

# TEMA: INTRODUCCIÓN A POO OBJETOS EN JAVA

---

Taller de Programación.

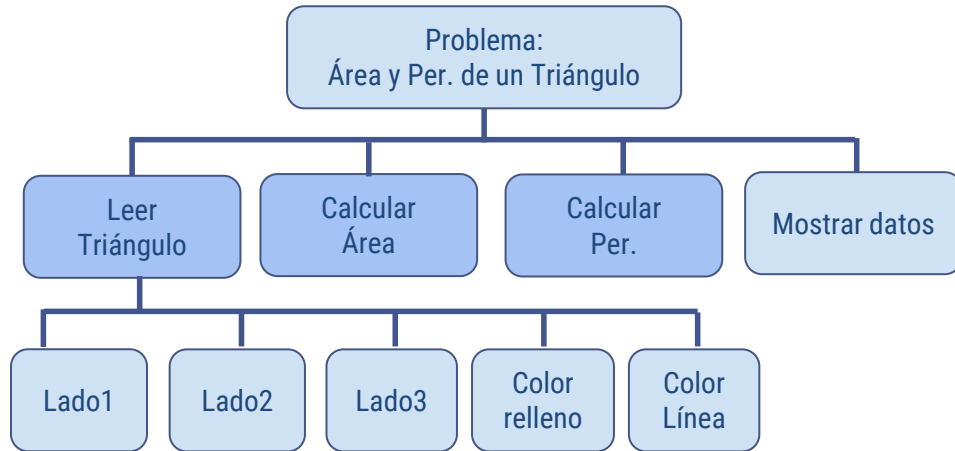
Módulo: Programación Orientada a Objetos

# Paradigmas de programación

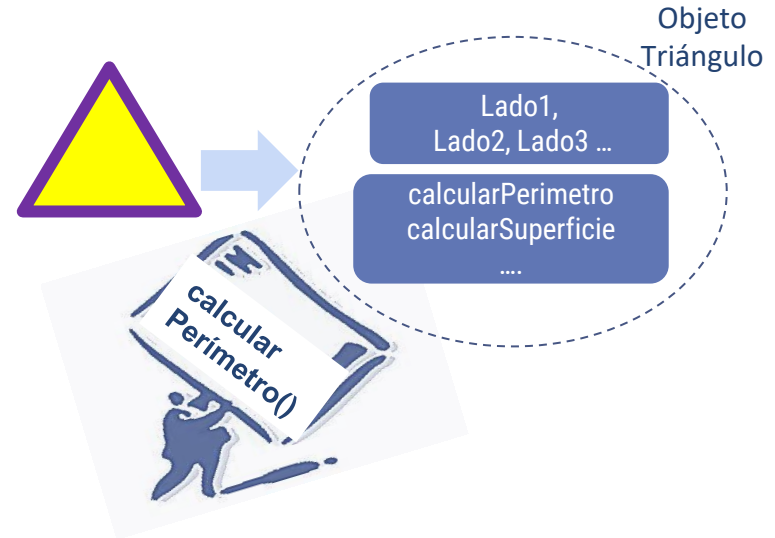


- Indica la manera de estructurar y organizar las tareas de nuestro programa.
- Los lenguajes de programación suelen ser multiparadigma.
- Hasta ahora: Imperativo
- Este curso: POO

# Paradigmas de programación



## Desarrollo Orientado a Objetos



¿Qué paradigma utilizar?

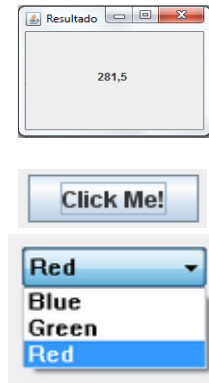
# Conceptos básicos de POO. Objeto

- **Objeto:** *abstracción* de un objeto del mundo real, definiendo qué lo caracteriza (estado interno) y qué acciones sabe realizar (*comportamiento*).
- ¿Qué cosas son objetos? *“Todo es un objeto”*

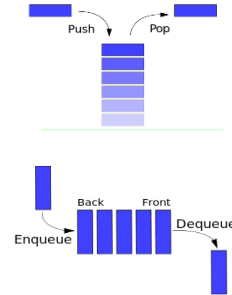
Objetos  
Físicos



Elementos de  
interfaces  
gráficas



Estructuras de  
datos



Seres vivos



Roles



# Conceptos básicos de POO. Objeto

- Ejemplos

**Características:**

Raza  
Edad en años  
Color pelaje

**Comportamiento:**

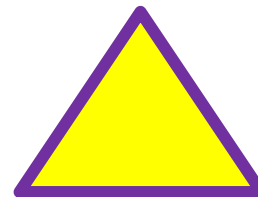
ladrar / gruñir / aullar  
(entre otras)

**Características:**

Marca  
Color  
Velocidad

**Comportamiento :**

arrancar / frenar / acelerar  
(entre otras)

**Características:**

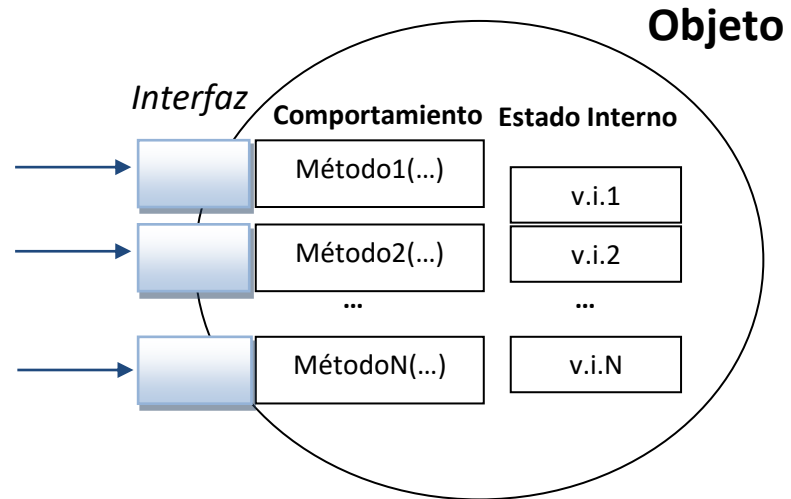
Lado1  
Lado2  
Lado3  
Color de línea  
Color de relleno

**Comportamiento :**

calcular área /  
calcular perímetro /  
(entre otras)

# Conceptos básicos de POO. Objeto

- **Objeto:** entidad que combina en una unidad
  - **Estado interno:** compuesto por datos/atributos que caracterizan al objeto y relaciones con otros objetos con los cuales colabora. Se implementan a través de **variables de instancia**.
  - **Comportamiento:** acciones o servicios a los que sabe responder el objeto. Se implementan a través de **métodos de instancia** que operan sobre el estado interno. Los servicios que ofrece al exterior constituyen la **interfaz**.

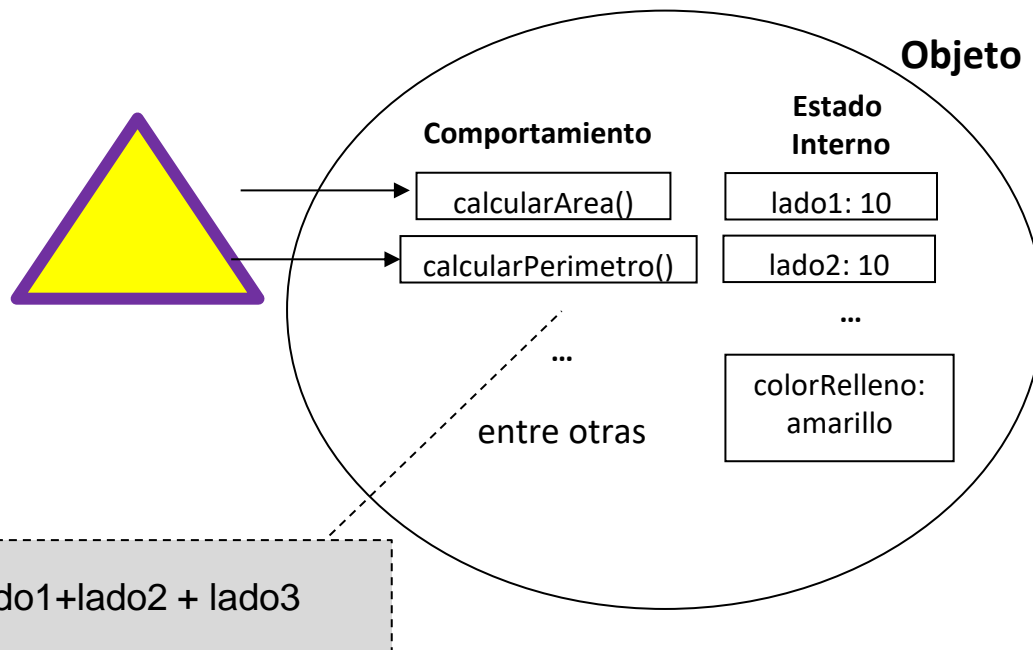


## **Encapsulamiento (ocultamiento de información)**

Se oculta la implementación del objeto hacia el exterior. Desde el exterior sólo se conoce la interfaz del objeto. Facilita el mantenimiento y evolución del sistema ya que no hay dependencias entre las partes del mismo.

# Conceptos básicos de POO. Objeto

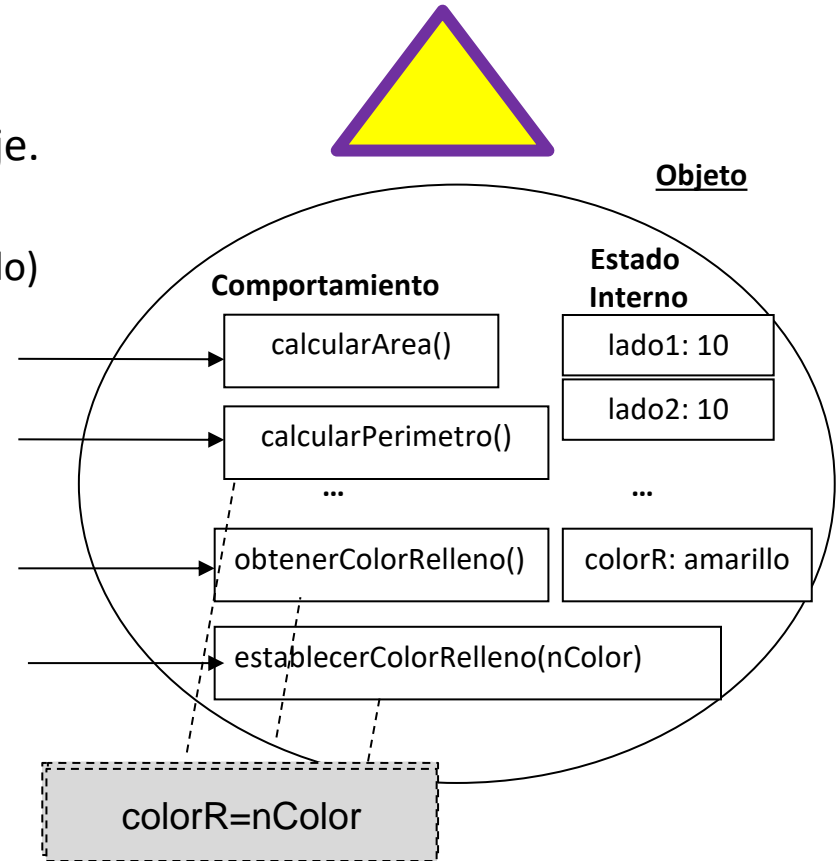
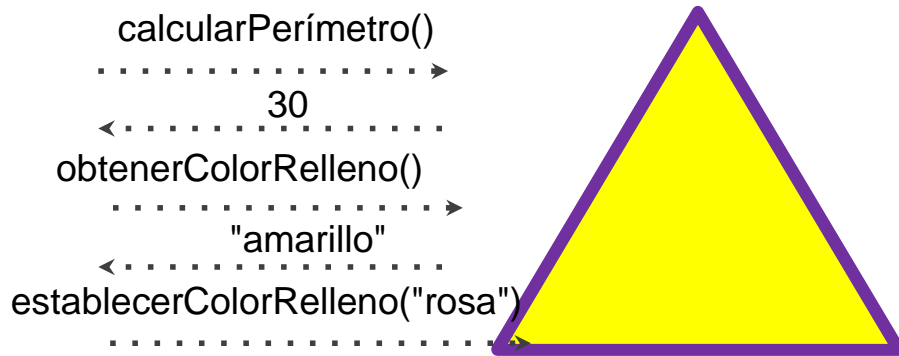
*Todo cómputo en la aplicación es realizado por objetos*



¿Cómo le pido al objeto que calcule el perímetro y me lo devuelva?

# Conceptos básicos de POO. Mensaje

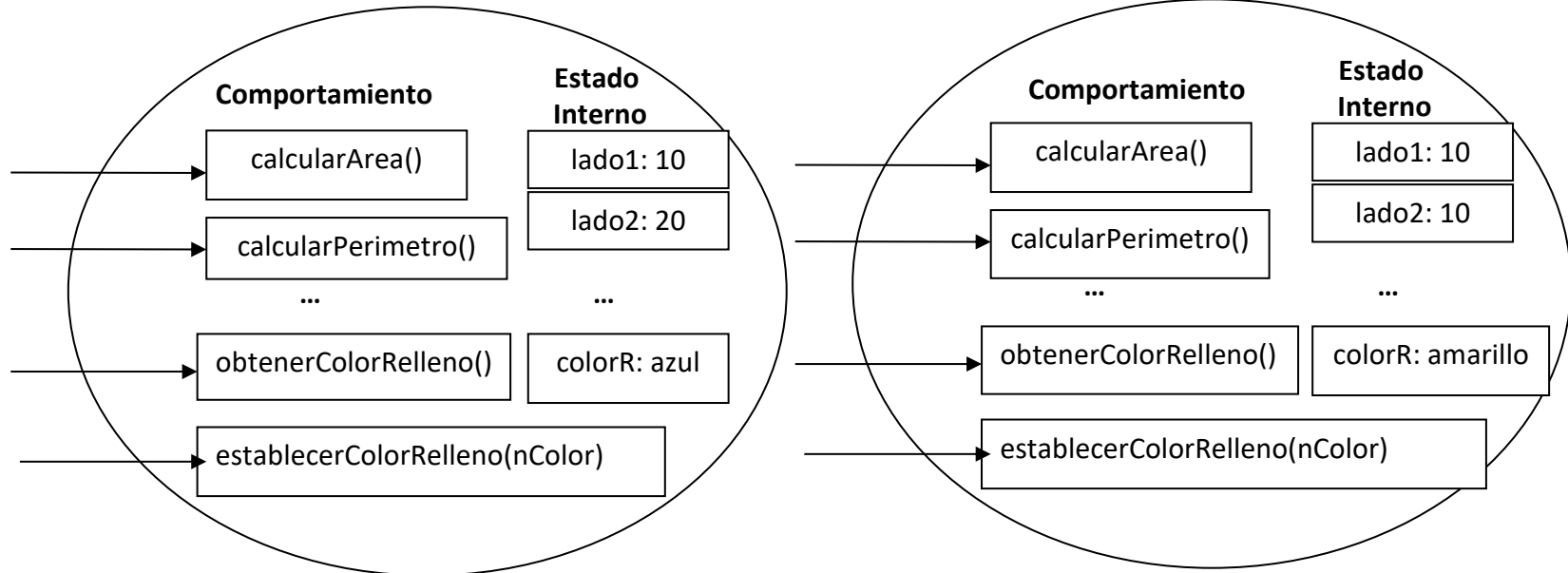
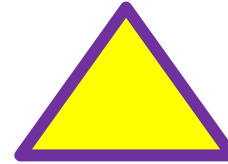
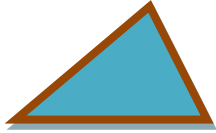
- **Envío de Mensaje:** provoca la ejecución del método indicado por el nombre del mensaje.
  - Puede llevar datos (parámetros del método)
  - Puede devolver un dato (resultado del método)





# Conceptos básicos de POO. Clase

- ¿Cuántos objetos ves?



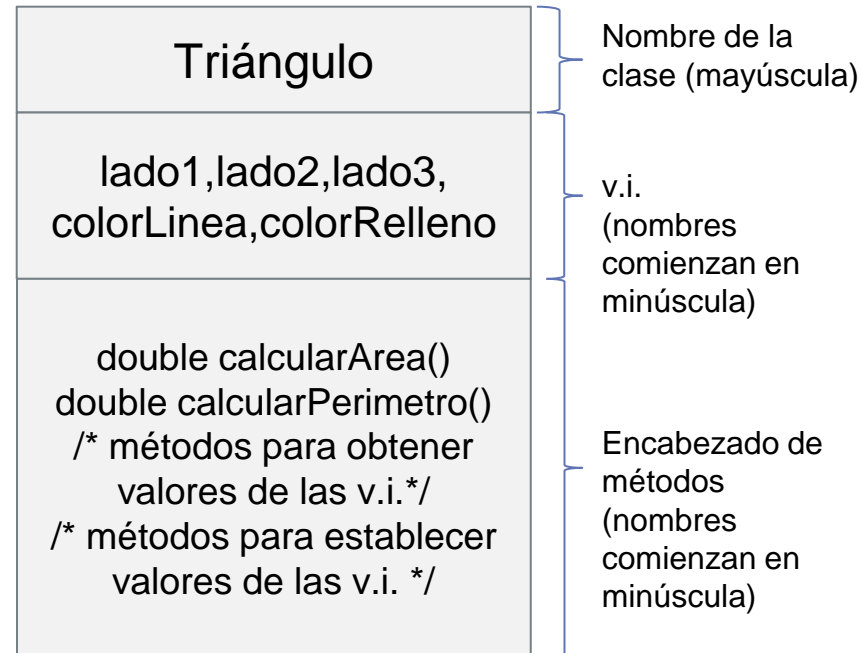
# Conceptos básicos de POO. Clase

- Una *clase* describe un conjunto de objetos comunes (mismo tipo).

## Consta de:

- La declaración de las v.i. que implementan el estado del objeto.
- La codificación de los métodos que implementan su comportamiento.
- Un objeto se crea a partir de una clase (el objeto es *instancia* de una clase).

- Representación gráfica de una clase



# Conceptos básicos de POO. **Instanciación** (creación de objeto)

- La *instanciación* se realiza enviando un mensaje de creación a la clase.

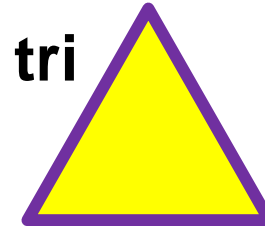
- Reserva de espacio para el objeto.
- Ejecución el código inicializador o *constructor*

Constructor: puede tomar valores pasados en el mensaje de creación. Inicializa el objeto (v.i.s) con valores recibidos.

- Devuelve la referencia al objeto.
- Asociar la referencia a una variable (a través de ella podemos enviarle mensajes al objeto).

new Triangulo (10,10,10,  
"amarillo","violeta")

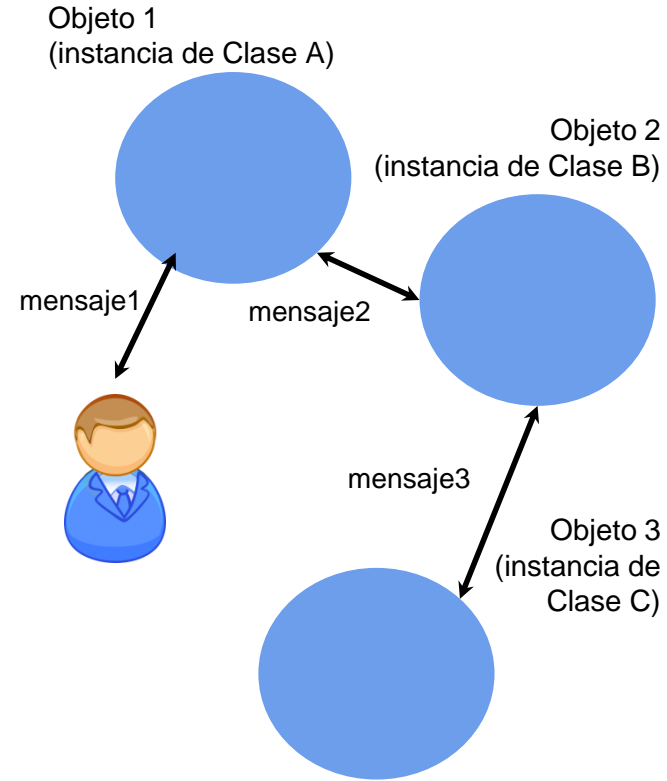
----->  
referencia  
<-----



Triángulo
lado1,lado2,lado3, colorLinea,colorRelleno
double calcularArea() double calcularPerimetro()  /* métodos para obtener valores de las v.i. */ /* métodos para establecer valores de las v.i. */

# Programa orientado a objetos

- Los programas se organizan como una **colección de *objetos*** que cooperan entre sí enviándose mensajes.
- Cada objeto es instancia de una ***clase***.
- Los objetos se crean a medida que se necesitan.
- El **usuario** le envía un mensaje a un objeto, en caso de que un objeto conozca a otro puede enviarle un mensaje, así los mensajes fluyen por el sistema.
- Cuando los objetos ya no son necesarios se borran de la memoria.



# Desarrollo de SW Orientado a Objetos

## Pasos:

- Identificar los objetos a abstraer en nuestra aplicación.

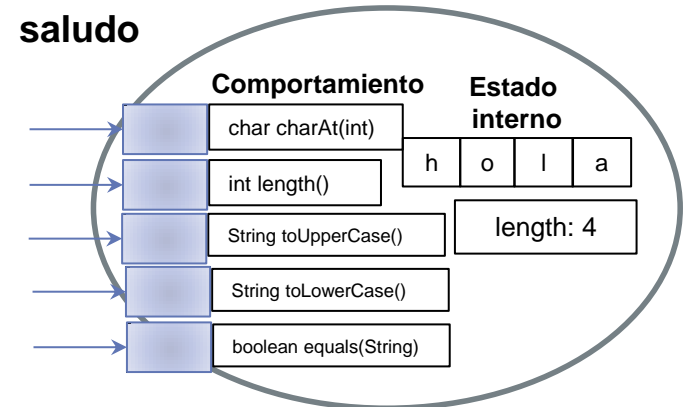
*“Lea las especificaciones del sistema que desea construir.*

*Subraye los sustantivos si su objetivo es un programa orientado a objetos”. Grady Booch*

- Identificar las características relevantes de los objetos
  - Identificar las acciones relevantes que realizan los objetos
- 
- Los objetos con características y comportamiento similar serán instancia de una misma *clase*.

# Objetos en Java

- Java incluye bibliotecas de clases que permiten crear objetos de uso común.
- Ej. clase *String*, clase *Point2D.Double*, clase *Date*, colecciones, ...
- En general se crean enviando un mensaje de creación a la clase (*new*).
- ¿Qué es un string? Es un objeto!!!
  - `String saludo = "hola";`
  - Otra forma:
    - `String saludo = new String("hola");`



# Objetos en Java. **Instanciación** (creación de objeto)

- Declarar variable para mantener la referencia:

```
NombreDeClase miVariable;
```

- Enviar a la clase el mensaje de creación y guardar referencia:

```
miVariable= new NombreDeClase(valores para inicialización);
```

- Se puede unir los dos pasos anteriores:

```
NombreDeClase miVariable= new NombreDeClase(...);
```

## Ejemplo

```
String saludo;
```

```
saludo= new String("hola");
```

```
String saludo = new String ("hola");
```

- Secuencia de pasos en la instanciación (creación de objeto):

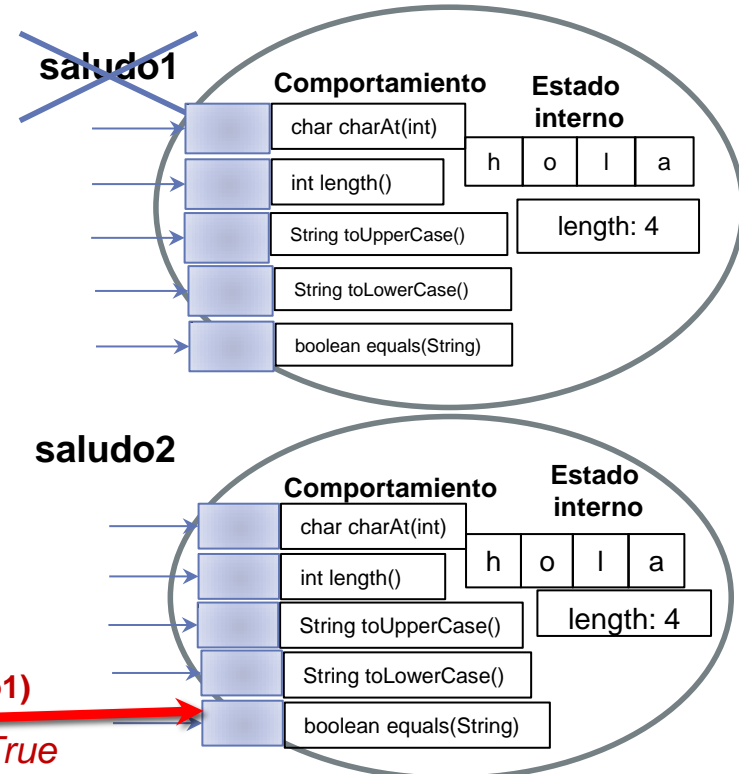
- *Reserva de Memoria*. Las variables de instancia se inicializan a valores por defecto o explícito (si hubiese).
- *Ejecución del Constructor* (código para inicializar variables de instancia con los valores que enviamos en el mensaje de creación).
- *Asignación de la referencia a la variable*.

# Objetos en Java. Referencias

- Referencia a un objeto: ubicación en memoria del objeto.
- El valor null.
- Ejemplo
  - `String saludo1 = new String("hola");`
- Asignación: copia referencias.
  - `String saludo2 = new String("hola");`
  - `saludo1 = saludo2;`
- Recolector de basura:
  - libera memoria de objetos no referenciados.
- Comparación de objetos con `==` y `!=`
  - *Comparan referencias*
  - `System.out.println(saludo1==saludo2);` **False**
- Comparación del contenido de objetos
  - *Enviar mensaje equals al objeto, pasando como argumento el objeto a comparar*

**saludo2.equals(saludo1)**

**True**





# Envío de mensaje al objeto

<https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

## Sintaxis

Puede devolver resultado

Objeto receptor del mensaje

Nombre del mensaje

Argumentos

**objeto.nombreMétodo(...);**

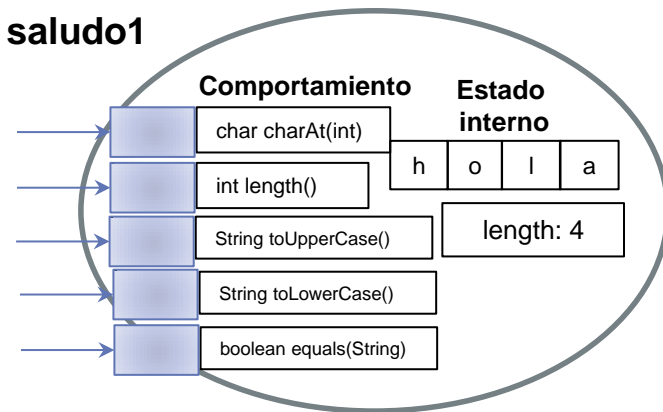
## Ejemplo

```
public class Demo01EnvioMensaje {
    public static void main(String[] args) {
        String saludo1 = new String("hola");
        System.out.println(saludo1.length());           //Imprime 4
        System.out.println(saludo1.charAt(0));           //Imprime h
        System.out.println(saludo1.toUpperCase().equals("HOLA"));
    }
}
```

1) Envío de msg toUpperCase a saludo1  
Devuelve un objeto String

2) Envío msg equals al objeto retornado por  
saludo1.toUpperCase()

**saludo1**



Regla de precedencia:  
los mensajes se  
ejecutan de izq a der

//Imprime true