

# Tema 3. Variables: definición, tipos y uso en Java

## Punto 2 - Parte 2: Uso de una variable Java

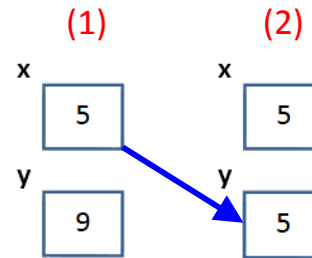
- Principios básicos: estado de una variable y su modificación. Traza de ejecución
- Asignación
  - Inicialización de variables, según su tipo y ámbito. Literales. Expresiones. Compatibilidad de tipos. Valores por defecto
  - Copia e intercambio
  - Objetos desreferenciados y Garbage Collector
- Otras operaciones sobre variables, según su tipo
  - Igualdad
  - Comparación
- Detalles, ejemplos y ejercicios con operadores de tipo primitivo: desbordamiento, compatibilidad (automática y forzosa), división (entera y real), precedencia (asociatividad por la izquierda y uso paréntesis) , operadores aritméticos no simples, operaciones con char, operadores relacionales y lógicos cortocircuitados

# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Copia (del valor) de una variable

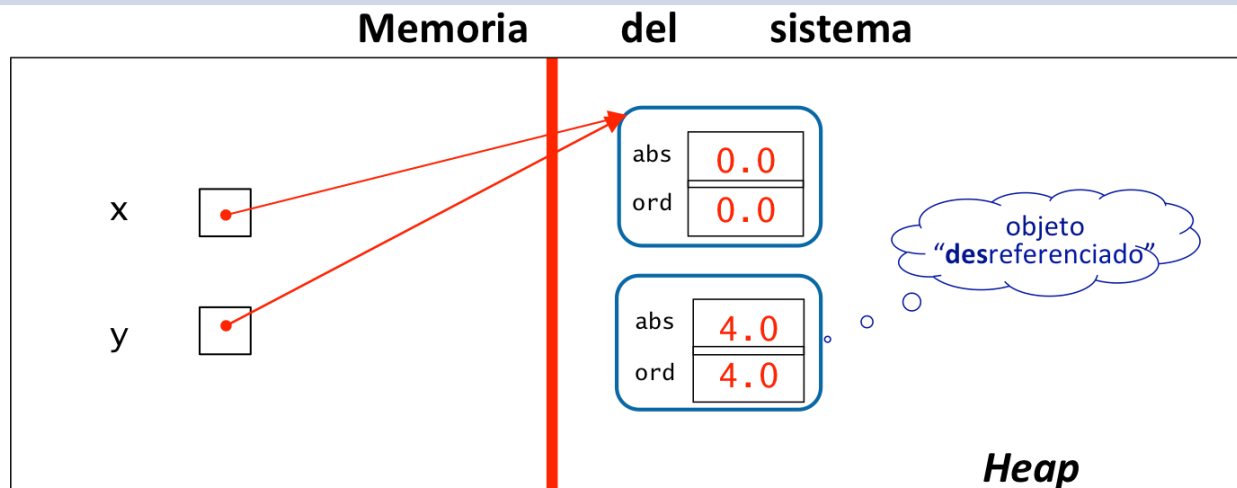
- Realiza la **traza** del código que se te propone a continuación  
¿Qué hace? **Copiar** el valor de x en y, o **copiar** x en y

```
int x = 5, y = 9;  
y = x;
```



- Realiza la **traza** del código que se te propone a continuación.  
¿y contiene una **copia** del valor de x o del objeto al que referencia x?

```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);  
y = x;
```



# Uso de una variable en Java: **Asignación**

## **Objeto desreferenciado - *Garbage Collector***

¿Qué ocurre con un objeto “**desreferenciado**”? ¿Se queda para siempre en el Heap, ocupando espacio inútilmente?

**NO**, gracias al ***Garbage Collector***

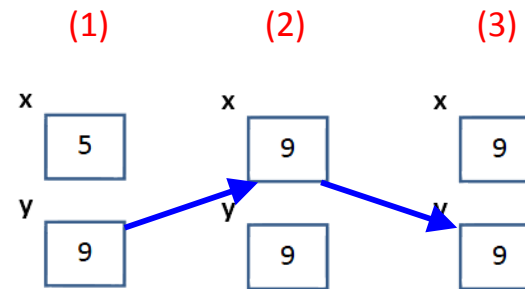
- Es un subsistema de la JVM que recupera la memoria que ocupan los objetos desreferenciados para que pueda volver a ser utilizada
- **Entra en funcionamiento automáticamente**, aunque es posible inhabilitar su funcionamiento si se desea o hacer que se ejecute mediante un método de la clase System (System.gc())
- Es frecuente en los lenguajes basados en el uso de una máquina virtual, como Java, C# o Python. En otros lenguajes, por ejemplo C++ o Ada, es necesario que el programador libere explícitamente la memoria que ha dejado de utilizarse

# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Intercambio (del valor) de 2 variables (I)

- Realiza la **traza** del código que se te propone a continuación  
¿Qué sucede? ¿Se **intercambian** los valores de x e y?

```
int x = 5, y = 9; (1)  
x = y; (2)  
y = x; (3)
```



El valor que almacena una variable se pierde cuando se le asigna uno nuevo

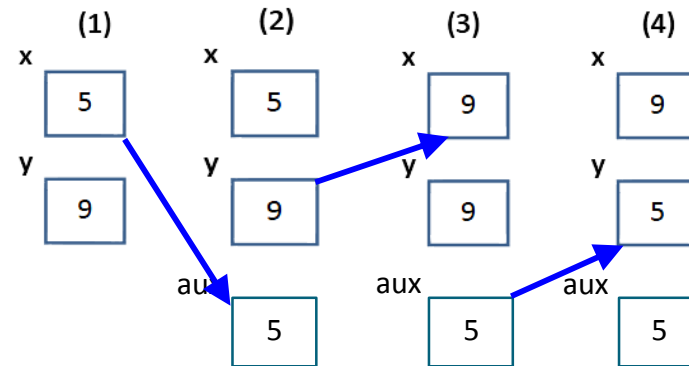
↪ Las variables de programación **NO** son variables matemáticas

# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Intercambio (del valor) de 2 variables (II)

- Realiza la **traza** del código que se te propone a continuación.  
¿Qué sucede? ¿Se **intercambian** los valores de x e y?

```
int x = 5, y = 9; (1)  
int aux = x; (2)  
x = y; (3)  
y = aux; (4)
```



# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Intercambio (del valor) de 2 variables (III)

- Realiza la **traza** del código que se te propone a continuación.  
¿Se **intercambian** los **valores** de x e y o los **objetos** que referencian?

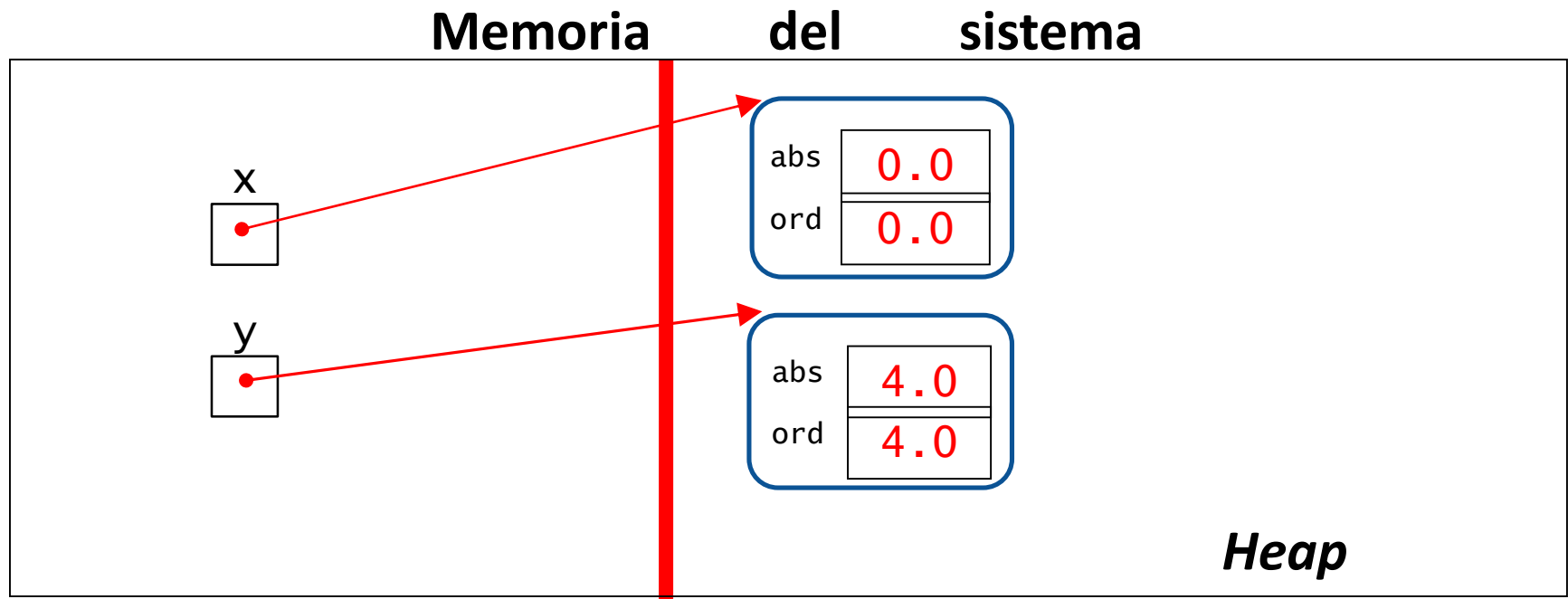
```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);
```

```
PuntoR aux = x;
```

```
x = y;
```

```
y = aux;
```

↪ 2 objetos y 2 referencias



# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Intercambio (del valor) de 2 variables (IV)

- Realiza la **traza** del código que se te propone a continuación.  
¿Se **intercambian** los **valores** de x e y o los **objetos** que referencian?

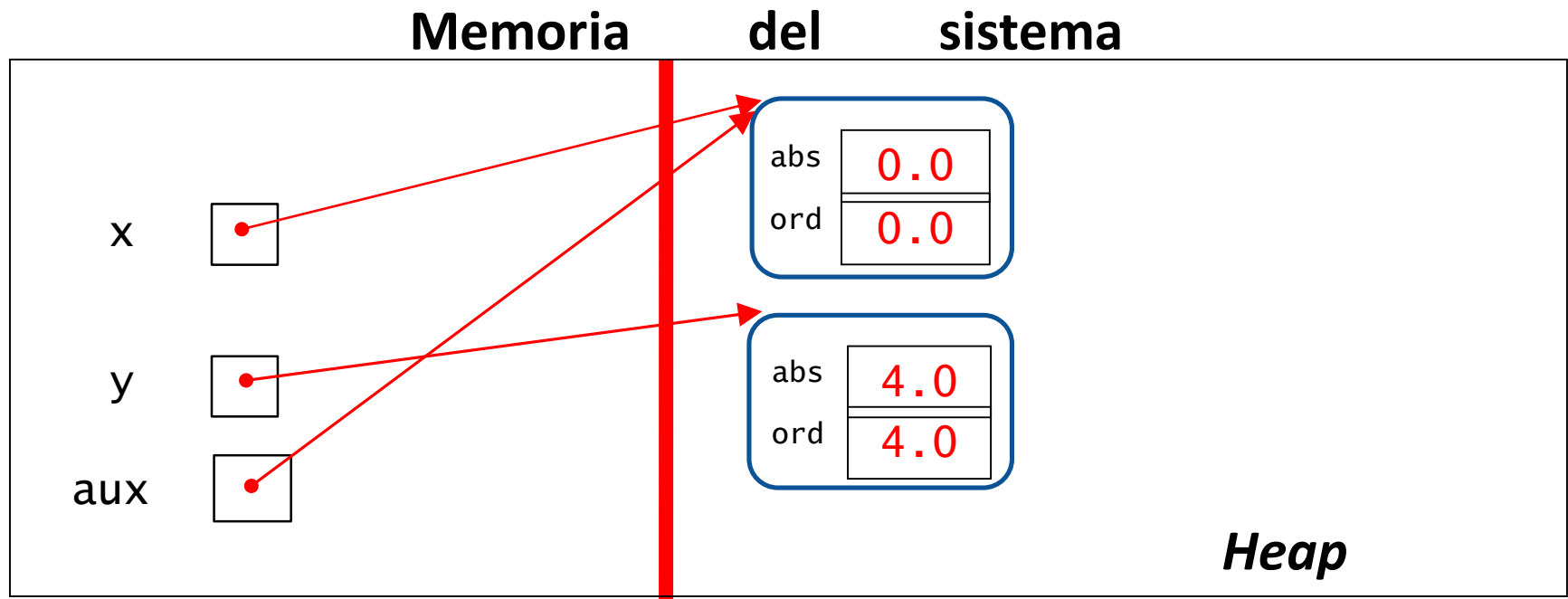
```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);
```

```
PuntoR aux = x;
```

```
x = y;
```

```
y = aux;
```

↪ 2 objetos y 3 referencias



# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Intercambio (del valor) de 2 variables (V)

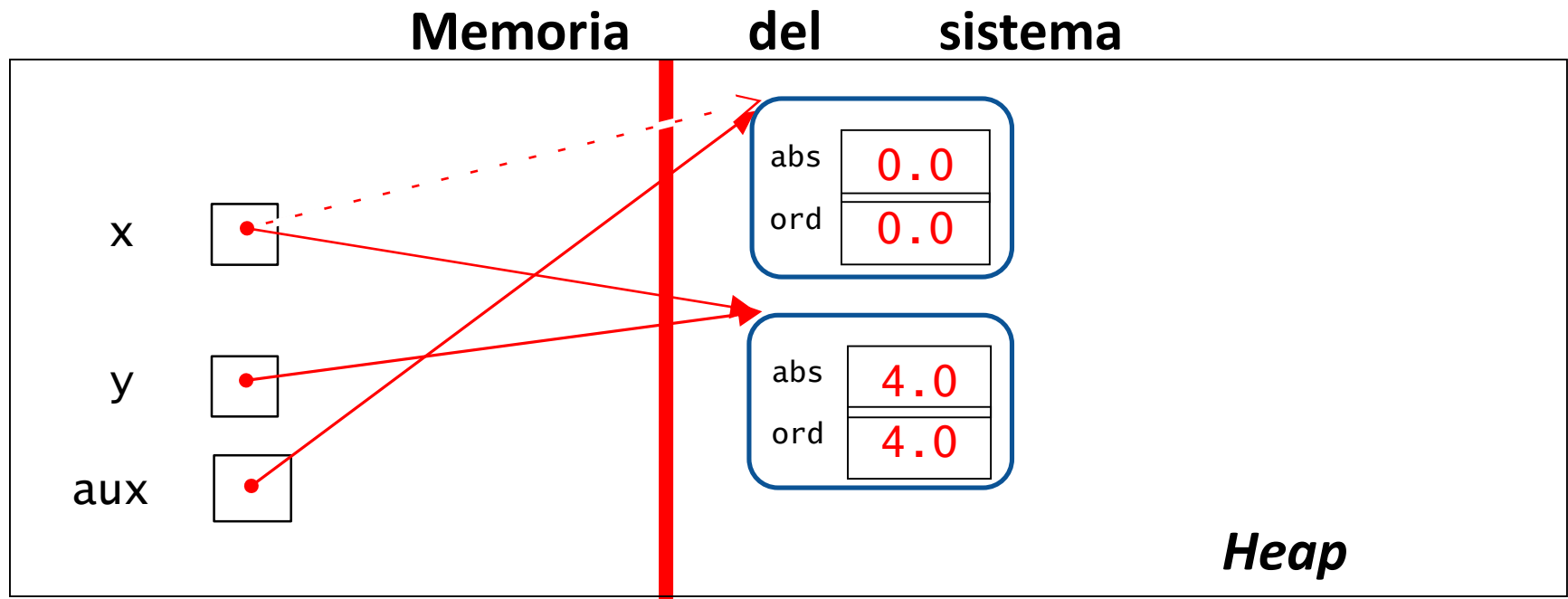
- Realiza **traza** del código que se te propone a continuación.  
¿Se **intercambian** los **valores** de x e y o los **objetos** que referencian?

```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);  
PuntoR aux = x;
```

```
x = y;
```

```
y = aux;
```

↩ Cambio de referencia





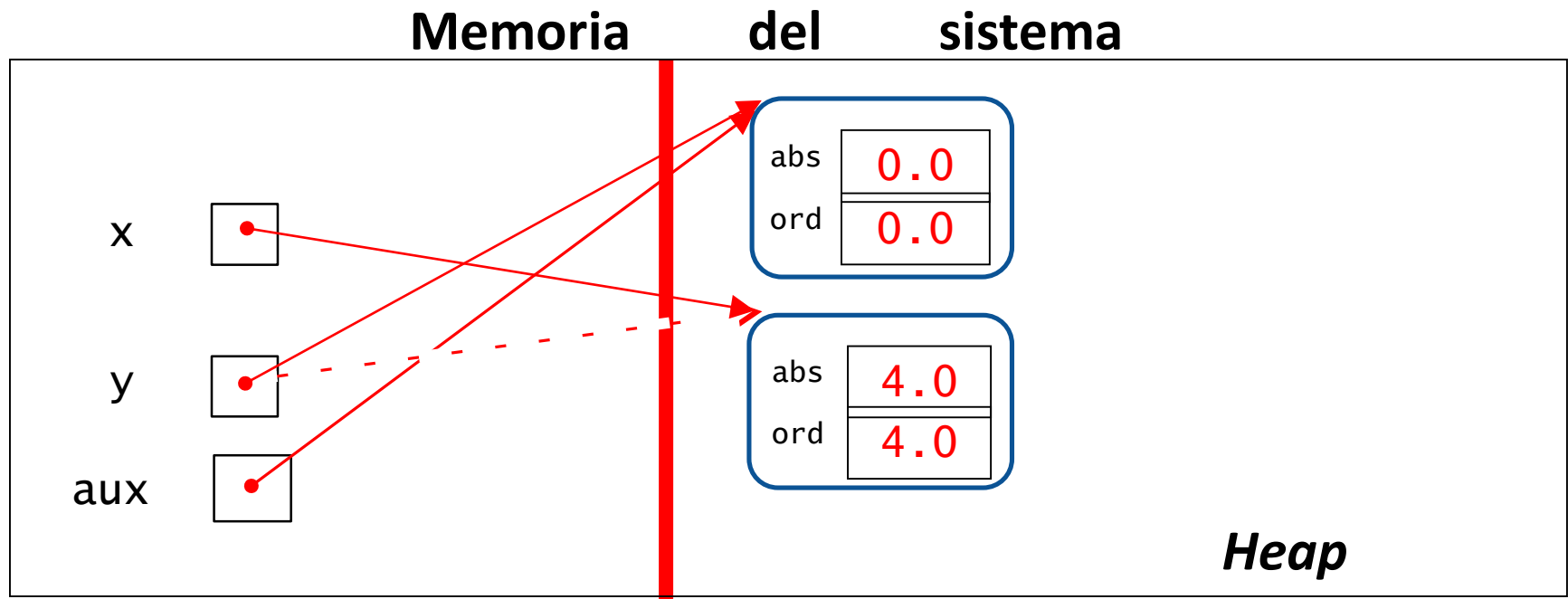
# Uso de una variable en Java: Asignación

## ¿Cuándo se usa? Intercambio (del valor) de 2 variables (VI)

- Realiza la **traza** del código que se te propone a continuación.  
¿Se **intercambian** los **valores** de x e y o los **objetos** que referencian?

```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);  
PuntoR aux = x;  
x = y;  
y = aux;
```

↪ Intercambio de referencias



# Uso de una variable en Java: Igualdad (del valor) de 2 variables (I)



BlueJ: ejemplos - Tema3

Copia las siguientes instrucciones en el **CodePad** de BlueJ y observa el resultado de su ejecución en su terminal... **¿Está claro para qué sirven los operadores == y !=?**

```
int x = 5, y = 9;  
x = y;  
  
boolean esIgual = (x == y); // op. relacional de igualdad  
System.out.println("x es igual a y? " + esIgual);  
boolean esDistinto = (x != y); // op. relacional de desigualdad  
System.out.print("x es distinto de y? " + esDistinto);
```

```
x es igual a y? true
```

```
x es distinto de y? false
```

# Uso de una variable en Java:

## Igualdad (del valor) de 2 variables (II)



BlueJ: ejemplos - Tema3

Copia las siguientes instrucciones en el **CodePad** de Bluej y observa el resultado de su ejecución en su terminal... **¿Por qué no es el mismo que el de la trasa anterior?**

```
int x = 5, y = 9;
int aux = x;
x = y;
y = aux;

boolean esIgual = (x == y);
System.out.println("x es igual a y? " + esIgual);
boolean esDistinto = (x != y);
System.out.print("x es distinto de y? " + esDistinto);
```

```
x es igual a y? false
```

```
x es distinto de y? true
```

# Uso de una variable en Java: Igualdad (del valor) de 2 variables (III)



Bluej: ejemplos - Tema3

Copia las siguientes instrucciones en el **CodePad** de Bluej y observa el resultado de su ejecución en su terminal...

**¿Por qué es el que es si los puntos (0, 0) y (4, 4) NO son iguales?**

```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);  
y = x;  
  
boolean esIgual = (x == y);  
System.out.println("x es igual a y? " + esIgual);  
boolean esDistinto = (x != y);  
System.out.print("x es distinto de y? " + esDistinto);
```

```
x es igual a y? true
```

```
x es distinto de y? false
```

# Uso de una variable en Java: Igualdad (del valor) de 2 variables (IV)



Bluej: ejemplos - Tema3

Copia las siguientes instrucciones en el **CodePad** de Bluej y observa el resultado de su ejecución en su terminal... **¿Por qué no es el mismo que el de la trasa anterior?**

```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);
PuntoR aux = x;
x = y;
y = aux;
boolean esIgual = (x == y);
System.out.println("x es igual a y? " + esIgual);
boolean esDistinto = (x != y);
System.out.println("x es distinto de y? " + esDistinto);
esIgual = (aux == x);
System.out.println("aux es igual a x? " + esIgual);
esIgual = (aux == y);
System.out.print("aux es igual a y? " + esIgual);
```

```
x es igual a y? false
x es distinto de y? true
aux es igual a x? false
aux es igual a y? true
```

# Uso de una variable en Java:

## Comparación (del valor) de 2 variables (I)



BlueJ: ejemplos - Tema3

Copia las siguientes instrucciones en el **CodePad** de Bluej y observa el resultado de su ejecución en su terminal... **¿Está claro para qué sirven los operadores >, <, >= y <=?**

```
int x = 5, y = 9;
boolean esMayor = (x > y); // op. relacional "mayor que"
System.out.println("x es mayor que y? " + esMayor);
boolean esMenor = (x < y); // op. relacional "menor que"
System.out.println("x es menor que y? " + esMenor);
boolean esMayorOIgual = (x >= y); // op. "mayor o igual que"
System.out.println("x es mayor o igual que y? " + esMayorOIgual);
boolean esMenorOIgual = (x <= y); // op. "mayor o igual que"
System.out.println("x es menor o igual que y? " + esMenorOIgual);
```

```
x es mayor que y? false
x es menor que y? true
x es mayor o igual que y? false
x es menor o igual que y? true
```

# Uso de una variable en Java:

## Comparación (del valor) de 2 variables (II)



BlueJ: ejemplos - Tema3

Copia las siguientes instrucciones en el **CodePad** de Bluej y observa el resultado de su ejecución en su terminal... **¿Está claro para qué sirven los operadores `>=` y `<=`?**

```
int x = 5, y = 5;
boolean esMayorOIgual = (x >= y); // op. "mayor o igual que"
System.out.println("x es mayor o igual que y? " + esMayorOIgual);
boolean esMenorOIgual = (x <= y); // op. "mayor o igual que"
System.out.println("x es menor o igual que y? " + esMenorOIgual);
```

```
x es mayor o igual que y? true
x es menor o igual que y? true
```

# Uso de una variable en Java:

## Comparación (del valor) de 2 variables (III)



Bluej: ejemplos - Tema3

Copia lo siguiente en el **CodePad** de Bluej y observa el resultado de su ejecución en su terminal... **¿Algún problema?**

```
PuntoR x = new PuntoR(), y = new PuntoR(4, 4);  
x > y
```



Bluej: ejemplos - Tema3

Añade lo siguiente al **CodePad** de Bluej y observa el resultado de su ejecución en su terminal... **¿Está claro lo que pasa?**

```
PuntoR z = new PuntoR();  
x >= y
```



# ¿Y qué pasa con los objetos?

- **Inicialización:** vía operador **new**, al ejecutar el correspondiente **método constructor** de su tipo
- **Modificación de su valor:** vía operador **.**, al ejecutar el correspondiente **método modificador** de su tipo
- **Copia de un objeto:** vía operador **.**, al ejecutar el correspondiente **método “clone”** de su tipo
- **Igualdad de 2 objetos:** vía operador **.**, al ejecutar el correspondiente **método “equals”** de su tipo
- **Comparación de 2 objetos:** vía operador **.**, al ejecutar el correspondiente **método “compareTo”** de su tipo

# ¿Y qué pasa con los objetos?

## Ejemplo 1: Traza de la inicialización de un objeto

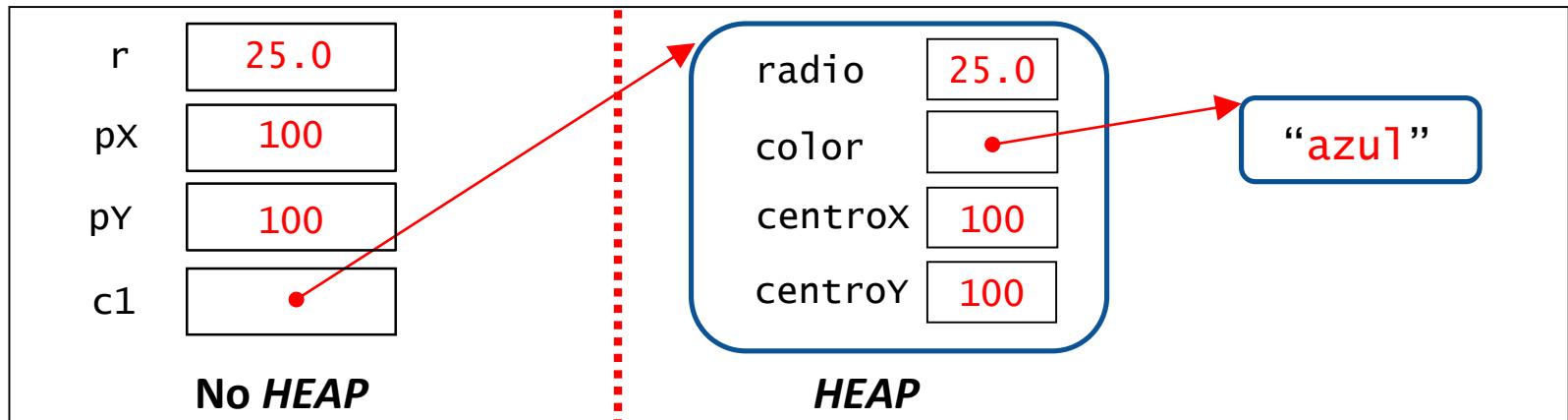


BlueJ: ejemplos - Tema3

Ejecuta las siguientes instrucciones en el **CodePad** de BlueJ y comprueba que el estado de la memoria es el que muestra la figura que aparece detrás de ellas

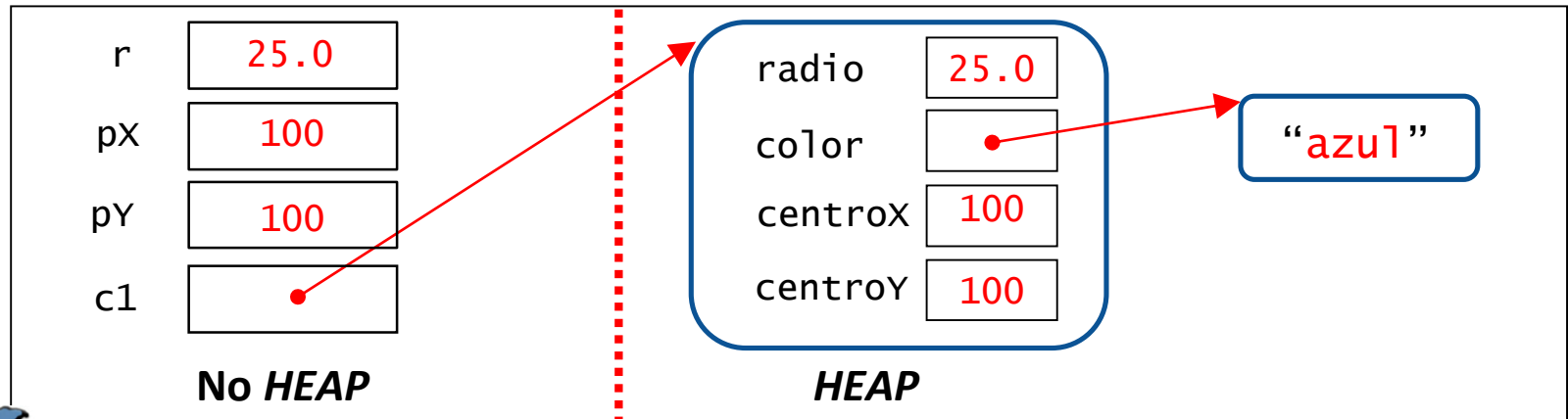
```
double r = 25.0;  
int px = 100, py = 100;  
Circulo c1 = new Circulo(r, "azul", px, py);
```

### Memoria del sistema



# ¿Y qué pasa con los objetos?

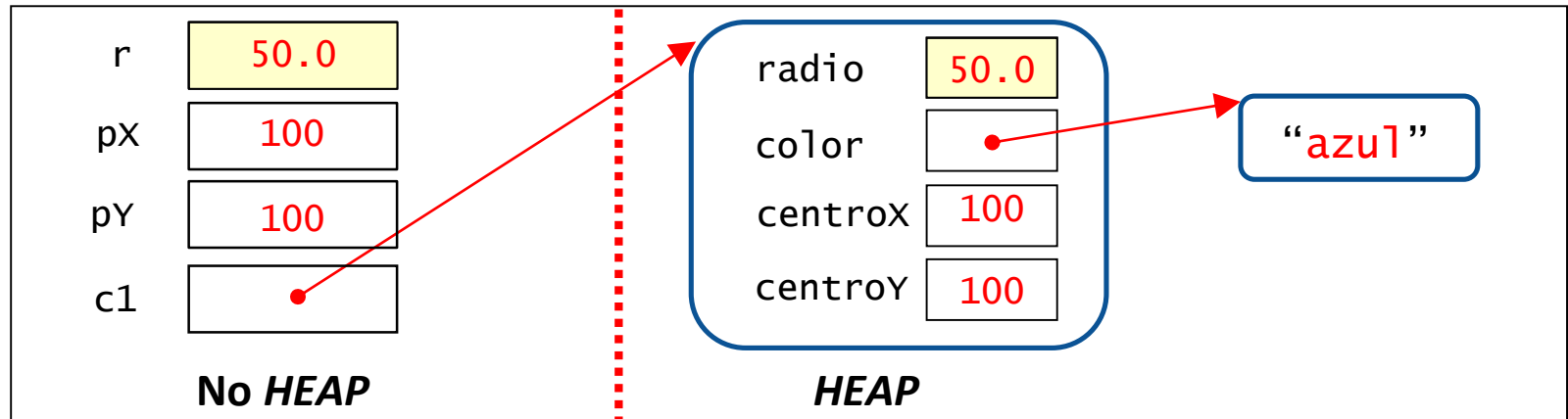
## Ejemplo 2: Taza de la modificación de un objeto



BlueJ: ejemplos - Tema3

Ejecuta las siguientes instrucciones en el **CodePad** de BlueJ y dibuja el estado de la memoria resultante

```
r = 2 * c1.getRadio();  
c1.setRadio(r);
```



# Resumen

- Una variable informática **NO** es una variable matemática
- Un **objeto NO** es la **referencia** que lo apunta/maneja
- La **COMPATIBILIDAD de tipos** es la base de la evaluación de una expresión que se asigna a una variable
- Las **operaciones** que se le aplican a una variable difieren según su **tipo** sea...

➡ **Primitivo: operadores “matemáticos”** (asignación, aritméticos, lógicos y relacionales) y de **casting**

**¡OJO CON EL OPERADOR + que “suma” Strings!**

➡ **Referencia: métodos** que se aplican a las variables referencia usando, los operadores “informáticos” **new** y **.**

Además, en muy determinados casos, se pueden manipular referencias con los operadores asignación (**=**) y los relacionales **==** y **!=**

**NOTA:** también se pueden usar los operadores de **casting** e **instanceof** pero, por el momento, no lo haremos

# Detalles, ejemplos y ejercicios

## Nota

Para que puedas comprobar si has entendido cómo se evalúa una expresión aritmética con operadores simples y unarios (propiedades de los operadores, reglas de precedencia y asociatividad por la izq., cómo evalúa la JVM una asignación, etc.), hemos seleccionado varios ejercicios entre los propuestos en el Capítulo 3 del libro de la asignatura. Intenta resolverlos con la ayuda del material que te proporcionamos



BlueJ: ejercicios – Tema 3

- **Descarga** (desde la carpeta Tema 3 de la PoliformaT) el **proyecto BlueJ ejercicios – Tema 3**
- **Abre el proyecto** y prepárate para ejecutar los programas que contiene y usar el **Code Pad** para resolver dudas o razonar tus respuestas

# Detalles, ejemplos y ejercicios

## Desbordamiento

Realizar operaciones con números puede producir que el resultado exceda la capacidad de representación del tipo. Se habla de **desbordamiento**.

- En la aritmética **entera**, se obtiene un resultado **incorrecto**

byte	$127 + 1 = -128$
short	$32767 + 1 = -32768$
int	$2147483647 + 1 = -2147483648$
long	$9223372036854775807 + 1 = -9223372036854775808$

- En la aritmética **real**, se obtiene un resultado **Infinity** o **-Infinity**

float	$1e38f * 10$	Infinity
double	$1e308 * 10$	Infinity

Los infinitos se propagan en la evaluación de expresiones

$(5.0 / 0.0) + 166.386$	Infinity
-------------------------	----------

# Detalles, ejemplos y ejercicios

## Compatibilidad (I) - Conversión de tipo automática

byte → short → int → long → float → double



BlueJ: ejercicios - Tema 3

- Copia las siguientes instrucciones en el **Code Pad** de BlueJ y observa el resultado de su ejecución... **¿Algún problema?** En caso afirmativo, indica su origen

```
int j = 55, k;
```

```
long x, y, z;
```

```
x = 98; OK
```

```
y = j; OK
```

```
z = 9 * j; OK
```

- Añade la siguiente instrucción al **Code Pad** de BlueJ y observa el resultado de su ejecución... **¿Algún problema?** En caso afirmativo, indica su origen

```
k = 55L; No se puede asignar un long(55L) a un int(k): NO "cabe"
```

- Añade la siguiente instrucción al **Code Pad** de BlueJ y observa el resultado de su ejecución... **¿Algún problema?** En caso afirmativo, indica su origen

```
j = y;
```

## Compatibilidad (II) - Conversión forzada de tipo o *casting*

(**tipo**) expresión



Bluej: ejercicios - Tema 3

Para arreglar los problemas de compatibilidad se puede hacer que un `Long` “quepa a la fuerza” en un `int`: basta hacer un *casting* a `int` del `Long`

**Copia** las siguientes instrucciones **en el Code Pad** de Bluej para ver la sintaxis Java de esta conversión “forzada” y comprobar que funciona

```
int j = 55, k;  
long y;  
k = (int) 55L;  
j = (int) y;
```

OK: compatibilidad **forzada**

OK: compatibilidad **forzada**

Añade las siguientes instrucciones al **CodePad** de Bluej y observa el resultado de su ejecución... **¿Algún problema de compatibilidad? ¿Se puede arreglar con un *casting*?**

```
double d = 123.67;  
int dTruncado = d;
```

No se puede asignar un `double` a un `int` más que forzando la compatibilidad vía *casting*

```
int dTruncado = (int) d;  
dTruncado  
123  
d  
123.67 (double)
```



# Detalles, ejemplos y ejercicios: División (I)

## Operadores aritméticos involucrados

- El resultado de dividir (operador `/`) dos enteros es el **cociente de la división** (entera); si lo que se quiere obtener es el **resto de la división** hay que aplicar el operador `%`

```
> 10 / 3
3 (int)
> 10 % 3
1 (int)
> 10 / 3.0
3.333333333333335 (double)
> 10.0 / 3.0
3.333333333333335 (double)
> |
```

PERO si -al menos- uno de los operandos es un nº real el **resultado de la división (operador `/`) es también un nº real real**

- La división de un nº entero por cero produce una **Excepción** (error de ejecución)

```
> 100 / 0
Exception: java.lang.ArithmeticException (/ by zero)
```

- La división de un **número real** por cero **NO** produce una Excepción sino el resultado **Infinity**, **-Infinity** o **NaN** (error lógico)

```
> 5.0 / 0.0
Infinity (double)
> -5.0 / 0.0
-Infinity (double)
> 0.0 / 0.0
NaN (double)
> |
```

# Detalles, ejemplos y ejercicios: División (II)

## Uso del *casting* para forzar una división real



BlueJ: ejercicios – Tema 3

Copia lo siguiente en el **CodePad** de BlueJ...

**¿Qué valor de `res` se mostrará? ¿Por qué?**

```
int dividendo = 6, divisor = 10;  
double res = dividendo / divisor;  
res  
0.0 (double)
```

**¿Qué única instrucción de las anteriores modificarías para que el valor de `res` fuese `0.6`? ¿Cómo la modificarías?**

**PISTA:** **NO** puedes cambiar el tipo de ninguna variable **pero Sí** usar un (único) *casting* a `double`

```
int dividendo = 6, divisor = 10;  
double res = (double) dividendo / divisor;  
res  
0.6 (double)
```

- Si escribes “`(double) (dividendo / divisor)`” el valor de `res` es, de nuevo, `0.0`... **¿Por qué?**
- Si escribes “`((double) dividendo) / divisor`” el valor de `res` es `0.6`... Tras consultar la tabla de precedencia de operadores (ver traspa 57), **indica por qué SOBРАН los paréntesis**

# Detalles, ejemplos y ejercicios: División (III)

## Combinando el uso de los operadores / y % para resolver problemas



BlueJ: ejercicios - Tema 3

El siguiente código calcula cuántos días “completos” hay en una cantidad de segundos dada y cuántos segundos restan tras el cálculo, i.e. no “alcanzan” para formar un día

```
long segundos = 765432;  
long dias = segundos / (24 * 60 * 60); // número de días  
segundos = segundos % (24 * 60 * 60); // segundos que restan
```

Para obtener este código se emplean una regla de tres básica y las propiedades de la


**división entera:**  $24 * 60 * 60$  segundos  $\rightarrow$  1 día  
 $765432$  segundos  $\rightarrow$   $x$  días }  $x = 765432 / (24 * 60 * 60)$

**OJO:**  $x$  es, obviamente, el COCIENTE de la división planteada; así que, los segundos sobrantes son el RESTO de esa división. En Java se pueden obtener estos valores aplicando, respectivamente, los operadores / y %

- **Escribe en el Code Pad** de BlueJ las instrucciones necesarias para calcular cuántas horas y cuántos minutos hay en una cantidad dada de segundos y cuántos segundos restan de la cantidad inicial tras estos cálculos

# Detalles, ejemplos y ejercicios: Precedencia de operadores (I)

## Tabla de Precedencias



Grupo	Clasificación	Operadores
0	Paréntesis	( )
1	Operadores unarios posfijos	(parámetros) expr++ expr--
2	Operadores unarios prefijos	++expr --expr +expr -expr !
3	Creación y <i>casting</i>	new (tipo) expr
4	Multiplicación	* / %
5	Suma	+ -
6	Relacionales	> >= < <=
7	Igualdad	== !=
8	Conjunción lógica	&
9	Disyunción exclusiva	^
10	Disyunción lógica	
11	Conjunción cortocircuitada	&&
12	Disyunción cortocircuitada	
13	Operador ternario	? :
14	<i>Asignación</i>	= += -= *= /= %=

# Detalles, ejemplos y ejercicios: Precedencia de operadores (II)

## Precedencia, Asociatividad por la izquierda y paréntesis

- La precedencia de los distintos operadores (ver tabla anterior) que aparecen en una expresión es la que determina el orden en el que aplicará cada uno de ellos a la hora de evaluar la expresión. **Por ejemplo, la expresión  $x + y * z$  se evalúa como  $x + (y * z)$  porque el operador  $*$  tiene una prioridad más alta que el  $+$**
- Si los operadores que aparecen en una expresión tienen la misma prioridad se evalúan de izquierda a derecha (**asociatividad por la izquierda**). **Por ejemplo, la expresión  $x + y + z$  se evalúa como  $(x + y) + z$**
- La precedencia y la asociatividad se pueden alterar con el uso de paréntesis



BlueJ: ejemplos - Tema3

**¡Úsalos solo cuando resulte imprescindible!**

Copia cada una de las siguientes expresiones en el **Code Pad** de BlueJ, observa el resultado de su evaluación y **explícalo** en términos de la **precedencia de los operadores que aparecen en ella**

$2 + 3 + \text{"test"}$

$\text{"test"} + 2 + 3$

$\text{"test"} + 2 * 3$

# Detalles, ejemplos y ejercicios: Precedencia de operadores (III)

## Operadores Aritméticos Simples. Ejercicio 1 - Capítulo 3

```
public class Prueba {  
    public static void main(String[] args) {  
        double x, y;  
        x = 5.0;  
        y = 7 / 9 * (x + 1);  
        System.out.println("x = " + x + ", y = " + y);  
    }  
}
```



BlueJ: ejercicios - Tema 3

- **Comprueba** que al ejecutar el programa **Prueba** del proyecto el resultado que aparece en la primera línea del terminal de BlueJ es:  $x = 5.0$ ,  $y = 0.0$
- **Ejecuta tú en un papel**, a mano, este programa... **¿Qué valores de x e y obtienes?**

Si son diferentes de los que salen en el terminal de BlueJ... Lo siento, pero algo estás haciendo mal. Si son los mismos, explica por qué motivo **y** vale **0**

**Recuerda:** evaluar alguna subexpresión en el *Code Pad* de BlueJ te puede ayudar tanto a resolver dudas como a razonar tu respuesta

# Detalles, ejemplos y ejercicios: Precedencia de operadores (IV)

## Operadores Aritméticos Simples. Ejercicio 3 - Capítulo 3

123456 / 10 se evalúa a 12345 y 123456 % 10 se evalúa a 6  
123456 / 100 se evalúa a 1234 y 123456 % 100 se evalúa a 56  
123456 / 1000 se evalúa a 123 y 123456 % 1000 se evalúa a 456  
123456 / 10000 se evalúa a 12 y 123456 % 10000 se evalúa a 3456  
123456 / 100000 se evalúa a 1 y 123456 % 100000 se evalúa a 23456



BlueJ: ejercicios - Tema 3

- **Comprueba** que al ejecutar el programa **Ejercicio3C3** del proyecto el resultado que aparece en el terminal de BlueJ es el del cuadro superior.
- **Evalúa tú en un papel**, a mano, las expresiones del programa... **¿Obtienes los mismos resultados?**

Si son diferentes de los que salen en el terminal de BlueJ... Lo siento, pero algo estás haciendo mal. Si son los mismos, responde...

1. ¿Para qué sirve obtener el cociente (operador /) y el resto (operador %) de la división entera de un número entre 10, entre 100, ...?
2. Si tienes un nº entero, p. ej. **123456**, ¿cómo obtener el segundo de sus dígitos, i.e. su nº de decenas (**5** para el ejemplo)? ¿Y el tercero, i.e. su nº de centenas (**4** para el ejemplo)? ¿Y el primero, i.e. el nº de unidades (**6** para el ejemplo)?

# Detalles, ejemplos y ejercicios: Precedencia de operadores (V)

## Operadores Aritméticos Simples.- Ejercicio 4 - Capítulo 3

$3 / 4 * (a * a - b)$  se evalúa a 0, cuando el resultado que se quiere es 16.5

$a / b * 1000 + 304$  se evalúa a 1304, cuando el resultado que se quiere es 1970.6666666666667

$(100 / a + b / 2) * 5$  se evalúa a 105, cuando el resultado que se quiere es 107.5



BlueJ: ejercicios - Tema 3

- **Comprueba** que al ejecutar el programa **Ejercicio4C3** del proyecto el resultado que aparece en el terminal de BlueJ es el del cuadro superior.
- **Evalúa tú en un papel**, a mano, las expresiones del programa... **¿Obtienes los mismos resultados?**

Si son diferentes de los que salen en el terminal de BlueJ... Lo siento, pero algo estás haciendo mal. Si son los mismos, **¿qué modificarías en cada expresión para que se evalúe al resultado correcto que indica el ejercicio?**



# Detalles, ejemplos y ejercicios

## Operadores Aritméticos Unarios. **Ejercicio 9 - Capítulo 3**

12 8 6

6 8 6

6 8 14

22 8 14

23 9 14

24 10 33

¿Por qué se obtiene este resultado?



BlueJ: ejercicios - Tema 3

- **Comprueba** que al ejecutar el programa **TestOperador** del proyecto el resultado que aparece en el terminal de BlueJ es el del cuadro superior
- **Evalúa tú en un papel**, a mano, las expresiones del programa... **¿Obtienes los mismos resultados?**

Si son diferentes de los que salen en el terminal de BlueJ... Lo siento, pero algo estás haciendo mal. Si son los mismos, **explica por qué motivo**

# Detalles, ejemplos y ejercicios

## Operaciones con variables de tipo char

```
char ch1 = 'A';  
char letraB = (char) ((int) ch1 + 1);  
System.out.println("La letra que sigue a la A es " + letraB);  
char letraC = 'B' + 1;  
System.out.println(((int) letraC) + " es el código de " + letraC);  
char letraN = '\u006E'; //código Unicode hexadecimal de la n  
letraN += 'A' - 'a';  
System.out.print("Letras " + '\u006E' + " y " + letraN);
```



BlueJ: ejercicios - Tema 3

- **Ejecuta** las anteriores instrucciones en el *Code Pad*
- **Analiza** el resultado que aparece en el terminal de BlueJ. Al hacerlo...
  - **Ten en cuenta que** un literal de tipo carácter se representa internamente como un valor entero positivo (pero sin la representación en complemento a 2)
  - **Consulta la tabla de codificación ASCII** (ver transparencia nº 10 del *pdf* de la primera sesión del tema), para comprobar el código numérico de cada char

# Detalles, ejemplos y ejercicios

## Operadores relacionales

```
int x = 5;  
boolean b1 = 6 == x,  
        b2 = x <= 7,  
        b3 = (4 + x) > 10,  
        b4 = 'a' < 'b',  
        b5 = true == false;  
b2 = b3 = 5.5 != 6.3;
```



BlueJ: ejercicios - Tema 3

- **Ejecuta** las anteriores instrucciones en el **Code Pad** de BlueJ e **indica** el valor que toman b1, b2, b3, b4 y b5.
- **Explica** qué sucede si ahora ejecutas las siguientes instrucciones:

```
b5 = true >= false;  
String s1 = new String("5"), s2 = new String("7");  
boolean b6 = s1 <= s2;
```

**Recuerda:** cuando los operandos son, bien de tipo `boolean`, bien variables Referencia, SOLO se pueden emplear los operadores relacionales `==` y `!=`

# Detalles, ejemplos y ejercicios

## Operadores lógicos cortocircuitados



BlueJ: ejercicios - Tema 3

**Responde:** para que cada una de las siguientes expresiones se evalúe a true, ¿cuáles son los valores que puede tomar la variable x?

Al hacerlo **recuerda que** si en una expresión aparecen operadores **cortocircuitados** NO se continúa con su evaluación si se obtiene el resultado antes de evaluar toda la expresión

```
x >= 15 && x <= 20
```

```
x < 15 || x > 20
```

```
!(x >= 15 && x <= 20) // ! es el operador negación (not)
```

```
x > 15 || x == 15
```

```
(x >= 0 && x < 5 || x >= 10 && x <= 20) && x % 2 != 1
```