

# Uso de clases y objetos en Java

Jorge I. Meza

[jimezam@autonoma.edu.co](mailto:jimezam@autonoma.edu.co)



## Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International

This license requires that reusers give credit to the creator. It allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, for noncommercial purposes only. If others modify or adapt the material, they must license the modified material under identical terms.

👤 **BY:** Credit must be given to you, the creator.

🚫 **NC:** Only noncommercial use of your work is permitted.

Noncommercial means not primarily intended for or directed towards commercial advantage or monetary compensation.

🔄 **SA:** Adaptations must be shared under the same terms.

# Contenidos

- Implementación de clases
- Instanciación de objetos
- Valor `null`
- Invocación de atributos y métodos
- Referencias y objetos
- Caso particular: *Strings*

# Clase Persona

```
// Persona.java
```

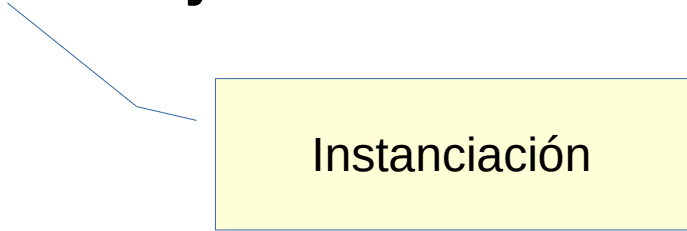
```
class Persona {  
    String nombre;  
    int edad;  
  
    void respirar() {  
        System.out.println  
            ("Estoy respirando");  
    }  
  
    // ...  
}
```

```
void saludarAmigo(Persona p) {  
    System.out.println  
        ("Hola " + p.nombre);  
}  
  
boolean soyMayorDeEdad() {  
    if(edad > 18)  
        return true;  
  
    return false;  
}  
}
```

# Conceptos clave

Receta → Torta

Clase → Objeto



Instanciación

# Instanciación o creación de objetos

- Las clases sirven como plantillas para crear a los objetos. Definen sus atributos y métodos.
- Los objetos se deben crear antes de poder ser utilizados en el programa.
- La creación de un objeto es un proceso de dos pasos que se puede o no hacer en una sola línea de código.

# Instanciación o creación de objetos

```
Persona j u a n = new Persona ( ) ;
```

Clase que  
servirá como  
plantilla para el  
objeto

Identificador  
del objeto  
creado

Método  
constructor  
(Lo conocemos  
más adelante)

Operador para  
la creación de  
objetos

# Instanciación o creación de objetos

```
Persona j u a n = new Persona ( ) ;
```

Se crea el objeto `j u a n` de tipo `Persona` utilizando el constructor `Persona ( )`.



# Instanciación o creación de objetos

```
Persona juan;
```

...

...

```
juan = new Persona ();
```

**Declaración** del objeto  
juan de tipo Persona

**Creación** del objeto juan  
utilizando el constructor  
Persona ()

# Instanciación o creación de objetos



```
Persona juan;
```

```
...
```

```
juan.nombre = "lolo";
```

```
...
```

```
juan = new Persona();
```

En este segmento de código, antes del `new`, el valor del objeto `juan` es `null`, por lo tanto no está listo para ser utilizado.

# Instanciación o creación de objetos

**Exception** in thread "main"

**java.lang.NullPointerException:**

Cannot assign field "nombre" because  
"<local1>" is null

at Persona.main(Persona.java



# Valor nulo

null

- Es un valor que **sólo puede aplicarse a objetos** (no a tipos simples de datos).
- Significa la ausencia de un valor conocido.
- Es el valor de un objeto después de su declaración pero antes de su creación.

# Valor nulo

```
Perro p;
```

```
System.out.println(p);
```

**null**

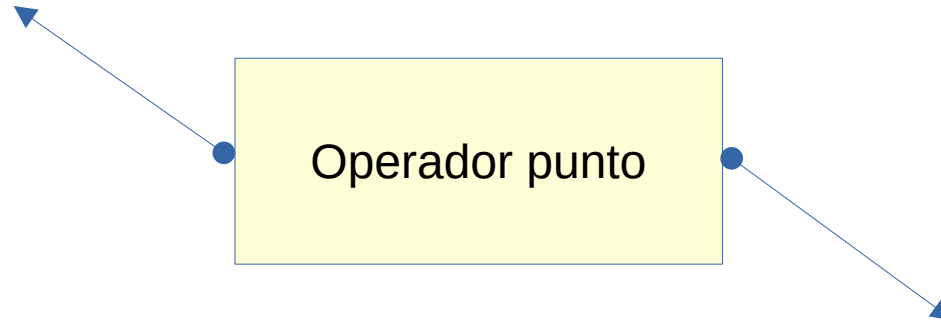
```
p = new Perro();
```

```
System.out.println(p);
```

**Perro@1dbd16a6**

# Invocación de atributos

```
Persona conductor = new Persona();  
conductor.nombre = "Juan";
```

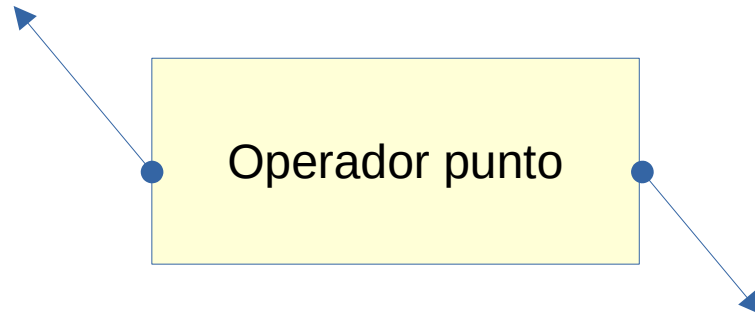


```
System.out.println(conductor.nombre);
```

**Juan**

# Invocación de métodos

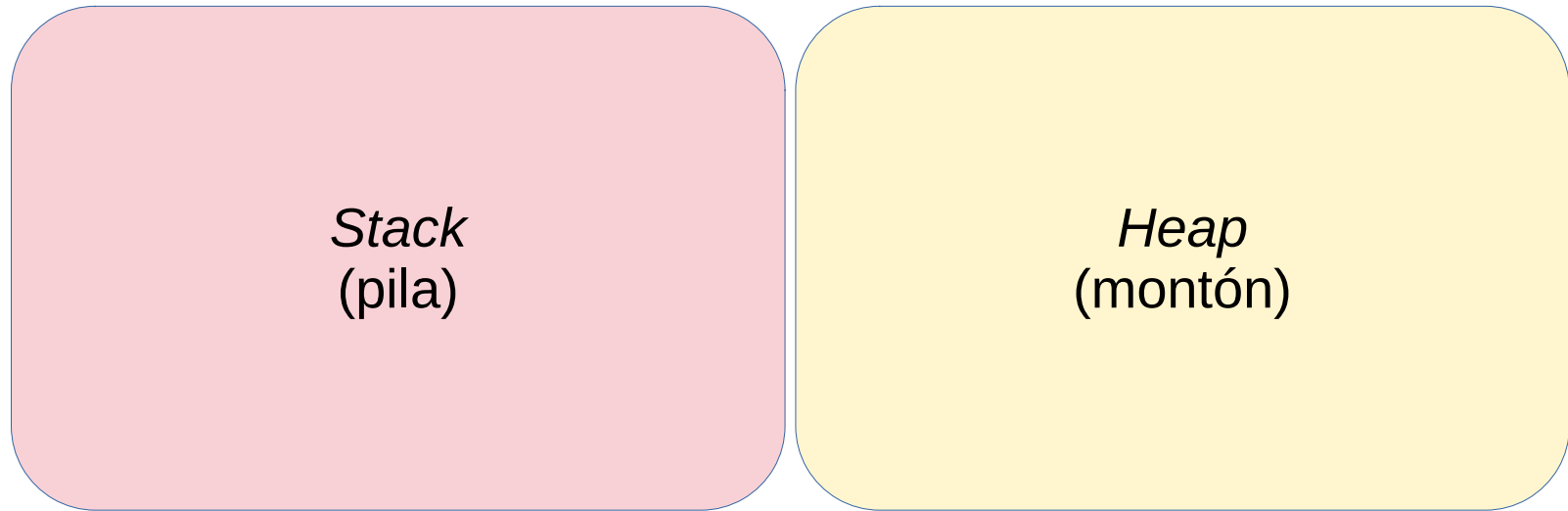
```
Persona conductor = new Persona();  
conductor.respirar();
```



```
String nombre = conductor.getNombre();
```

# Referencias y objetos

## Memoria de la máquina virtual de Java (JVM)





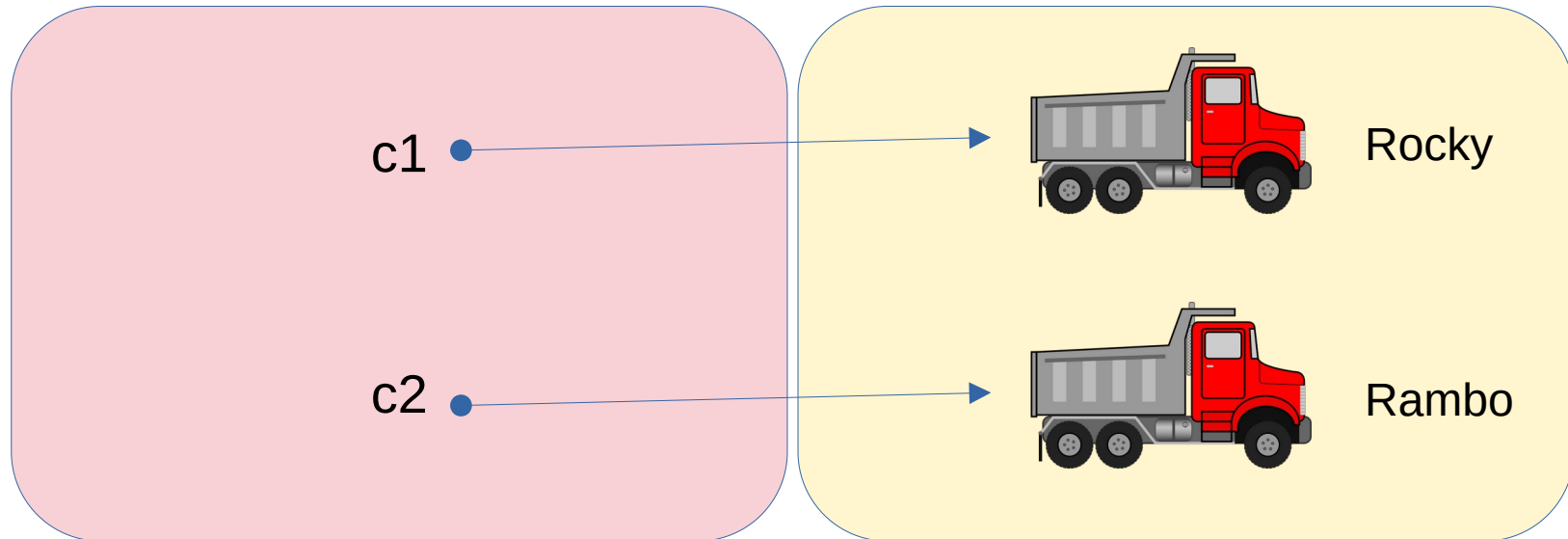
# Referencias y objetos

<i>Stack (Pila)</i>	<i>Heap (Montón)</i>
LIFO	Estructura dinámica
Tamaño fijo	Grande y flexible
<b>Variables locales, parámetros</b>	<b>Objetos</b>
Vida corta (mientras el método está en ejecución)	Vida más larga (hasta que el garbage collector lo libera)
Acceso local al método	Acceso compartido

# Referencias y objetos

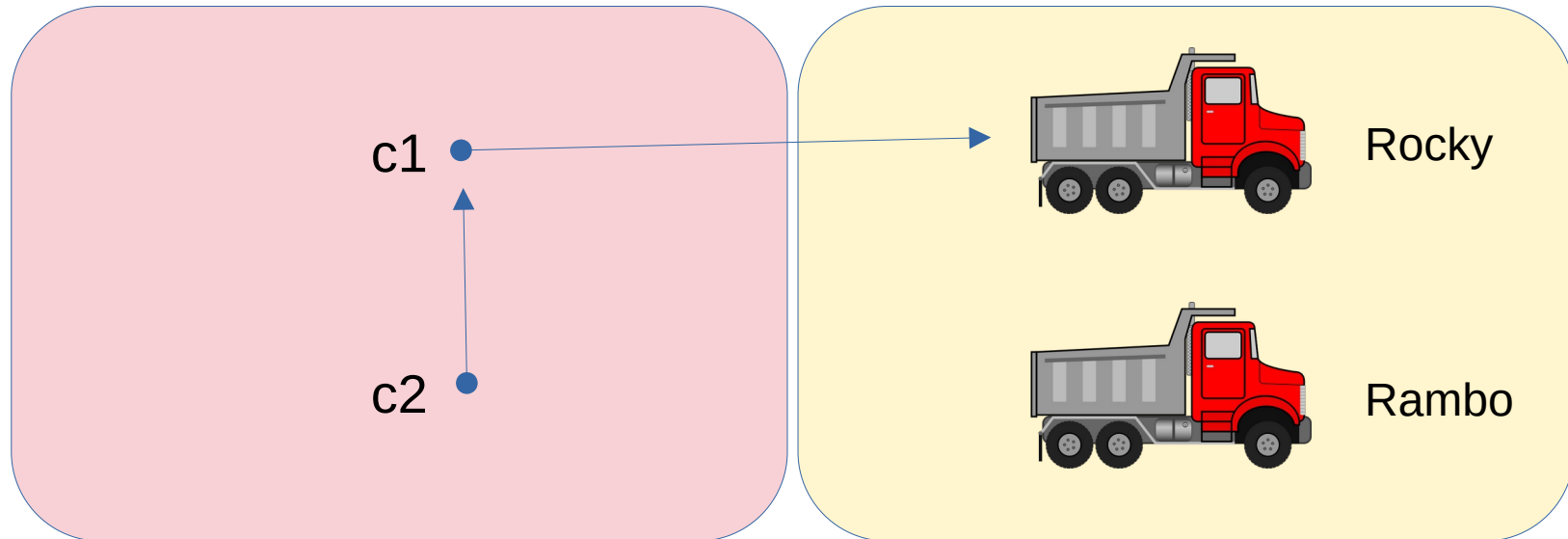
```
Camion c1 = new Camion("Rocky")
```

```
Camion c2 = new Camion("Rambo")
```



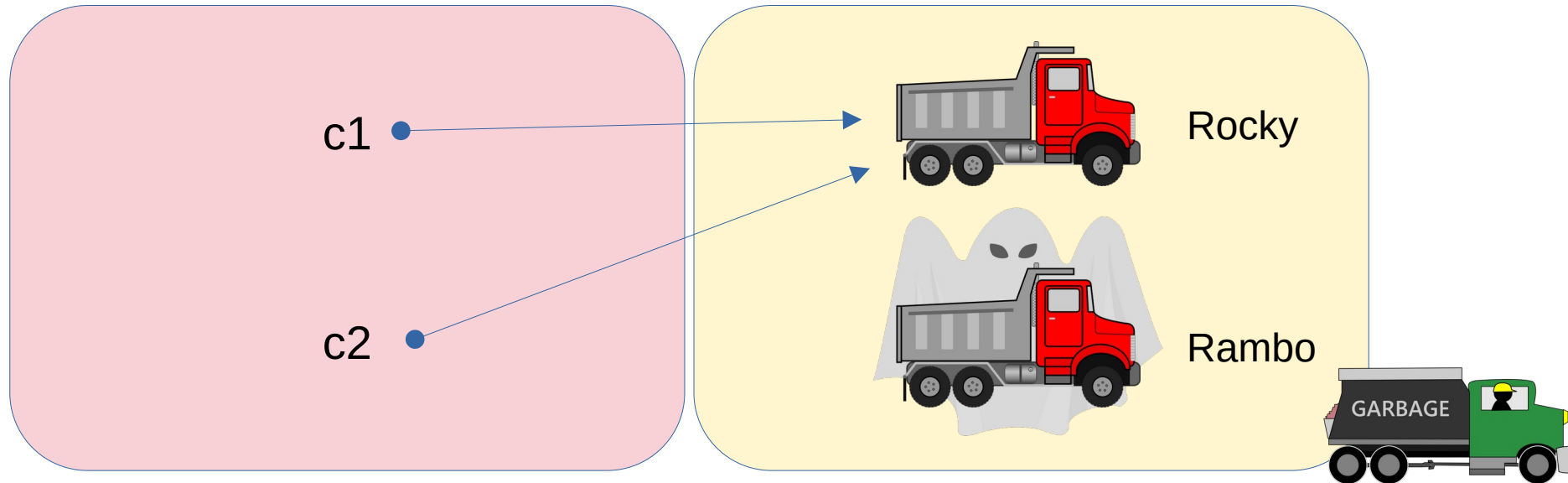
# Referencias y objetos

```
Camion c1 = new Camion("Rocky")  
Camion c2 = new Camion("Rambo")  
c2 = c1
```



# Referencias y objetos

```
Camion c1 = new Camion("Rocky")  
Camion c2 = new Camion("Rambo")  
c2 = c1
```



# Caso particular: *Strings*

```
String s11 =  
"Hola";
```

```
String s12 =  
"Hola";
```

```
System.out.println  
(s11 == s12);
```

```
String s21 = new  
String("Hola");
```

```
String s22 = new  
String("Hola");
```

```
System.out.println  
(s21 == s22);
```

# Caso particular: *Strings*

```
String s11 =  
"Hola";  
String s12 =  
"Hola";  
System.out.println  
(s11 == s12);  
true
```

```
String s21 = new  
String("Hola");  
String s22 = new  
String("Hola");  
System.out.println  
(s21 == s22);  
false
```

# Caso particular: *Strings*

- Cuando se vaya a **comparar igualdad de *cadenas* (u *objetos* en general)**, es mejor utilizar el método `equals` en lugar del operador `==`.
- Cuando se vaya a **comparar igualdad de tipos de *datos simples* (como `int`, `float`, `char`, `byte`, `boolean`, etc)**, es seguro utilizar el operador `==`.

# Caso particular: *Strings*

```
String s1;  
String s2;  
if (s1 == s2) {  
    // ...  
}  
if (s1.equals(s2))  
{  
    // ...  
}
```

```
long x1;  
long x2;  
if (x1 == x2) {  
    // ...  
}  
if (x1.equals(x2))  
{  
    // ...  
}
```

¡Los tipos simples de  
datos no tienen  
métodos!





¿Alguna pregunta?