



Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😊 

➤ El Paradigma de Orientación a Objetos

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Comprender cómo realizar la depuración en el IDE Eclipse
- ✓ Aprender a documentar código con JavaDoc



LEARNING PATHWAY

4.5

Start! 🏁

**El Paradigma de
Orientación a Objetos**

Clases y objetos: Atributos y
Estado

Clases y Objetos

Cuenta Bancaria

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Reconocer la importancia de la Orientación a Objetos



Comprender el concepto e implementación de las Clases y los Objetos





Rompehielo 🧊

¿Y tú que piensas?: 🙌

Vean con atención la imagen y respondan en el chat y levantando la mano:

- ¿Son los mismos objetos?
- ¿Qué características tienen en común?
- ¿En qué se diferencian?
- ¿Cómo los clasificarías/agruparías?



Sedan



Wagon/Minivan/MPV



SUV



Hatchback



Cabriolet/Coupe



Sport/Supercar



Pickup



4WD



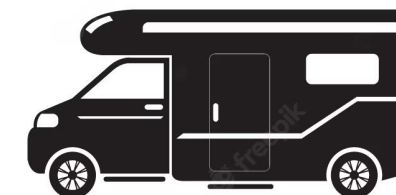
Mini Truck



Micro/Electric



Van



Campervan

➤ El Paradigma de Orientación a Objetos



El Paradigma de Orientación a Objetos

¿Qué es un Paradigma de programación?:

Es una manera o estilo de programación. Existen diferentes formas de diseñar un programa y varios modos de trabajar para obtener los resultados que necesitan los programadores.

La Programación Orientada a Objetos (**POO**) es un paradigma que modela elementos del mundo real como objetos que tienen propiedades y comportamientos. Esto permite representar entidades y sus características para crear programas más intuitivos.

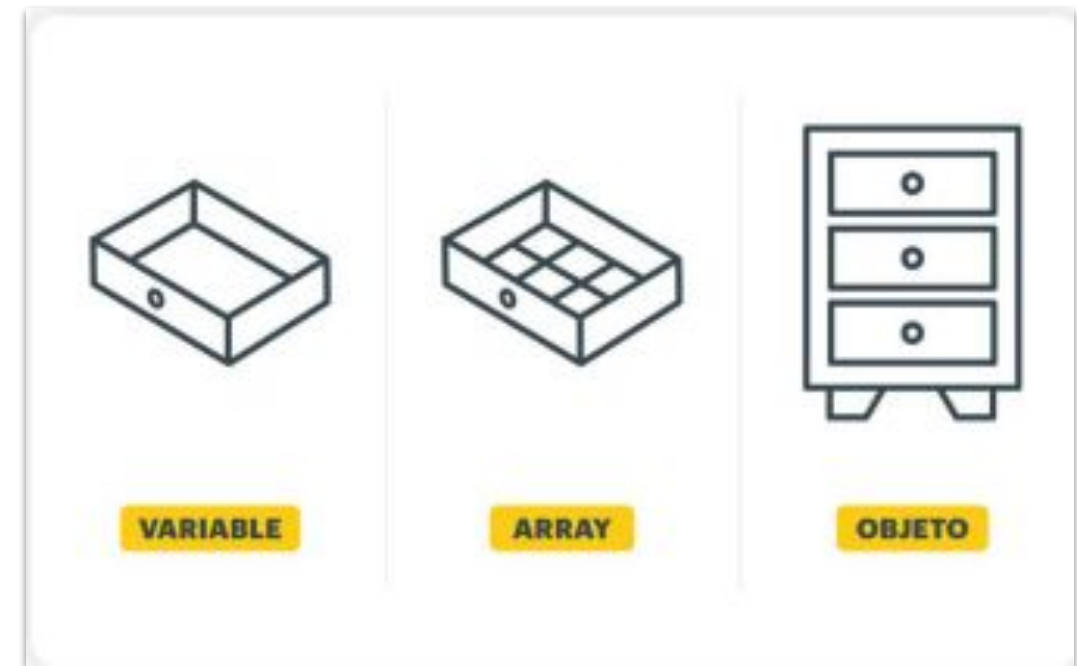
También permite organizar el código en módulos o **clases** reutilizables, y sus respectivas instancias y **objetos**, lo que facilita la creación y mantenimiento de programas complejos.



El Paradigma de Orientación a Objetos

La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener.

Sigue el principio de desarrollo de software utilizado por muchos programadores **DRY (Don't Repeat Yourself)**, para evitar duplicar el código y crear de esta manera programas eficientes.





Clases

Son las plantillas o moldes que utilizaremos para crear objetos. Definen las propiedades y comportamientos que los objetos de esa clase tendrán. Entonces, una clase es una especie prototipo de objetos: define los **atributos** que componen ese tipo de objetos y los **métodos** que pueden emplearse para trabajar con esos objetos.

En su forma más simple, una clase se define por la palabra reservada **class** seguida del nombre de la clase. El **nombre de la clase debe empezar por mayúscula**. Si el nombre es compuesto, cada palabra debe empezar por mayúscula.



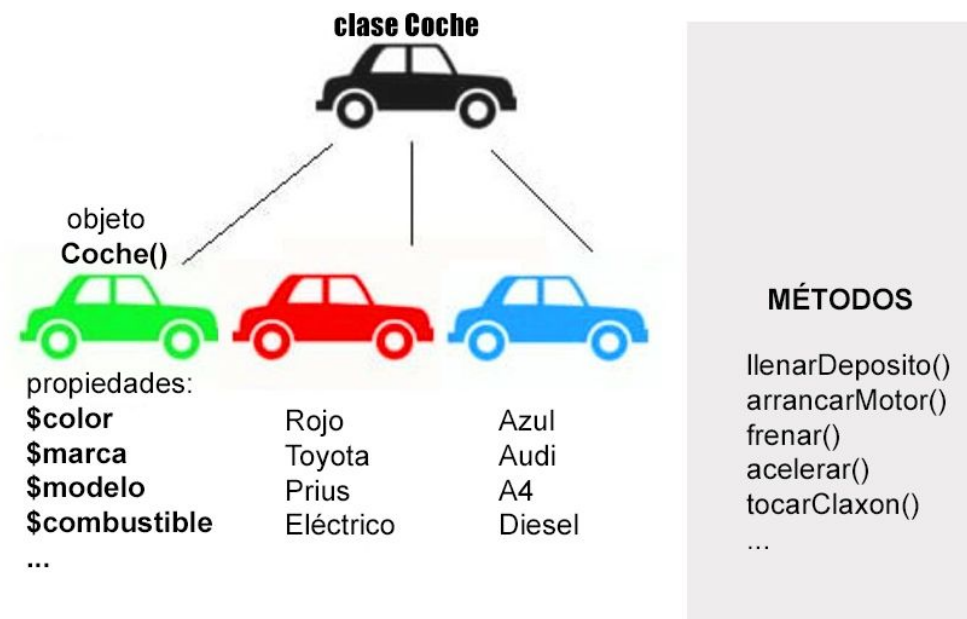
La definición de la clase se pone entre las llaves de apertura y cierre.

```
public class NombreClase {  
    // atributos ;  
    // constructores ;  
    // métodos propios ;  
}
```

Objetos

Son instancias específicas de una clase. Se crean a partir de la clase y tienen su propio estado y comportamiento.

- **Estado:** viene dado por sus **atributos**, que almacenan los datos del objeto. Por ejemplo, el color y marca de un coche.
- **Comportamiento:** viene dado por sus **métodos** que son las operaciones que puede realizar el objeto. Por ejemplo, el coche puede frenar y acelerar.





Estado de un objeto: Atributos

Los **atributos** son características comunes a todos los objetos.

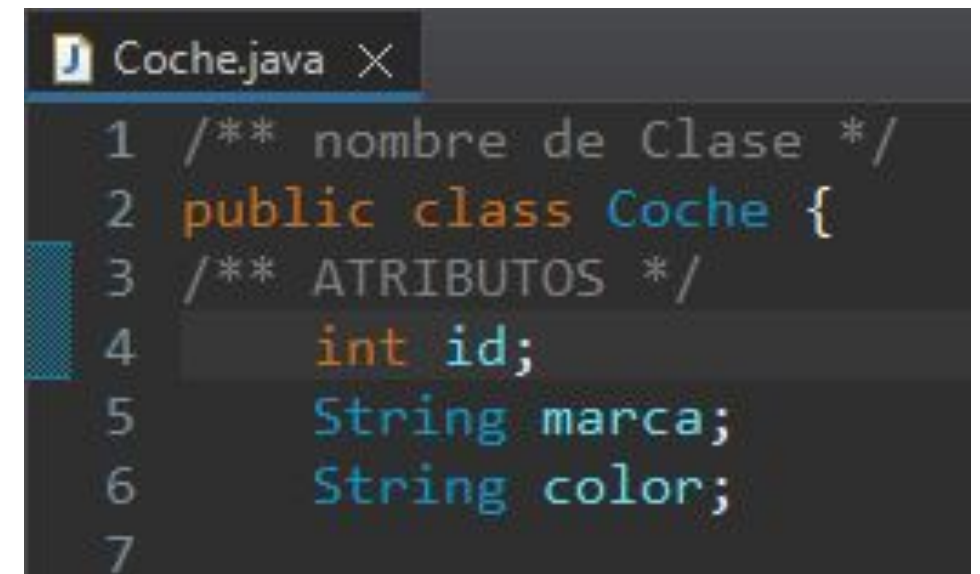
El **estado** o información de un objeto **se almacena en atributos**. Los atributos pueden ser de tipos primitivos de Java o del tipo de otros objetos.

La declaración de un atributo de un objeto tiene la siguiente forma:

<tipo de dato> <nombre del atributo> ;

<tipo>: indica la clase a la que pertenece el atributo definido.

<nombre>: puede ser cualquier identificador válido y denomina el atributo que está siendo declarado.



```
Coche.java X
1 /** nombre de Clase */
2 public class Coche {
3 /** ATRIBUTOS */
4     int id;
5     String marca;
6     String color;
7
```

Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.



LIVE CODING

Ejemplo en vivo

Billetera Virtual:

Vamos a crear una Wallet desde el comienzo! Para esto, crearemos una clase llamada Cuenta que manipulara los siguientes datos:

- 1. Atributos de la clase Cuenta:*
 - Número de cuenta (int)*
 - Titular (String)*
 - Saldo (double)*
- 2. Crear un objeto billetera1 y asignarle valores a estos atributos.*

Tiempo: 25 minutos



Ejercicio N° 1

Cuenta Bancaria



Cuenta Bancaria

Hora de codear: 🙌

La POO nos ayuda a programar pensando y modelando la realidad. Vamos a poner en práctica lo aprendido.

Consigna: 📝

- Crea un proyecto llamado Banco.
- Crea una Clase llamada CuentaBancaria
- Agrega los siguientes atributos y dales un valor:
 - entero número de cuenta
 - doble saldo actual
 - cadena titular

Tiempo 🕒: 25 minutos



○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender la importancia de la POO**
- ✓ **Conocer la implementación de Clases y el estado y comportamiento de los Objetos**



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. Material 1 (Foro)
 - a. *Lectura Módulo 4, Lección 5: páginas 1 - 4*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

Time-out!

🕒 5 min.

