

# Sesión 02: Objetos, referencias y variables

## Programación 2

---

Ángel Herranz

2019-2020

Universidad Politécnica de Madrid

# En capítulos anteriores...

- Sobre los IDEs
- Clases = *Plantillas*

# En el capítulo de hoy...

- Variables
- Valores primitivos
- Objetos
- Referencias

# ¿Qué significa esto?

```
class A {  
    int x = 42;  
}
```

# ¿Qué significa esto?

```
class A {  
    int x = 42;  
}
```

Plantilla para crear objetos de la clase A

# ¿Qué significa esto?

```
class A {  
    int x = 42;  
}
```

Plantilla para crear objetos del **tipo** A

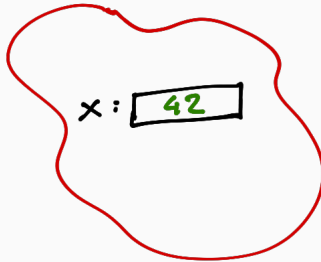
# ¿Qué significa esto?

```
class A {  
    int x = 42;  
}
```

Plantilla para crear objetos de la clase A

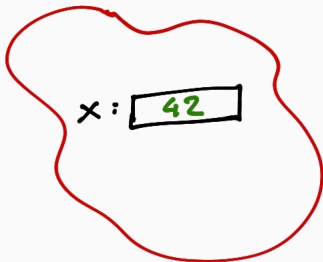
¿Y cómo son los objetos de la clase A?

# Objetos de la clase A





# Objetos de la clase A



El objeto tiene una variable  $x$  que resulta ser  
un dato entero<sup>1</sup>

---

<sup>1</sup>Formalmente un valor entre  $-2^{31}$  y  $2^{31} - 1$

# ¿Y cómo se crean?

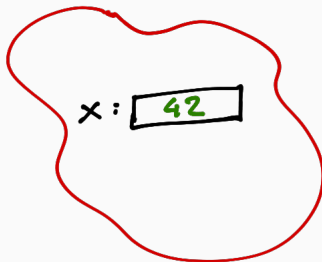
```
public class Sesion02 {  
    public static void main(String args[]) {  
        A a;  
        a = new A();  
    }  
}
```

a: null

# ¿Y cómo se crean?

```
public class Sesion02 {  
    public static void main(String args[]) {  
        A a;  
        a = new A();  
    }  
}
```

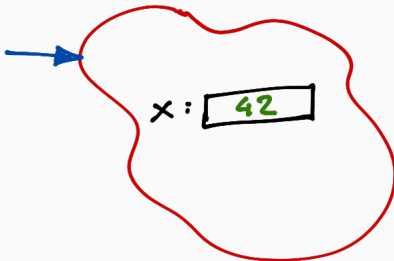
a: null



# ¿Y cómo se crean?

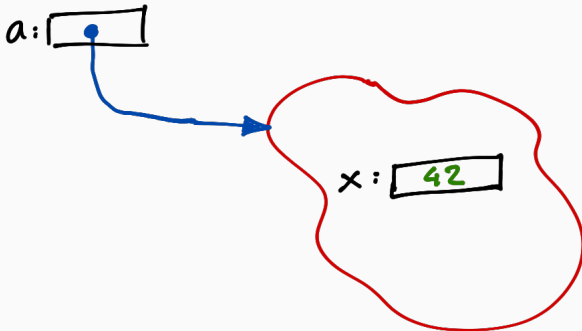
```
public class Sesion02 {  
    public static void main(String args[]) {  
        A a;  
        a = new A();  
    }  
}
```

a: null



# ¿Y cómo se crean?

```
public class Sesion02 {  
    public static void main(String args[]) {  
        A a;  
        a = new A();  
    }  
}
```

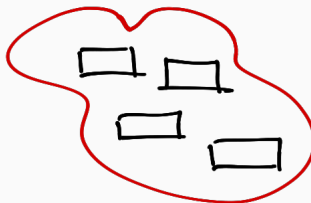


# Nuestro lenguaje gráfico

x:   
VARIABLE

42 3,1415  
true

DATOS PRIMITIVOS



OBJETO

null 

REFERENCIAS

# Variables

- Las representamos como **cajas**
- Sólo pueden contener
  - datos primitivos**<sup>2</sup>
  - o **referencias**<sup>3</sup>
- Cada variable tiene **su tipo**
  - Minúscula: tipos primitivos
  - Mayúscula: clases
- Las variables viven dentro de los objetos o en la **pila de ejecución** (métodos).

---

<sup>2</sup>Booleanos, caracteres, enteros, *float*

<sup>3</sup>**null** o una dirección de memoria

# Datos primitivos

- Los representamos en **verde**
- Son datos booleanos, caracteres, enteros o *floats*<sup>4</sup>
- Se guardan **dentro de las variables**

---

<sup>4</sup>`boolean, char, byte, short, int, long, float, double`



# Objetos

- Los representamos como **formas amorfas con cajas negras dentro**
- Cada caja es una **variable**
- No viven dentro de las variables
- Viven en el *Heap*
- Cada objeto tiene una dirección de memoria en la que vive
- Dicha dirección de memoria es resultado del **new**

# Referencias

- Son las direcciones de memoria de los objetos
- Las representamos como **flechas**
- Se guardan **dentro de las variables**
- Hay una referencia especial: **null**<sup>5</sup>

---

<sup>5</sup>Internamente es 0 aunque nunca se puede manejar como tal

# Sintaxis: operador new

*Crea un objeto*

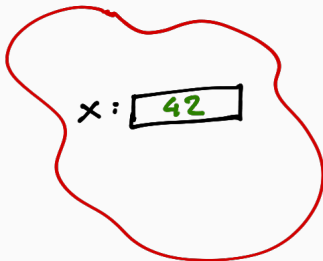
**new** \_

# Sintaxis: operador new

*Crea un objeto*

**new** \_

**new** A()



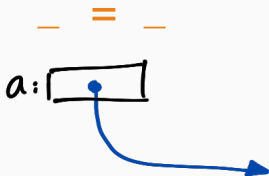
# Sintaxis: operador *asignación*

*Modifica la caja*

*\_ = \_*

# Sintaxis: operador *asignación*

*Modifica la caja*



`a = new A()`

# Sintaxis: operador *punto*

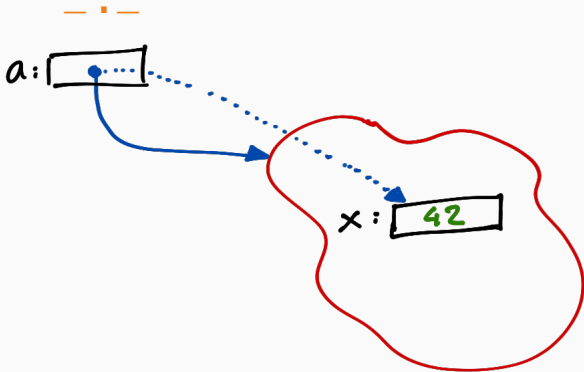
*Siga la felcha*



# Sintaxis: operador *punto*

*Siga la felcha*

a.x

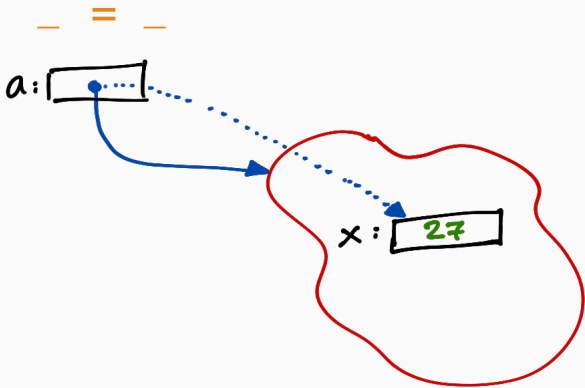




# Sintaxis: operador *asignación*

*Modifica la caja*

`a.x = 27`



## Experimenta y dibuja

- Implementa la clase A
- Escribe un programar principal que  
cree un objeto de la clase A  
e imprima su atributo en out
- ¿Puedes *imprimir el objeto*?
- Crea varios objetos e imprime “cosas que se te ocurran”
- ¡Dibuja!

## Dibuja, apuesta y experimenta i

```
A a1 = new A();  
A a2 = new A();  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

## Dibuja, apuesta y experimenta i

```
A a1 = new A();  
A a2 = new A();  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

42

42

false

true

## Dibuja, apuesta y experimenta ii

```
A a1 = new A();  
A a2 = new A();  
a2.x = 27;  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

## Dibuja, apuesta y experimenta ii

```
A a1 = new A();  
A a2 = new A();  
a2.x = 27;  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

42

27

false

false

## Dibuja, apuesta y experimenta iii

```
A a1 = new A();  
A a2 = a1;  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

## Dibuja, apuesta y experimenta iii

```
A a1 = new A();  
A a2 = a1;  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

42

42

true

true



## Dibuja, apuesta y experimenta iv

```
A a1 = new A();  
A a2 = a1;  
a2.x = 27;  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

## Dibuja, apuesta y experimenta iv

```
A a1 = new A();  
A a2 = a1;  
a2.x = 27;  
System.out.println(a1.x);  
System.out.println(a2.x);  
System.out.println(a1 == a2);  
System.out.println(a1.x == a2.x);
```

---

27

27

true

true

¿Puedo hacer esto?

```
System.out.println(new A().x);
```

¿Puedo hacer esto?

```
System.out.println(new A().x);
```

=

```
System.out.println(new A().x);
```

# ¿Para qué sirven las clases?

# ¿Para qué sirven las clases?

## Modelización

# ¿Para qué sirven las clases?

## Modelización

Por ejemplo: nuestro spotify

# ¿Para qué sirven las clases?

## Modelización

Por ejemplo: nuestro spotify, o nuestro pequeño motor de física



# ¿Para qué sirven las clases?

## Modelización

Por ejemplo: nuestro spotify, o nuestro pequeño motor de física, o nuestra zapatería. . .

# ¿Para qué sirven las clases?

## Modelización

Por ejemplo: **nuestro spotify**, o nuestro pequeño motor de física, o nuestra zapatería. . .

# Modelizamos las canciones

*Everything should be built top-down, except the first time.<sup>6</sup>*

*Epigrams on Programming (Alan J. Perlis)*

- ¿Cómo serán los datos con los que vamos a representar una canción?



---

<sup>6</sup>No vamos a definir la clase perfecta a la primera

# Modelizamos las canciones

*Everything should be built top-down, except the first time.<sup>6</sup>*

*Epigrams on Programming (Alan J. Perlis)*

- ¿Cómo serán los datos con los que vamos a representar una canción?
-  Definir una clase que represente canciones
-  Escribir un programa que pruebe que lo que hacemos tiene cierto sentido

---

<sup>6</sup>No vamos a definir la clase perfecta a la primera

# Propuesta de datos de una canción

*There are only two hard things in Computer Science: cache invalidation and **naming things**.*

*Phil Karlton*

---

<sup>7</sup>Esta lista de nombres representa una votación en clase

# Propuesta de datos de una canción

*There are only two hard things in Computer Science: cache invalidation and naming things.*

*Phil Karlton*

explicit, nombre, título, artista, interprete, autor,  
compositor, duración, álbum, audio, sonidos,  
notas, imagen, valoración<sup>7</sup>

---

<sup>7</sup>Esta lista de nombres representa una votación en clase