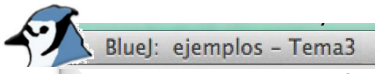
Tema 3. Variables: definición, tipos y uso en Java
Punto 1: Introducción a las variables en Java

- Definición (declaración) y tipo
- Clasificación según su tipo: variable de tipo Primitivo o Referencia
- Valores y reserva de memoria para una variable según su tipo
- Clasificación según su ámbito de declaración o rol: atributos, variables locales y parámetros



- **Descarga** de mi Tema 3 de PoliformaT el **proyecto BlueJ *ejemplos – Tema 3***
- **Clic** en el icono de BlueJ, para abrirlo
- Escribe en el **CodePad de Bluej** las instrucciones que ves a continuación

¿Cuál es el **estado de la memoria** tras ejecutarlas?
(sigue las indicaciones del profes@r para saberlo)

Introducción: **Variables**

... porque **representan "la información"** (datos, resultados, etc.) que es necesario manipular durante la resolución un problema

```
public class Circulo {
    private double radio;
    private String color;
    private int centroX, centroY;
    ...
    public double area() { return 3.14 * radio * radio; }
}
```

```
public class PrimerPrograma {
    public static void main(String[] args) {
        Pizarra miPizarra = new Pizarra("ESPACIO DIBUJO", 300, 300);
        Circulo c1 = new Circulo(50, "amarillo", 100, 100);
        miPizarra.add(c1);
        ...
    }
}
```

Introducción: **Tipo** de variable (I)

Observa las palabras que aparecen enmarcadas en estas clases.

¿Qué (único) nombre las describe a todas? ¿Por qué 2 colores de marco?

El **tipo** de una variable determina el conjunto de **valores** que puede almacenar
el conjunto de **operaciones** que se le pueden aplicar

```
public class Circulo {
     private double radio;
     private String color;
     private int centroX, centroY;
     ...
     public double area() { return 3.14 * radio * radio; }
}
```

```
public class PrimerPrograma {
    public static void main (String[] args) {
        Pizarra miPizarra = new Pizarra("ESPACIO DIBUJO", 300, 300);
        Circulo c1 = new Circulo(50, "amarillo", 100, 100);
        miPizarra.add(c1);
        ...
    }
}
```

Introducción: **Tipo de variable (II)**

Observa las palabras que aparecen enmarcadas en estas clases.

¿Qué (único) nombre las describe a todas? **¿Por qué 2 colores de marco?**

Primitivos: int, double, ...

Referencia: String, Circulo, Pizarra, ...

```
public class Circulo {
    private double radio ;
    private String color ;
    private int centroX , centroY ;
    ...
    public double area() { return 3.14 * radio * radio; }
}
```

¿almacena un **valor**?

¿almacena una **referencia** a un objeto?

```
public class PrimerPr
    public static void
        Pizarra miPizarra = new Pizarra("ESPACIO DIBUJO", 300, 300);
        Circulo c1 = new Circulo(50.0, "amarillo", 100, 100);
        miPizarra.add(c1);
        ...
    }
}
```

Introducción: **Tipo de variable (III)**

Tipos PRIMITIVOS Java (predefinidos)	**Nombre**
Numéricos - Enteros …… - Reales (en coma flotante) ……	 - `byte, short, int, long` - `float, double`
Caracteres	`char`
Lógicos	`boolean`

Tipos REFERENCIA Java	**Nombre**
Clases Tipo_de_Datos estándar en librerías como - `java.lang` …… - `java.util` ……	 - `System, String, Math …` - `Scanner …`
Clases de Usuario	- `Circulo, Pizarra, …`

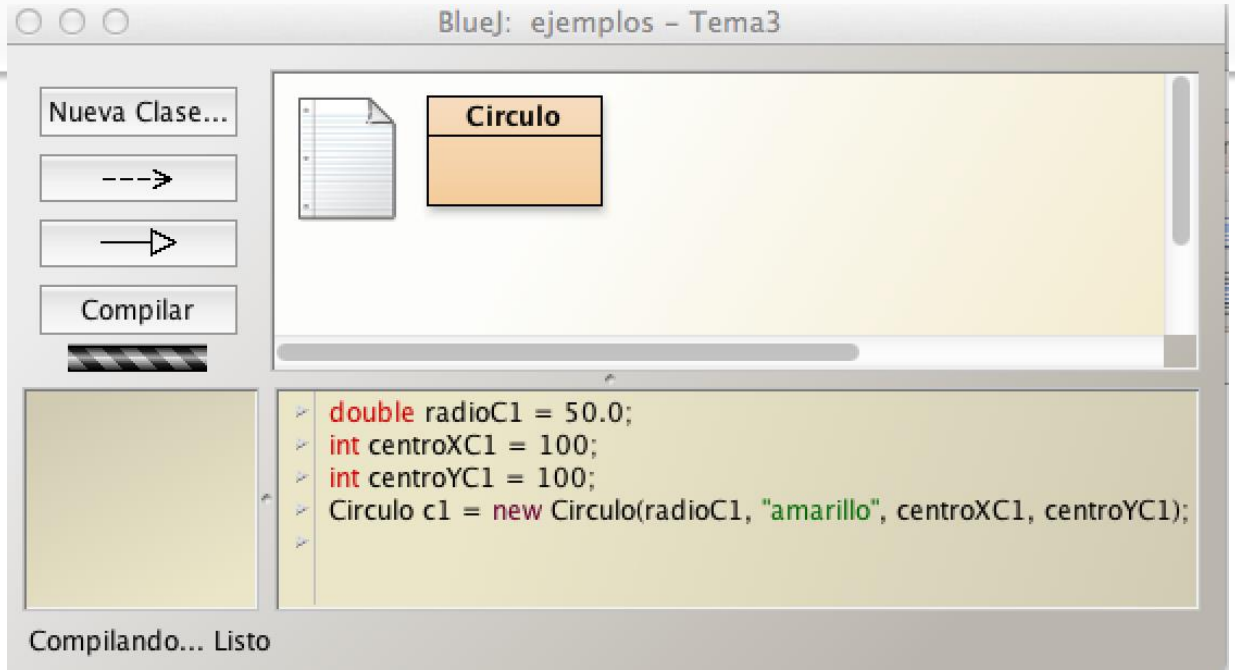
Introducción: **Memoria y tipo de variable (I)**

El **tipo** de una variable (Primitivo o Referencia) determina…

- el conjunto de **valores** que puede almacenar
- el conjunto de **operaciones** que se le aplican

Escribe en el **CodePad de Bluej** las siguientes instrucciones y, siguiendo las indicaciones del profes@r, responde: ¿cuál es el **estado de la memoria** tras su ejecución?

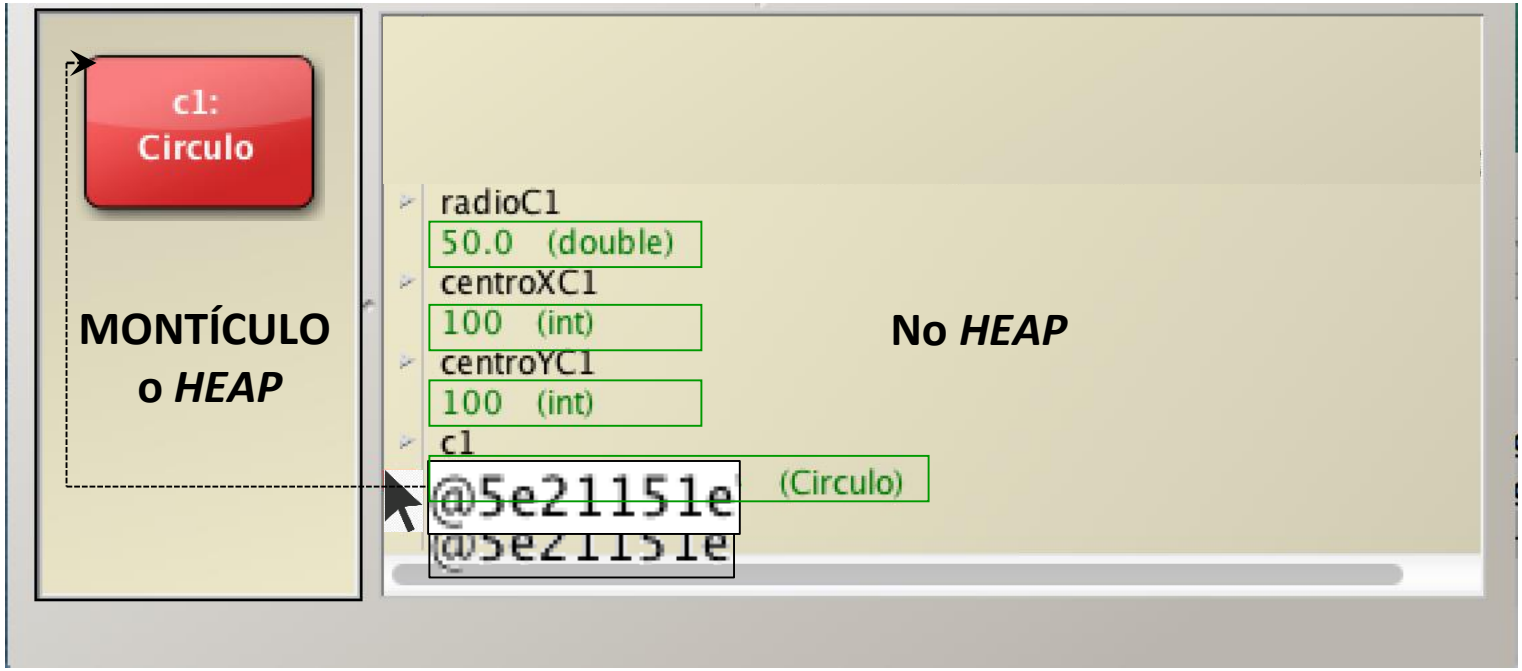
```
double radioC1 = 50.0;
int centroXC1 = 100, centroYC1 = 100;
Circulo c1 = new Circulo(radioC1, “amarillo”, centroXC1, centroYC1);
```



Introducción: **Memoria y tipo de variable (II)**

El **tipo** de una variable (Primitivo o Referencia) determina …

- el conjunto de **valores** que puede almacenar → **zona de memoria** que ocupa (*Heap* VS No *Heap*)
- el conjunto de **operaciones** que se le aplican



La variable c1 **referencia** al objeto … **PERO NO ES** el objeto

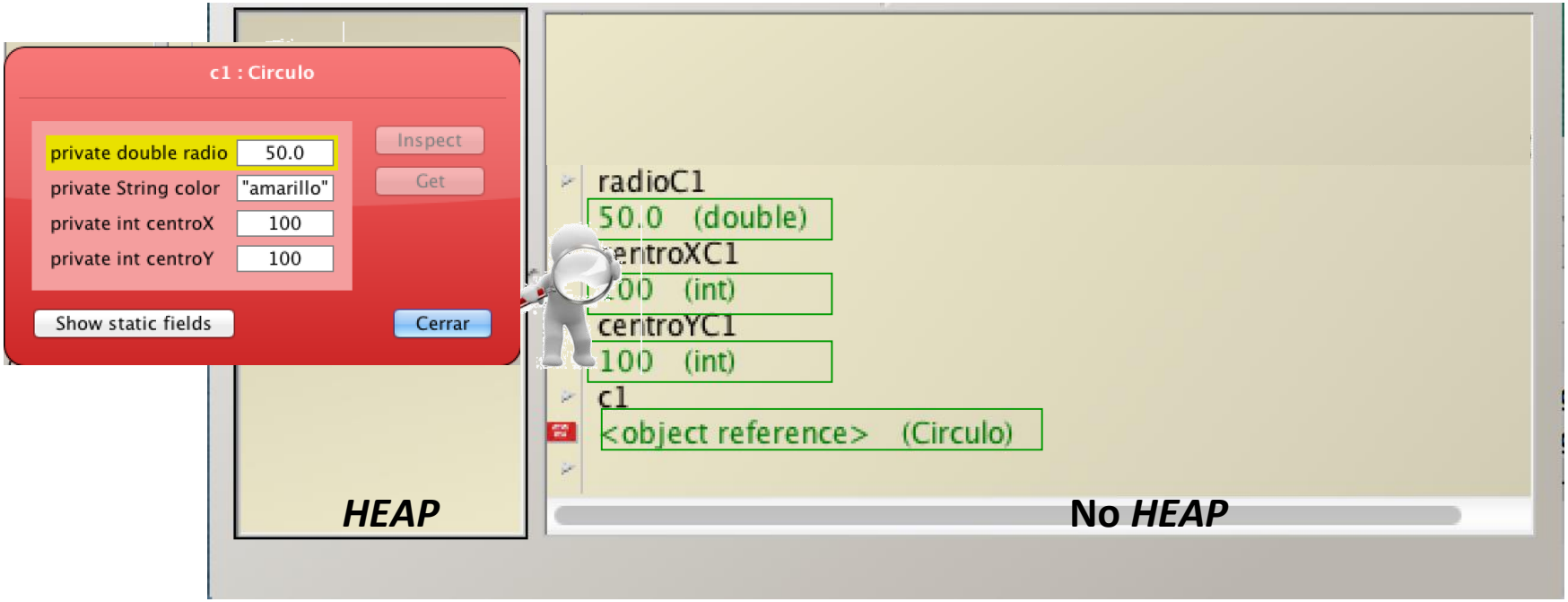
variable Referencia

Introducción: **Memoria y tipo de variable (III)**

El **tipo** de una variable (Primitivo o Referencia) determina …

- el conjunto de **valores** que puede almacenar
- el conjunto de **operaciones** que se le aplican

➔ **zona de memoria** y su **formato** (*Heap* VS No *Heap*)



Introducción: **Memoria y tipo de variable (IV)**

Tipo entero	Tamaño	Valor mínimo	Valor máximo
`byte`	8 bits	-128	127
`short`	16 bits	-32768	32767
`int`	32 bits	-2147483648	2147483647
`long`	64 bits	-2^{63}	$2^{63}-1$

Tipo real	Tamaño	Valor mínimo	Valor máximo	Precisión
`float`	32 bits	1.4×10^{-45}	3.4×10^{38}	7 dígitos
`double`	64 bits	4.9×10^{-324}	1.8×10^{308}	15

Tipo	Tamaño	Codificación
`char`	16 bits	Unicode

UTF-16

Tipo	Tamaño	Valores	Significado
`boolean`	1 bit	`true` `false`	Verdadero Falso

Tipo	Tamaño	Valores	Significado
Clase Java	32/64 bits	Direcciones de memoria	Referencia a un objeto de la clase

Codificación ASCII (7 bits), 128 primeros caracteres Unicode

Ejemplo - char 'A': fila 4 columna 1 -> 41 hexadecimal -> 65 decimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENO	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

Introducción: **Resumen (I)**

Recuerda que …

- La **información** (datos, resultados, etc.) a manipular durante la resolución de un problema se almacena en **variables**
- Una **variable** tiene asociado un **tipo de datos** que determina ...
  - el **conjunto de valores** que puede almacenar
    ⬇ la zona de memoria que ocupan y su formato
  - el **conjunto de operaciones** que se le pueden aplicar
- Primera **clasificación** de **tipos/variables Java:**
  - Primitivos ⬅➡ Variables que almacenan valores
  - Referencia ⬅➡ Variables que almacenan una referencia a un objeto
- **Declaración** de una variable en Java: **instrucción** que la define en Java
  - ¡OBLIGATORIA! **IMPOSIBLE** sino que almacene un valor o aplicarle una operación
  - **Sintaxis:** `tipo nombreVar1, nombreVar2, …, nombreVarN;`

parte de un bloque en el que la variable es conocida y se puede utilizar

Introducción: **Ámbito de declaración - rol de variable (I)**

Observa las siguientes **instrucciones** de declaración de variables Java

¿Son todas iguales? ¿Por qué?

```
public class Circulo {
    private double radio ;
    private String color ;
    private int centroX , centroY ;
    ...
    public void setRadio(double nuevo) { radio = nuevo; }
}
```

Atributos
- **SÍ** admiten modificadores, **ANTES** del tipo
- **Ámbito:** dentro de la clase, al menos

Parámetros

```
public class PrimerPrograma {
    public static void main(String[] args ) {
        Pizarra miPizarra;
        miPizarra = new Pizarra("ESPACIO DIBUJO", 300, 300);
        Circulo c1 ;
        ...
    }
}
```

Variables Locales -y Parámetros-
- **NO** admiten modificadores -excepto final-
- **Ámbito:** cuerpo del método donde se declaran

Introducción: **Resumen (II)**
Clasificación de variables en Java