Pero no puede hacerse con una variable que se encuentre hacia la izquierda.

Temario

¿Que es un casting de tipo?

Un casting de tipo cambia *temporalmente* el valor de una variable de un tipo declarado por otro tipo.

```
double distancia;
distancia = 9.0;
int puntos;
puntos = (int)distancia; // sin (int) daría error
```

Cuidado!

- El valor de (int)distancia es 9, pero el valor de distancia, tanto antes como después del casting, es 9.0
- El tipo de distancia NO cambia, permanece float.
- El valor distinto de cero a la derecha del punto decimal es truncado.

- Los caracteres se almacenan como enteros de acuerdo a un código especial
  - Cada carácter (letra, número, símbolo de puntuación, espacio, y tabulado) es asignado a un código entero diferente
  - Los códigos de mayúsculas y minúsculas son diferentes
  - Ej: 97 es el valor entero para ay 65 para A
- ASCII y Unicode son códigos de caracteres
- Haciendo casting de un valor char a un valor int devuelve el valor ASCII/Unicode.

```
char respuesta = 'y';
System.out.println(respuesta);
System.out.println((int)respuesta);
```

```
y
121
```

Temario

Expresiones

Operadores de Incremento y Decremento

- Utilizados para incrementar (o decrementar) el valor de una variable en 1
- Fáciles de usar, importante reconocerlos
- Operador de incremento: count++ ó ++count
- Operador de decremento: count–ó–count
- Operaciones equivalentes de incremento

$$\begin{cases} count++; \\ ++count; \\ count = count + 1; \end{cases}$$

$$\begin{cases} count--; \\ --count; \\ count = count - 1; \end{cases}$$

Expresiones

Operadores de Incremento y Decremento

- Utilizados para incrementar (o decrementar) el valor de una variable en 1
- Fáciles de usar, importante reconocerlos
- Operador de incremento: count++ ó ++count
- Operador de decremento: count–ó–count
- Operaciones equivalentes de incremento

$$\begin{cases} count++; \\ ++count; \\ count = count + 1; \end{cases}$$

$$\begin{cases} count--; \\ --count; \\ count = count - 1; \end{cases}$$

¡Atención!

En otro tipo de sentencias, ++m y m++ no son equivalentes

- Después de ejecutar

```
int m = 4;
int result= 3 * (++m);
```

El valor de result es 15 y el de m es 5 porque m se incrementa antes de ser evaluado result.

- Después de ejecutar

```
int m = 4;
int result = 3 * (m++);
```

El valor de result es 12 y el de m es 5 porque m se incrementa después de ser evaluado result.

Expresiones

Operadores de Incremento y Decremento

- Con esta definición de variables:

```
int n = 3;
int m = 4;
int result;
```

- ¿Qué valores tendrán result y m después de la ejecución de cada línea?

1.
```
result= n * ++m; // pre-incremento m
// equivalente a: m = m + 1;
//                result = n * m;
```

2.
```
result= n * m++; // post-incremento m
// equivalente a: result = n * m;
//                m = m + 1;
```

3.
```
result= n * --m; // pre-decremento m
```

4.
```
result= n * m--; // post-decremento m
```

Expresiones

Operadores de Incremento y Decremento

1. ```
   m = m + 1; //m = 4 + 1 = 5
   result= n * m; //result= 3 * 5 = 15
   ```

2. ```
   result= n * m; //result= 3 * 4 = 12
   m = m + 1; //m = 4 + 1 = 5
   ```

3. ```
   m = m -1; //m = 4 -1 = 3
   result= n * m; //result= 3 * 3 = 9
   ```

4. ```
   result= n * m; //result= 3 * 4 = 12
   m = m -1; //m = 4 -1 = 3
   ```

Temario

- Los paréntesis indican el orden en el cual las operaciones aritméticas deben ser ejecutadas

¡Atención!

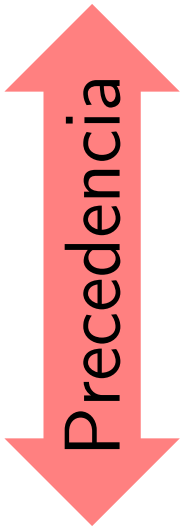
```
resultado1 =(costo + tasa) * descuento; // aqui es necesario
resultado2 = costo + (tasa * descuento) // aqui es opcional
```

- Sin paréntesis, una expresión es evaluada de acuerdo a las reglas de precedencia.
- Aún cuando los paréntesis no son necesarios, pueden ser usados para hacer el código más claro:

```
balance + (tasaInteres * balance)
```

Expresiones

Reglas de Precedencia de Operadores Matemáticos



MAYOR

1.	operadores unarios: +, -, ++, - -
2.	operadores aritméticos binarios: *, /, y %.
3.	operadores ariméticos binarios: + y -.

MENOR

- Los operadores aritméticos binarios*, /, y %, tienen precedencia más bajaque los operadores unarios+, -, ++, –, y !, pero tienen precedencia más altaque los operadores aritméticos binarios + y -.

- Operadores binarios con igual precedencia: el operador de la izquierda actúa antes que el (los) operador(es) de la derecha.
  - En `a + b - c` se calcula primero `(a+b)` y luego se le resta `c`.
  - En `a / b * c` se calcula primero `(a/b)` y el resultado se multiplica por `c`.

- Operadores binarios con igual precedencia: el operador de la izquierda actúa antes que el (los) operador(es) de la derecha.
  - En `a + b - c` se calcula primero `(a+b)` y luego se le resta `c`.
  - En `a / b * c` se calcula primero `(a/b)` y el resultado se multiplica por `c`.

Expresiones

Ejemplos de Precedencia

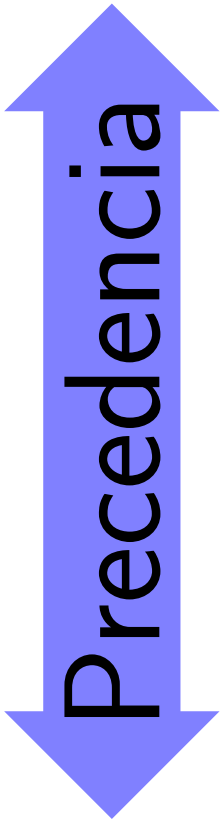
Expresión Matemática Ordinaria	Expresión Java	Expresión equivalente con Paréntesis
$rate^2 + delta$	rate*rate + delta	(rate*rate) + delta
$2(salary + bonus)$	2*(salary + bonus)	2*(salary + bonus)
$\frac{1}{time+3mass}$	1/(times + 3*mass)	1/ (time + (3*mass))
$\frac{a-7}{t+9v}$	(a - 7)/(t + (9*v))	(a - 7)/(t + (9*v))

- ¿En qué orden se ejecutará la siguiente operación?

```
score < min/2 - 10 || score > 90;
```

Expresiones

Reglas de Precedencia



MAYOR

MENOR

Operadores Unarios	1.	+, -, ++, - -
Operadores aritméticos binarios	2.	*, /, %.
	3.	+, -
Operadores de comparación	4.	>, <, >=, <=
	5.	== !=
Operadores booleanos	6.	&
	7.	|
	8.	&&
	9.	||

Expresiones

Ejemplo de Reglas de Precedencia

```
score < min/2 - 10 || score > 90;
```

Operadores aritméticos binarios	2.	*, /, %.
	3.	+, -
Operadores de comparación	4.	>, <, >=, <=
	5.	== !=
Operador Booleano	9.	$\|\|$

El orden esta determinado de la siguiente forma:

$$\underbrace{\underbrace{\underbrace{score < \underbrace{min/2}_{1} - 10}_{2} \,\|\, \underbrace{score < 90}_{3}}_{3}}_{4}$$

Supongamos que min=40; score= 5, ¿cuál es el valor de verdad de la expresión?

Expresiones Booleanas

Usos de ==

- == es apropiado para determinar si dos enteros o dos caracteres tienen el mismo valor.

```
if (a == 3) ...
```

donde a es de tipo integer.

Lectura Adicional

James Gosling, Bill Joy, Guy Steele, Gilad Bracha.
The Java Language Specification Third Edition.
Addison-Wesley, 1996-2005.
http://java.sun.com/docs/books/jls/

Apuntes de Cátedra.
Apunte sobre Tipos Primitivos
Facultad de Informática, Universidad del Comahue, 2022.

Achut Reddy
Java Coding Style Guide
Server Management Tools Group
Sun Microsystems, Inc. 2000