

# UT2 Objetos y clases

Módulo – Programación (1º)

Ciclos – Desarrollo de Aplicaciones Multiplataforma | Desarrollo de Aplicaciones Web

Cl María Ana Sanz

---

# Contenidos

---

- Objetos y clases
- Métodos
- Parámetros
- Tipos de datos
- Estado de un objeto
- Interacción de objetos
- Código fuente
- Valor de retorno de un método
- Objetos como parámetros
- Tipos de datos
  - tipos primitivos
  - tipos referencia
- Expresiones en Java

# Objetivos

---

- Diferenciar objeto y clase
- Aprender a trabajar con el entorno BlueJ
- Comprender el concepto de tipo de datos
- Saber construir y evaluar expresiones en Java

# Objetos y clases

- Leer punto 2.1
- **Clase**
  - abstracción que describe a un tipo particular de objetos
  - Plantilla para crear objetos
  - describe un tipo de objeto
  - Persona / Estudiante / Profesor
  - Coche
  - Figura / Círculo

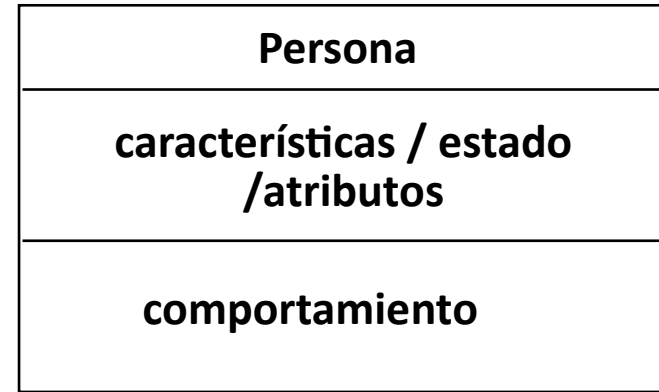
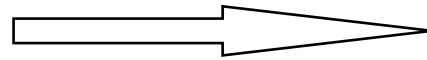
# Objetos y clases

## ■ Objeto

- instancia individual de una clase
- se crean a partir de las clases
- representación detallada y particular de algo de la realidad
- un objeto tiene estado, comportamiento e identidad
- la estructura y el comportamiento de objetos similares están definidas en su clase común
- *miCoche, tuCoche* – objetos de la clase Coche
- *juan, pedro* – objetos de la clase Persona

# Objetos y clases

**Clase**



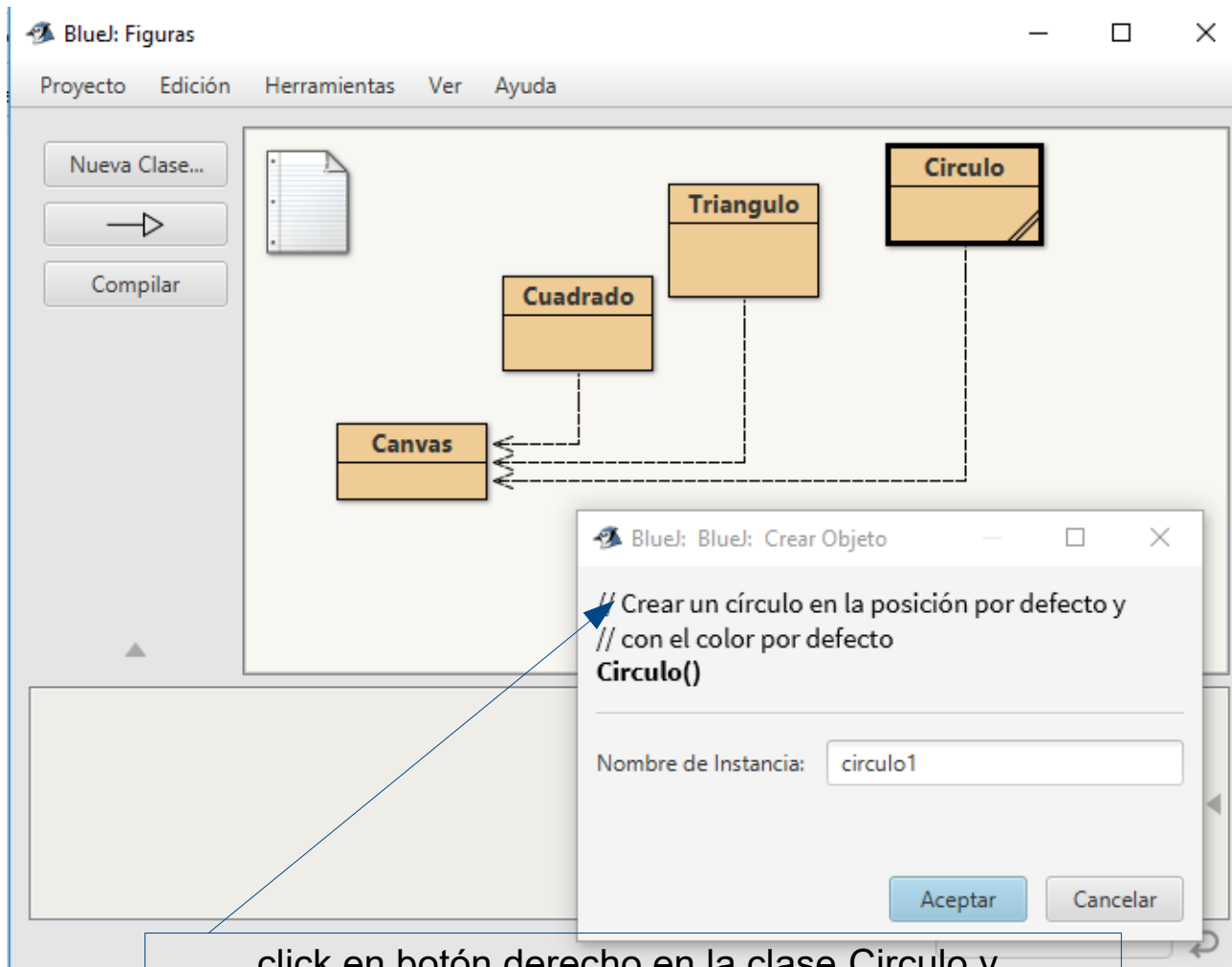
**Objetos**



# ¿De dónde extraemos los objetos?

- Analizando el problema
  - los objetos aparecen en el dominio del problema
  - sustantivos, nombres
- Representando el dominio del problema (a través de UML y un diagrama de clases)
- *“Un artista quiere utilizar un programa para hacer dibujos geométricos, círculos, cuadrados, triángulos. Las figuras deben tener un color, se han de visualizar en la pantalla, mover a los lados, arriba y abajo, cambiar de tamaño, esconder o mostrar, mover a una distancia fija o variable,....”*
  - Objetos – dibujo, círculos, cuadrado,
  - Estado de los objetos – color,
  - Comportamiento – mover, cambiar tamaño, ....

# Creando objetos - Proyecto Figuras



click en botón derecho en la clase **Circulo** y elegir **new Circulo()**  
introducir el nombre de la instancia

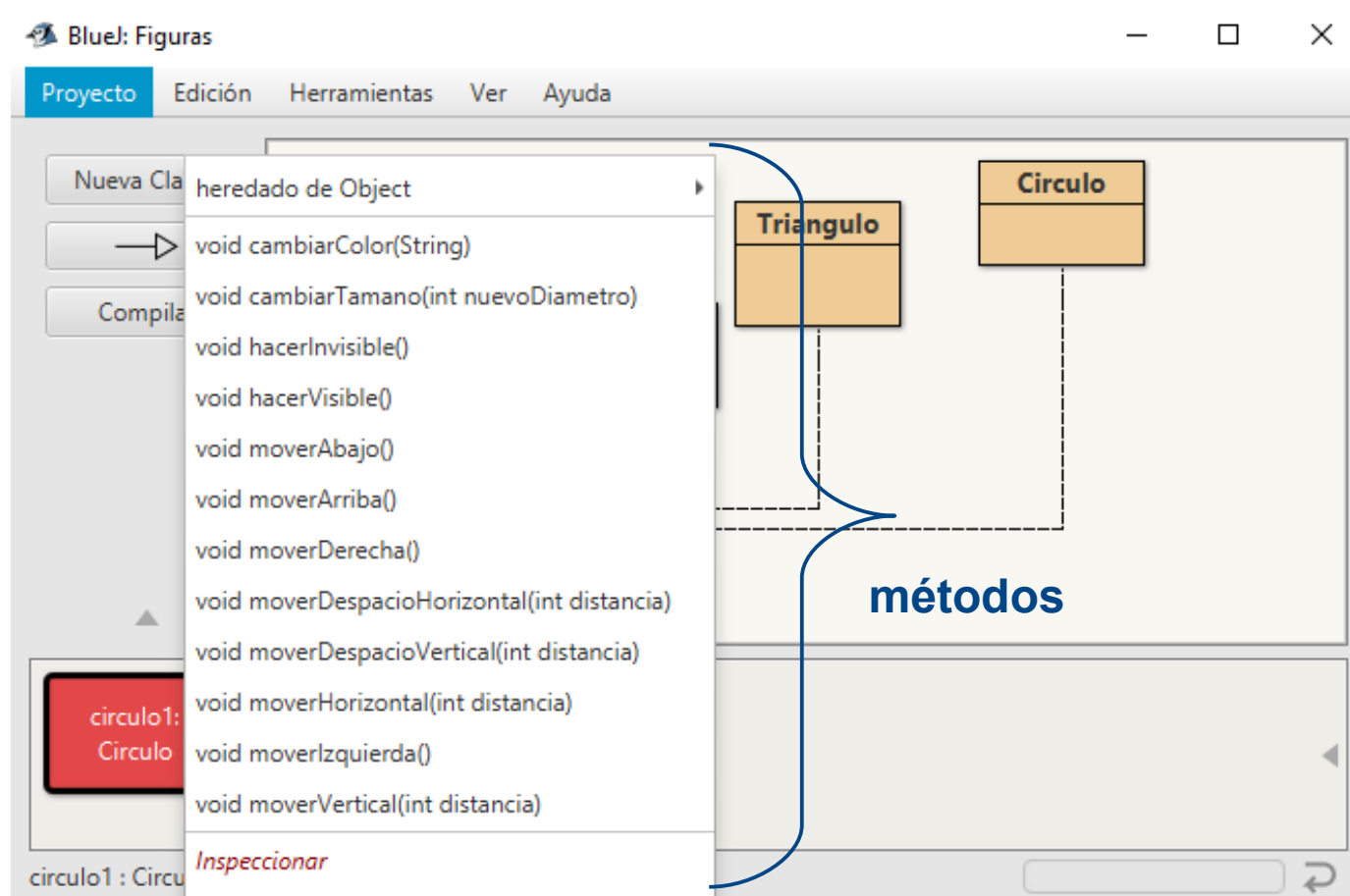
- 4 clases
  - Canvas, Cuadrado, Círculo, Triángulo
- Diagrama de clases en BlueJ (relaciones entre clases) muy simplificado
- Nombres de clases en mayúsculas y singular
- nombres de objetos en minúscula
- Ejer 2.1



# Llamando a métodos

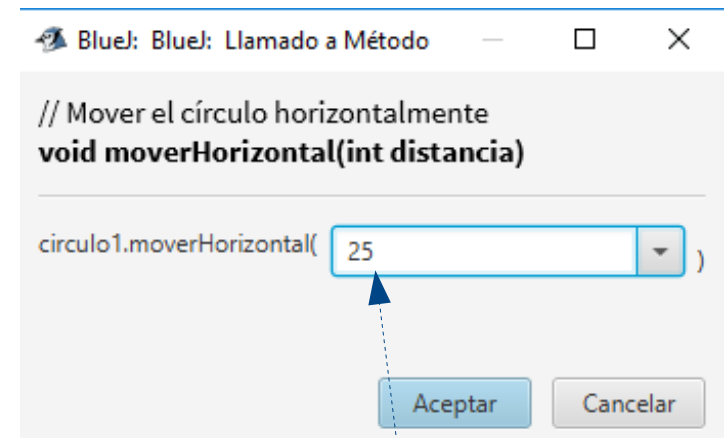
- A través de los métodos nos comunicamos con los objetos
- Los objetos hacen algo cuando invocamos a sus métodos
- Los métodos definen el comportamiento de un objeto
- Cuando invocamos un método sobre un objeto estamos enviando un mensaje al objeto.
- POO - conjunto de objetos que interactúan entre sí a través de mensajes.
- Algunos mensajes se los envía un objeto a sí mismo.
- Otros mensajes los envía un objeto a otro objeto
  - `circulo1.moverAbajo()`
- Ejer 2.2.

# Llamando a métodos



# Parámetros

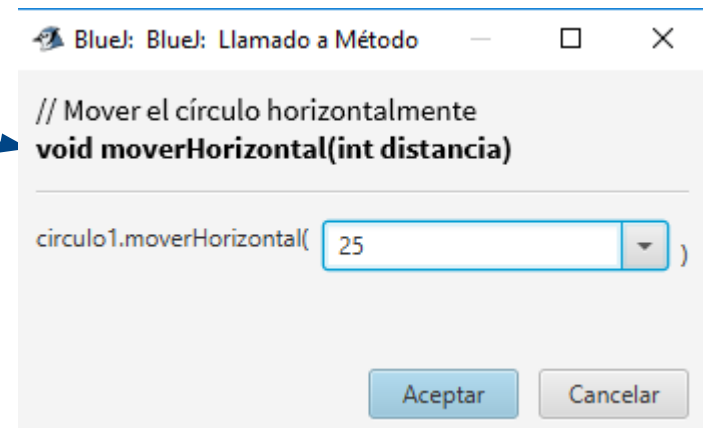
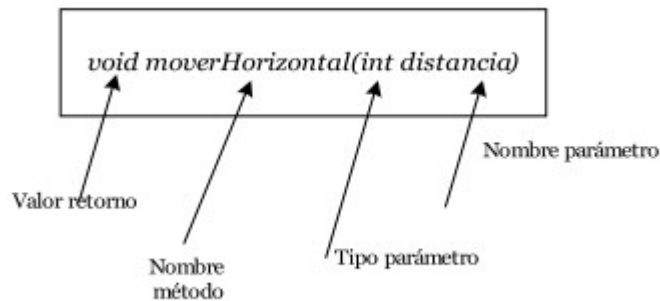
- Los métodos pueden tener parámetros
- Los parámetros proporcionan información adicional a los métodos
- Ejer 2.3
  - void moverHorizontal(int distancia)
  - Llama a moverVertical().
  - Invoca moverDespacioVertical().



parámetro

# Parámetros

- Cuando un método necesita un parámetro indica qué tipo de parámetro requiere
  - **void moverHorizontal(int distancia)**
- Esta línea es la **signatura** del método
  - información sobre el método (nombre y valor de retorno)
  - parámetros que necesita el método (tipo y nombre de cada parámetro)



- Método sin parámetros - nombre del método y paréntesis vacíos
  - **void moverAbajo()**

# Ejemplos signatures de métodos

- `void cambiarTamano(int nuevoAlto, int nuevoAncho)`
- `void hacerVisible()`
- `void enviarFlores(int cantidad, String tipoFlor, String aQuien)`
  
- `int`, `void`, `class` – palabras reservadas del lenguaje Java
- `moverHorizontal()`, `distancia`, `tipoFlor`, ...
  - nombres descriptivos (legibilidad)
  - nombres de clases en mayúsculas
  - notación *camelCase* para métodos y resto de variables
    - `tipoFlor`, `distancia`, `esVisible`

# Reglas de estilo del lenguaje Java

---

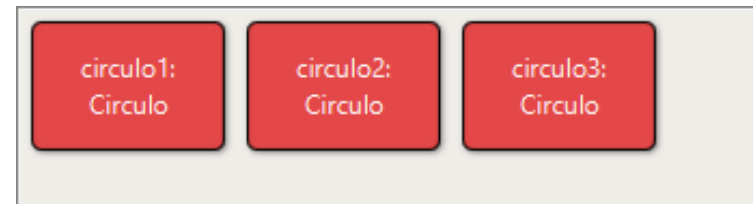
- Leer el código de la clase Circulo
- Código legible. Respetar reglas de estilo
  - escritura clara
  - nombres descriptivos (notación camelCase)
  - indentación correcta
  - { } en líneas separadas
  - uso de comentarios
  - blancos de separación

# Tipos de datos

- Los parámetros tienen tipos.
- Un **tipo de datos** indica qué valores puede tomar el parámetro.
  - `void cambiarColor(String color)`
  - `void enviarFlores(int cantidad, String tipoFlor, String aQuien)`
- Especificando el tipo el compilador sabe cuánta memoria reservar y con qué valores va a trabajar
- No solo los parámetros tienen tipo, también
  - atributos
  - variables locales y
  - valores de retorno de los métodos

# Múltiples instancias

- Dada una clase podemos crear varios objetos (varias instancias) a partir de ella.
- Ejer 2.5.



Múltiples instancias de la clase Circulo,  
Cada uno de ellos tendrá su propio color,  
tamaño, posición.  
Todos se comportarán de la misma manera

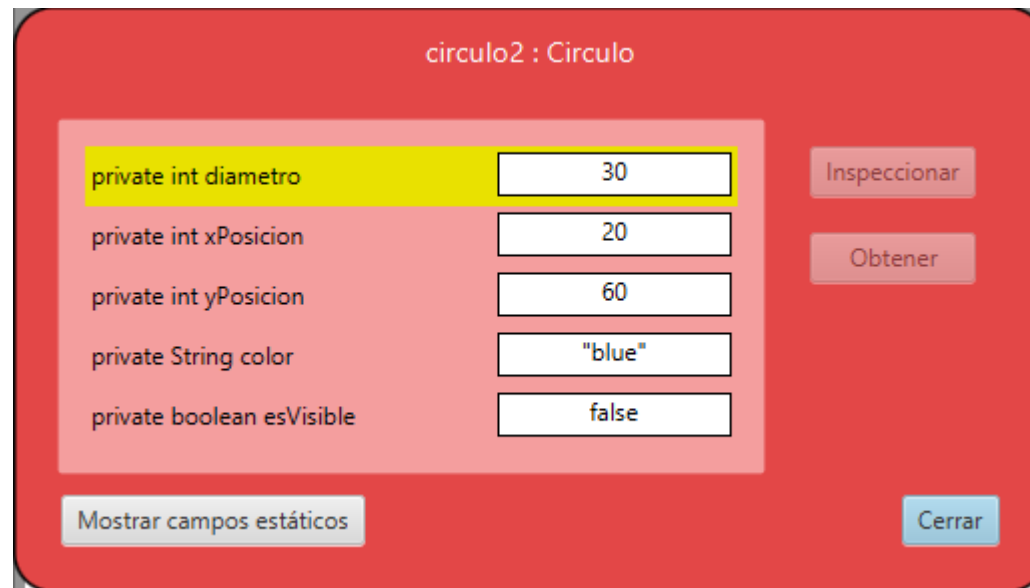


# Estado de un objeto

- Todo objeto tiene un estado.
- El estado de un objeto está definido por los valores de sus **atributos** (campos).
  - *diámetro, xPosicion, yPosicion, color, esVisible*
- Los atributos también tienen un tipo
- Algunos métodos
  - modifican el estado de un objeto (mutadores)
    - void moverIzquierda() - modifica el atributo *xPosicion*
  - otros consultan su estado (accesores)
    - boolean esVisible() - consulta el estado del atributo *esVisible*

# Estado de un objeto - Inspeccionar de BlueJ

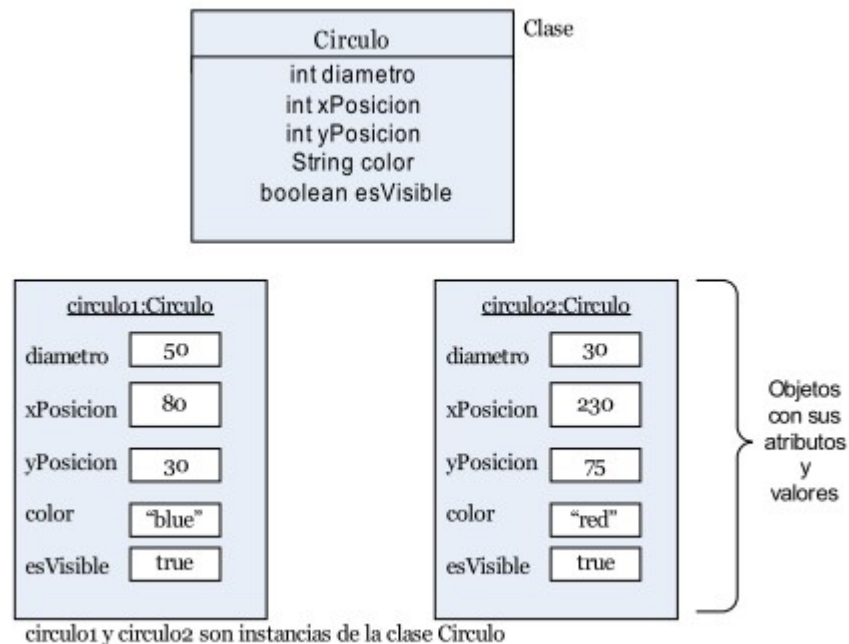
- En BlueJ podemos ver el estado de un objeto con la función *Inspeccionar*.



- Ejer 2.6

# Estado de un objeto

- Objetos de la misma clase tienen todos los mismos atributos.
  - El nombre, tipo y nº de los campos es el mismo mientras que el valor actual de cada atributo en cada objeto puede variar.
- Objetos de diferentes clases pueden tener diferentes atributos.



# Estado de un objeto / Comportamiento de un objeto

- Cuando se crea un objeto de una clase automáticamente tiene los atributos definidos en esa clase.
  - Los valores de esos atributos se almacenan en el objeto.
- Los métodos se definen también en la clase.
  - Todos los objetos de una misma clase tienen los mismos métodos (el mismo comportamiento).
- Ejer 2.7.

**Los métodos se invocan sobre los objetos,  
no sobre las clases.**

# Interacción de objetos

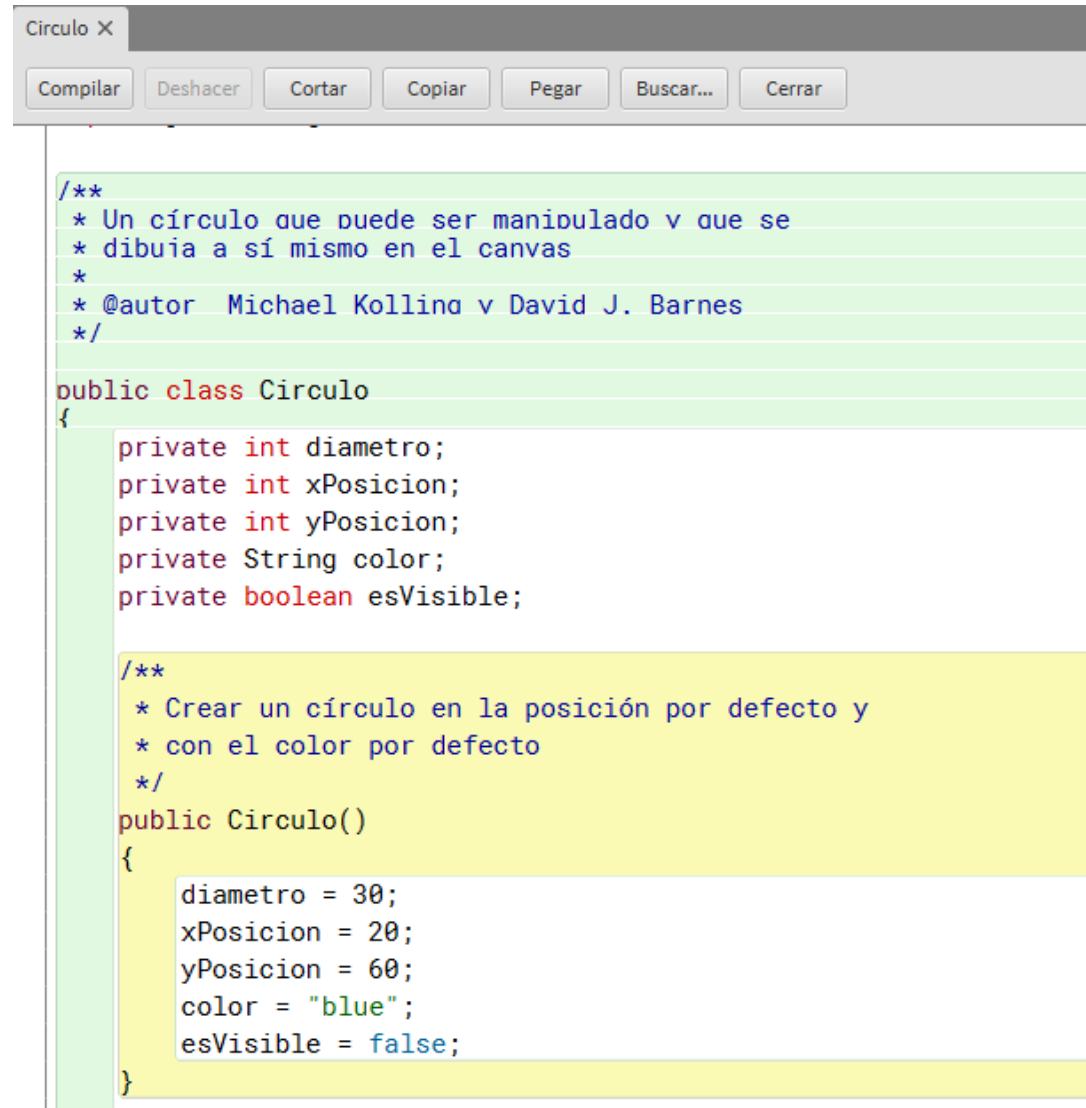
- Los objetos pueden crear otros objetos
- Los objetos interactúan entre ellos a través de llamadas a métodos – ***paso de mensajes***
- Un programa es un conjunto de objetos
  - El usuario de un programa inicia el programa (normalmente creando un primer objeto) y todos los demás objetos se crean (directa o indirectamente) a partir de éste.
- Ejer 2.8.
  - El objeto *dibujo* de la clase Dibujo realiza una tarea (el dibujo) pasando mensajes a los objetos que van a intervenir (un círculo, un triángulo, ...)

# Código fuente

- Cada clase tiene un código fuente asociado escrito en el lenguaje Java. A través de ese código se describe la estructura de la clase:
  - atributos (estado) de los objetos
  - métodos (comportamiento)
    - mensajes que se podrán enviar a esos objetos
- En BlueJ, cómo veo el código fuente? *Abrir Editor* desde el menú contextual o haciendo doble click sobre la clase.

# Código fuente

- Una vez escrito el código fuente de una clase se compila (botón *Compilar*). Si hacemos cambios al código fuente habremos de compilarla otra vez.
- Ejer 2.9. Ejer 2.10



```
/**
 * Un círculo que puede ser manipulado y que se
 * dibuja a sí mismo en el canvas
 *
 * @autor Michael Kollina y David J. Barnes
 */

public class Circulo
{
    private int diametro;
    private int xPosicion;
    private int yPosicion;
    private String color;
    private boolean esVisible;

    /**
     * Crear un círculo en la posición por defecto y
     * con el color por defecto
     */
    public Circulo()
    {
        diametro = 30;
        xPosicion = 20;
        yPosicion = 60;
        color = "blue";
        esVisible = false;
    }
}
```

# Valores de retorno

- Un método puede devolver un valor (valor de retorno), un resultado.

```
public String getNombre()
```

tipo del valor  
de retorno, en  
este caso  
String

```
public void cambiarNombre(String nuevoNombre)
```

no devuelve  
nada

- `public void moverHorizontal(int distancia)`

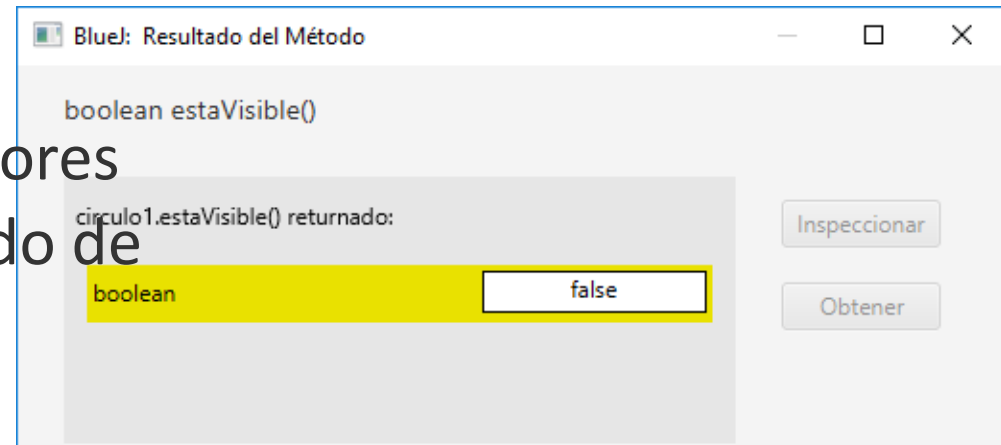
`public String getNombre()`

`public boolean estaVisible()`

- Los métodos que devuelven valores nos permiten consultar el estado de un objeto.

- Ejer 2.11

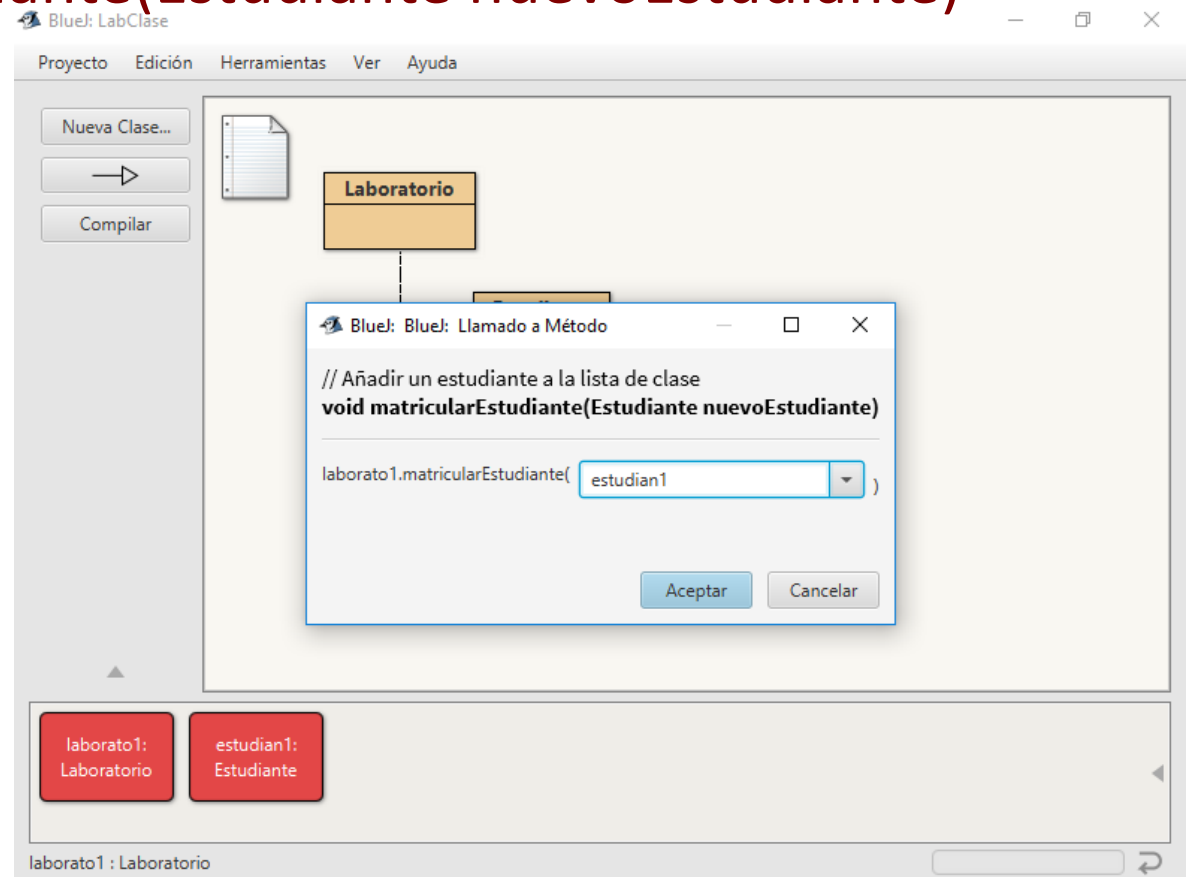
no devuelve nada





# Objetos como parámetros

- Los objetos pueden ser pasados como parámetros a métodos de otros objetos.
  - void matricularEstudiante(Estudiante nuevoEstudiante)**



# Objetos como parámetros

- Ejer 2.12. - Ejer 2.13 – Ejer 2.14 – Ejer 2.16
- Ejer 2.14 -
  - “Hola” (String) 101 (int) -4 (int) false (boolean)
  - “33” (String) -4.7 (double)
- Ejer 2.15 -
  - `public int calcularMedia(int a, int b)`
  - `public void setNombre(String nuevoNombre)`
- Ejer 2.16 -
  - un objeto

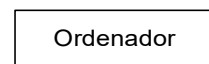


diagrama de  
clase

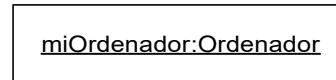


diagrama de  
objetos

# Tipos de datos

- **Tipo de datos**
  - conjunto de valores que una variable (atributo, parámetro, variable local) puede tomar
  - así sabe el compilador la cantidad de memoria que un atributo ocupará y la JVM reservará memoria para él
- Tipos de datos en Java
  - **primitivos** – simples (los que veremos de momento)
  - **referencia** – (apuntan a objetos)

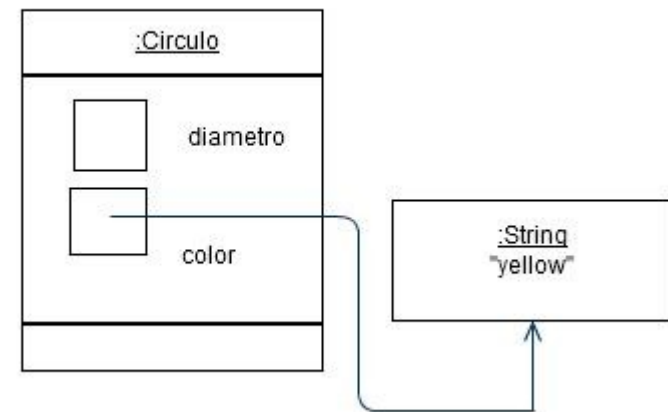
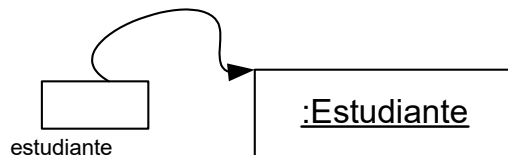
# Tipos de datos

- Tipos de datos en Java

- *primitivos* – El atributo o parámetro (la variable, en general) guarda directamente el valor de ese tipo.

```
private int altura;      178      altura
private boolean esVisible;
                        true      esVisible
```

- *referencia* - almacenan una referencia (puntero) a un objeto.



# Tipos primitivos en Java – Tipos enteros

Nombre del tipo	Precisión	Rango	Ejemplos
Tipos numéricos enteros			
<b>byte</b>	8 bits	-128 a 127	24 -2 123
<b>short</b>	16 bits	-32768 a 32767	1234 -23456
<b>int</b>	32 bits	$-2^{31}$ a $2^{31} - 1$	-2003 5409
<b>long</b>	64 bits	$-2^{63}$ a $2^{63} - 1$	4233781L 55L

byte – 8 bits (un octeto)

01111111 +127

10000000 -128

complemento a 2

$-2^7$  a  $2^7 - 1 = -128$  a  $128-1$  (127)

short – 16 bits (2 octetos)

int – 32 bits (4 octetos)

long – 64 bits (8 octetos)

**Precisión** – el nº de bits que utiliza la máquina virtual de Java para guardar el valor

**Rango** – conjunto de valores que comprende el tipo (valores que se pueden almacenar)

# Tipos primitivos en Java – Tipos reales

Nombre del tipo	Precisión	Rango	Ejemplos
Tipos numéricos reales			
<b>float</b>	32 bits	-3.4E38 a 3.4E38	43.889F
<b>double</b>	64 bits	-1.7E308 a 1.7E308	45.63 2.4e5 45.64

float – 32 bits (4 octetos) – 43.89F

double – 64 bits (8 octetos) – 43.89

$24.5 \text{ e}^{-3} == 24.5 \times 10^{-3} == 24.5 / 10^3$

$43.89 == 4389 \times 10^{-2} == 4389 / 10^2$

notación  
exponencial

# Tipos primitivos en Java – Tipos char y boolean

Nombre del tipo	Precisión	Rango	Ejemplos
<b>Otros tipos</b>			
<b>char</b>	16 bits	Un carácter Unicode (UTF-16)	'm' '?' '\u00F6'
<b>boolean</b>	Java no lo especifica	Valor booleano <i>true</i> o <i>false</i>	true false

- Ejemplos
  - enteros – *edad, nota, numeroPuertas*
  - reales – *peso, media, iva*
  - char – *estadoCivil* ('s' 'c' )
  - boolean – *estaPrestado, estaEncendida*

# Tipos primitivos en Java – Tipos char y boolean

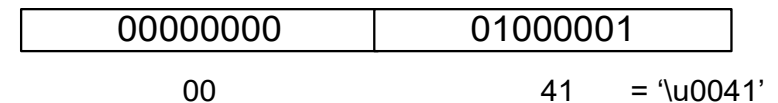
## ■ Ejemplos

- char – un carácter Unicode (16 bits – 4 dígitos hexadecimales)
  - private char letra;

letra = 'A'; letra = '\u0041'; letra = 65;

valor Unicode, en hexadecimal

en decimal



- borrar pantalla en BlueJ - '\u000C'
- secuencias de escape - '\n' '\\ ' '\t' '\"'

Carácter	Significado
\n	Salto de línea
\t	Tabulador
\"	Comillas dobles
'	Comillas simples
\\	Barra invertida



# Operadores y expresiones Java.

- Sobre cada uno de los tipos anteriores se pueden utilizar un conjunto de operadores para formar expresiones
- Una **expresión**
  - se construye agrupando operadores y operandos
  - se evalúa y produce un resultado de un determinado tipo.
- Expresiones
  - **aritméticas** – se construyen con operadores aritméticos
  - **lógicas** - se construyen con operadores lógicos y/o relacionales

# Expresiones aritméticas

- Se construyen con operadores aritméticos
- Los operandos son de un tipo primitivo numérico, entero o real
- Al evaluarlas producen un resultado numérico
- **Operadores aritméticos**                      +   -   \*   /   %
- /   %   resultado depende de los operandos, enteros o reales
- *Reglas de precedencia* de los operadores
  - se aplican cuando hay varios en una expresión
  - ( ) para cambiar la prioridad
  - a igual prioridad de izquierda a derecha
- Construiremos expresiones correctas en cuanto al estilo
  - blancos de separación entre operadores y operandos
  - ( ) para aclarar la expresión

# Precedencia de los operadores

Operadores	
( )	Paréntesis
++ --	Incremento / Decremento
+ - !	Suma / Resta (Unario)
* / %	Producto / División / Resto
+ -	Suma / Resta
< <= > >=	Comparación
== !=	Igual / Distinto
&&	boolean (y / and)
	boolean (or / o)
= += -= *= /= %=	Operadores de asignación

Mayor  
prioridad



Menor  
prioridad

# Expresiones aritméticas. Ejemplos

<b>Ejemplos</b>	$51 * 3 - 53 \Rightarrow 100$ $154 - 2 * 27 \Rightarrow 100$ $(200 - 5) / 2 \Rightarrow 97$ $2 * (47 + 3) \Rightarrow 100$
-----------------	---

<b>Ejemplos</b>	$5 + 3 \Rightarrow 8$ $5 / 3 \Rightarrow 1$ $5.0 / 3 \Rightarrow 1.66666$	$5 / 2 \Rightarrow 2$ $7 \% 3 \Rightarrow 1$
-----------------	---	---

- Ejer 2.17
- Ejer 2.18

## Ejer 2.17

$25 + 20 - 15$	<b>30</b>
$20 * 10 + 15 * 10$	<b>350</b>
$20 * 10 / 2 - 20 + 3 * 3$	<b>89</b>
$15 / 10 * 2 + 3 / 4 * 8$	<b>2</b>
$46 \% 9 + 4 * 4 - 2$	<b>15</b>
$45 + 43 \% 5 * (23 + 3 \% 2)$	<b>117</b>
$1.5 * 3$	<b>4.5</b>

## Ejer 2.18

$2\pi\text{radio}$ ( $\pi = 3.1416$ )	<code>2 * 3.1416 * radio</code> <code>Math.PI</code> (constante de la clase <code>Math</code> )
$2\pi\text{radio}^2$	<code>2 * Math.PI * radio * radio</code>
$a^2 + \frac{b^2}{c}$	<code>a * a + b * b / c</code>
$\frac{4}{3(r+34)} - 9(a+bc) + 3 + \frac{d(2+a)}{a+bd}$	<code>4 / (3 * (r + 34)) - 9 * (a + b * c) + 3 +</code> <code>+ d * (2 + a) / (a + b * d)</code>
$-b + \frac{\sqrt{b^2 - 4ac}}{2a}$	<code>- b + Math.sqrt(b * b - 4 * a * c) / (2 * a)</code>

# Expresiones booleanas

- Se construyen con operadores relacionales y/o lógicos
- Al evaluarlas producen un resultado lógico (booleano), *true* (cierto) o *false* (falso)
- Operadores relacionales usualmente se combinan con operandos y operadores aritméticos

■ **Operadores relacionales**                      ==    <    <=    >    >=    !=

■ **Operadores lógicos**                      &&    ||    !

- se evalúan según las tablas de verdad

# Tablas de verdad

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>	<b>a    b</b>	<b>!a</b>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>
a,b: expresiones booleanas				

- Ejemplos

<b>Expresión</b>	<b>Resultado al evaluarla</b>
4 > 5	false
4 != 5	true
( 4 != 5 ) && ( 4 > 2 )	true
( 4 != 5 ) && ( 4 < 2 )	false

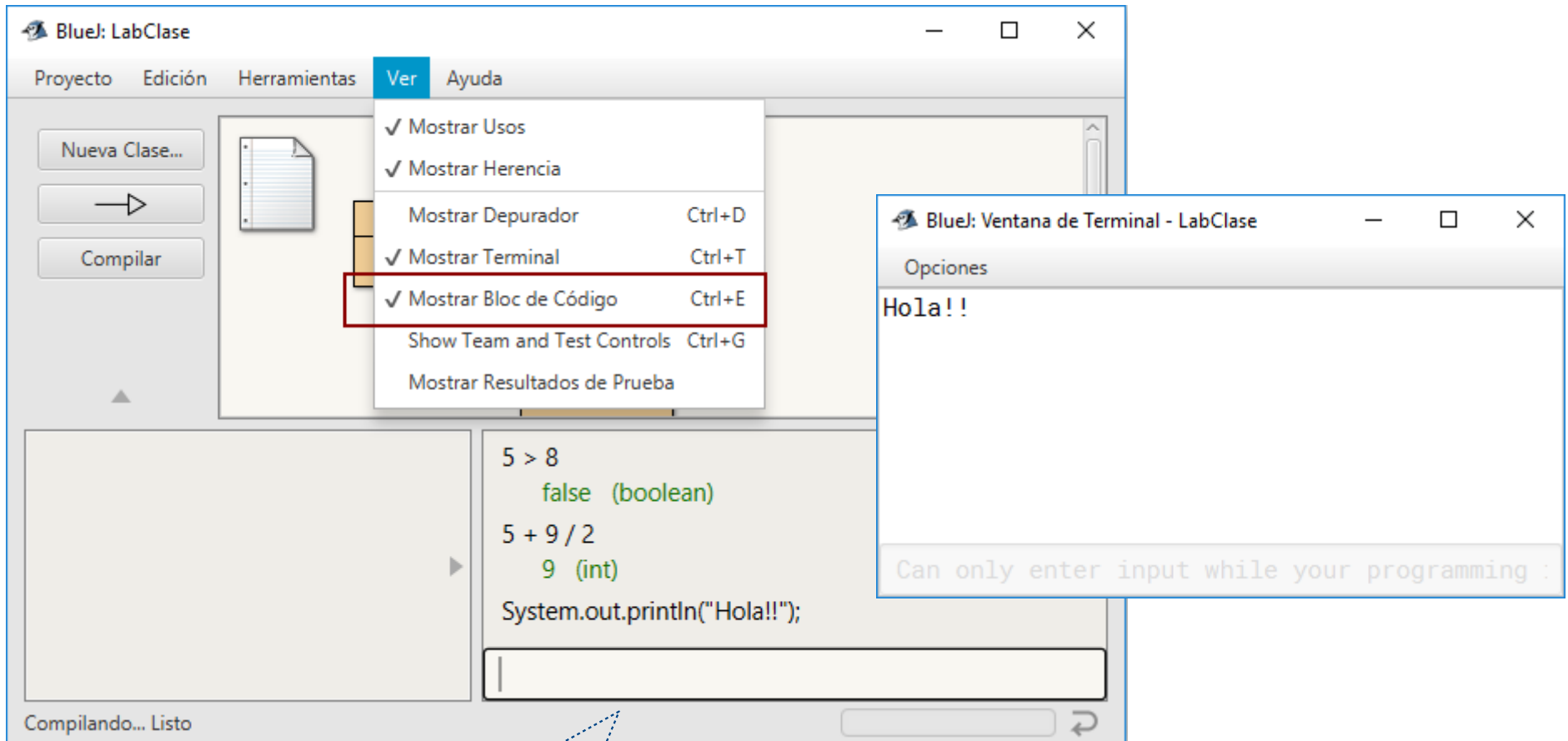


# Expresiones booleanas

- Evaluación en *cortocircuito*
  - Tan pronto como se conoce el resultado final de la expresión no se evalúan el resto de condiciones.

<code>edad &gt;= 65 &amp;&amp; sexo == 'M'</code>	si <i>edad</i> no es mayor o igual a 65 la condición es <i>false</i> y el resultado final de la expresión será también <i>false</i> , no se evalúa la segunda condición
<code><u>notaTeoria</u> &gt;= 5    <u>notaPractica</u> &gt; 7</code>	si <u><i>notaTeoria</i></u> es mayor o igual a 5 la condición es <i>true</i> y el resultado final de la expresión será también <i>true</i> , no se evalúa la segunda condición

# BlueJ y CodePad



El CodePad (Bloque de código) permite introducir expresiones y sentencias para evaluarlas/ejecutarlas de forma inmediata

- Ejer 2.19 - 2.20 - 2.21

## Ejer 2.19 (Sol.)

$(\text{true}) \ \&\& \ (3 > 4)$	<b>false</b>
$(\text{true}) \ \&\& \ (x > 4)$	<b>true</b>
$x \neq 3$	<b>true</b>
$(x > 0) \    \ (x < 0)$	<b>true</b>
$25 > 20 \ \&\& \ 13 > 5$	<b>true</b>
$10 + 4 < 15 - 3 \    \ 2 * 5 + 1 > 14 - 2 * 2$	<b>true</b>
$4 * 2 \leq 8 \    \ 2 * 2 < 5 \ \&\& \ 4 > 3 + 1$	<b>true</b>

## Ejer 2.20 (Sol.)

<b>a</b>	<b>private boolean estaVacía;</b>
<b>b</b>	<b>private int edad;</b>
<b>c</b>	<b>private double facturaLuz;</b>
<b>d</b>	<b>private char estadoCivil;</b>
<b>e</b>	<b>private String nombreAsignatura;</b>
<b>f</b>	<b>private double areaFigura;</b>
<b>g</b>	<b>private boolean estaAprobado;</b>

## Ejer 2.21 (Sol.)

<b>a</b>	<b>numero != 0</b>
<b>b</b>	<b>numero == 0</b>
<b>c</b>	<b>numero &gt;= 1 &amp;&amp; numero &lt;= 100</b>
<b>d</b>	<b>(numero &gt;= 1 &amp;&amp; numero &lt;= 100)    (numero &lt; 0)</b>
<b>e</b>	<b>numero % 2 == 0</b>
<b>f</b>	<b>numero % 4 == 0</b>
<b>g</b>	<b>(numero % 4 == 0) &amp;&amp; (numero % 100 != 0)</b>

<b>h</b>	<b>edad &gt;= 18</b>
<b>i</b>	<b>dia &gt;= 1 &amp;&amp; dia &lt;= 31</b>
<b>j</b>	<b>dia &lt; 1    dia &gt; 30</b>
<b>k</b>	<b>nota &gt;= 5</b>
<b>l</b>	<b>estadoCivil == 'S'    estadoCivil == 'C'</b>

# Algunos comentarios

- `nota >= 5`
  - en condiciones `if` y `while` va entre paréntesis
- `!(nota < 5)` - operador `!` unario
- `! nota < 5` – **Error!**
- `(a || b && c)`    `&&` se evalúa primero (`a,b,c` expresiones booleanas)
- `'A' < 'B'` es `true`
  - internamente se compara el código numérico de `'A'` (65) con el de `'B'` (66)
- `1 <= y <= 10` **Error!**
  - `(y >= 1) && (y <= 10)`

# Algunos comentarios

- **Leyes de Morgan**

- $\neg (A \ \&\& \ B) \implies \neg A \ || \ \neg B$
- $\neg (A \ || \ B) \implies \neg A \ \&\& \ \neg B$

A	B	A && B	!(A && B)	!A	!B	!A    !B
F	F	F	T	T	T	T
F	T	F	T	T	F	T
T	F	F	T	F	T	T
T	T	T	F	F	F	F

# Algunos comentarios

- **Leyes de Morgan**

- $\neg (A \ \&\& \ B) \implies \neg A \ || \ \neg B$
- $\neg (A \ || \ B) \implies \neg A \ \&\& \ \neg B$

- $\neg ((\text{edad} < 12) \ || \ (\text{edad} \geq 65)) \implies$

- $\neg (\text{edad} < 12) \ \&\& \ \neg (\text{edad} \geq 65) \implies$  (aplicada ley de Morgan)
- $(\text{edad} \geq 12) \ \&\& \ (\text{edad} < 65)$

- $\neg (\text{dia} \geq 1 \ \&\& \ \text{dia} \leq 31) \implies$

- $\neg (\text{dia} \geq 1) \ || \ \neg (\text{dia} \leq 31) \implies$  (aplicada ley de Morgan)
- $\text{dia} < 1 \ || \ \text{dia} > 31$

- $\neg ( (\text{valor} < 0) \ || \ (\text{valor} > 100) ) \implies$

- $\neg (\text{valor} < 0) \ \&\& \ \neg (\text{valor} > 100) \implies$  (aplicada ley de Morgan)
- $(\text{valor} \geq 0) \ \&\& \ (\text{valor} \leq 100)$



# Repaso y cuestionario

- Hacer cuestionario UT2
- 

- `!(x > 0) && (x > 0)` con `x = 12` **????**
- `(x > 0) || (x < 0)` **????**
- `(x != 1) == !(x == 1)` con `x = 1` **????**

# Repaso y cuestionario

- Hacer cuestionario UT2
- 
- `!(x > 0) && (x > 0)` con `x = 12` **false**
  - `(x > 0) || (x < 0)` **true**
  - `(x != 1) == !(x == 1)` con `x = 1` **true**

# Repaso y cuestionario

- Entrada gratuita en un museo
  - Ser menor de 18 años
  - Ser estudiante entre 18 y 25 años
  - Estar desempleado

```
private boolean esDesempleado;  
private boolean esEstudiante;  
private int edad;
```

```
public boolean tieneEntradaGratuita() {  
    return ??????????????????;  
}
```

# Repaso y cuestionario

- Entrada gratuita en un museo
  - Ser menor de 18 años
  - Ser estudiante entre 18 y 25 años
  - Estar desempleado

```
private boolean esDesempleado;  
private boolean esEstudiante;  
private int edad;
```

```
public boolean tieneEntradaGratuita() {  
    return  edad < 18 ||  
           (edad <= 25 && esEstudiante) ||  
           esDesempleado;  
}
```

# Repaso y cuestionario

- Una clase Laboratorio incluye los siguientes métodos:
  - el método `matricularEstudiante()` que dado un estudiante (objeto de la clase Estudiante) lo inscribe en el laboratorio
  - el método `numeroEstudiantes()` que devuelve el nº de estudiantes inscritos
  - el método `primerEstudiante()` que devuelve el estudiante que se matriculó en primer lugar
- Indica las firmas de dichos métodos

# Repaso y cuestionario

- Una clase Laboratorio incluye los siguientes métodos:
  - el método **matricularEstudiante** que dado un estudiante (objeto de la clase Estudiante) lo inscribe en el laboratorio  
**public void matricularEstudiante(Estudiante nuevoEstudiante)**
  - el método **numeroEstudiantes** que devuelve el nº de estudiantes inscritos  
**public int numeroEstudiantes()**
  - el método **primerEstudiante** que devuelve el estudiante que se matriculó en primer lugar  
**public Estudiante primerEstudiante()**