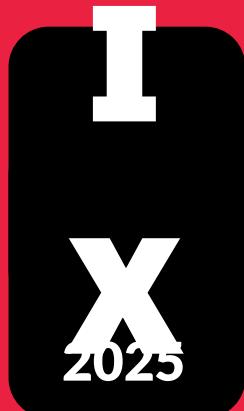


# Orientación a Objetos



**UNRN**

Universidad Nacional  
de Río Negro

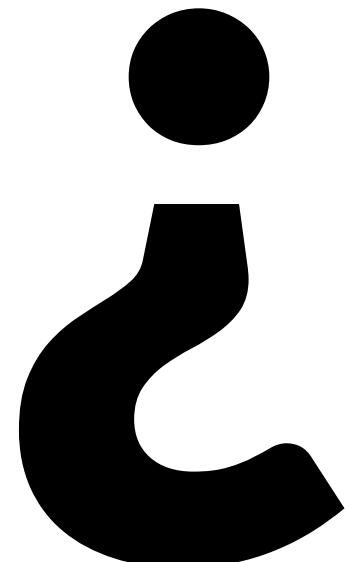
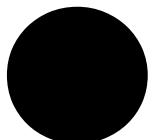


# Para agregar

[https://en.wikipedia.org/wiki/Law\\_of\\_Demeter](https://en.wikipedia.org/wiki/Law_of_Demeter)

# Un pequeño flashback a Programación 1





# **Como se resuelve un problema de forma *'estructurada'***

**datos  
arreglos  
struct**

**Funciones  
Procedimientos**

La información y el código que trabaja sobre esto

**UNRN**

Universidad Nacional  
de Río Negro

# Tenemos los datos, y el código que lo utiliza

```
typedef struct {
    char nombre[20];
    int edad;
} Persona;

void saludar(Persona p)
{
    printf("Hola, soy %s y tengo %d años\n", p.nombre, p.edad);
}
```

**¿Qué pasaría si quiero  
que distintas personas  
puedan saludarse de  
diferentes formas según  
su nacionalidad?**

# Saludar empieza a crecer

```
typedef struct {  
    char nombre[20];  
    char nacionaliad[30];  
    int edad;  
} Persona;
```

```
void saludar(Persona p)
```

**Y la estructura tambien, para alojar más información**

**Cada cambio en los  
datos afecta  
*prácticamente*  
todo el programa**

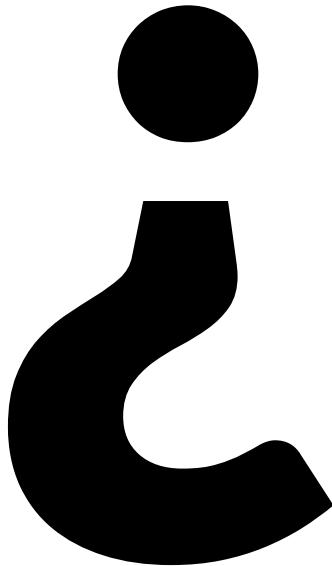
y  
Que pasaría si...

# Hacemos algo así...

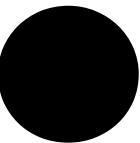
```
typedef struct {
    char nombre[20];
    int edad;
} persona_t;

const persona_t agenda[MAX_CONTACTOS];

void saludar(int posicion)
{
    struct p = agenda[posicion];
    printf("Hola, soy %s y tengo %d años\n", p.nombre, p.edad);
}
```



# Y dinámico



# Se complica BASTANTE

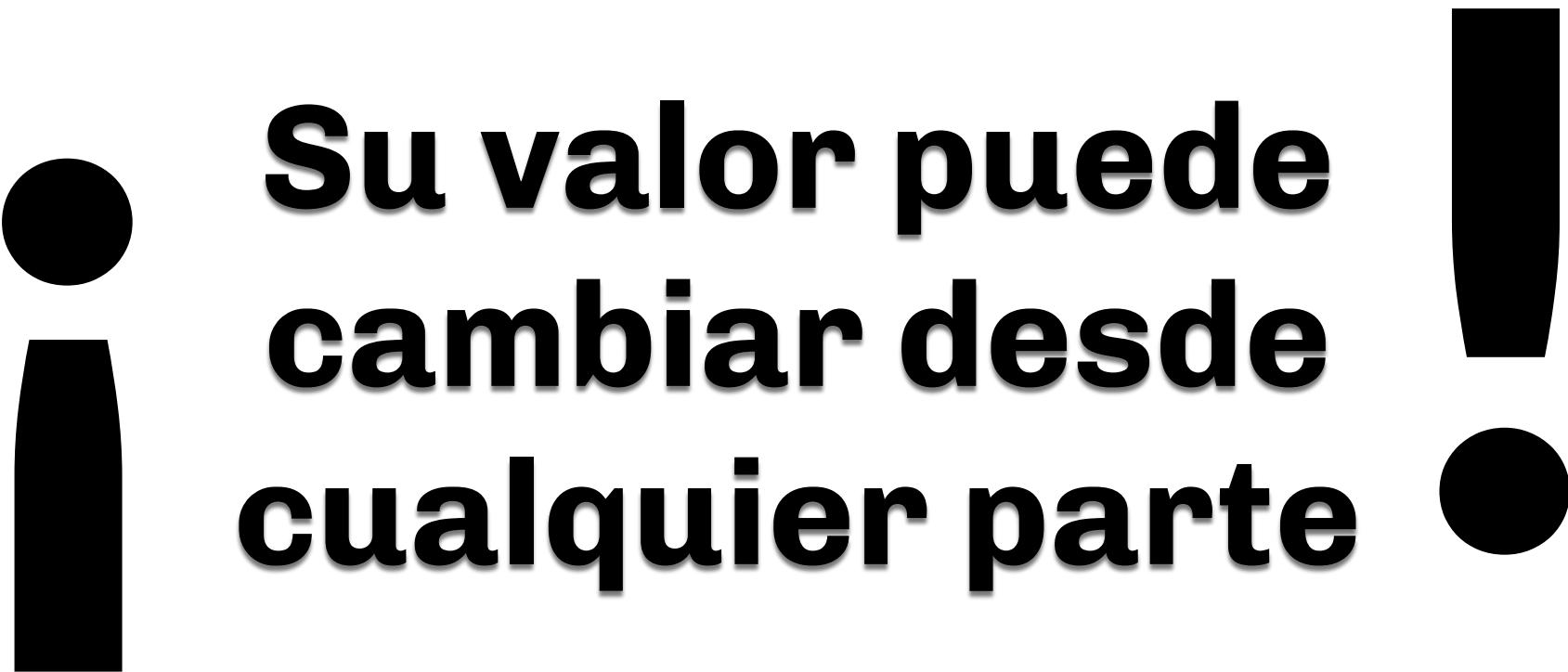
```
typedef struct {
    char nombre[20];
    int edad;
} persona_t;

int cantidad_contactos;
persona_t* agenda;

void saludar(int posicion)
{
    cantidad_contactos = 0;
    agenda = null;
}
```



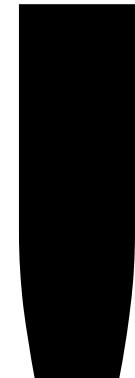
**existen diseños  
que suavizan  
estos problemas**



**Su valor puede  
cambiar desde  
cualquier parte**



**Su valor puede  
cambiar de  
cualquier forma**

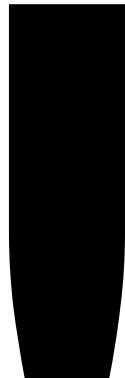


**Solo hay una  
variable global  
para todos**

**No podríamos  
agregar una  
agenda adicional  
sin reescribir**

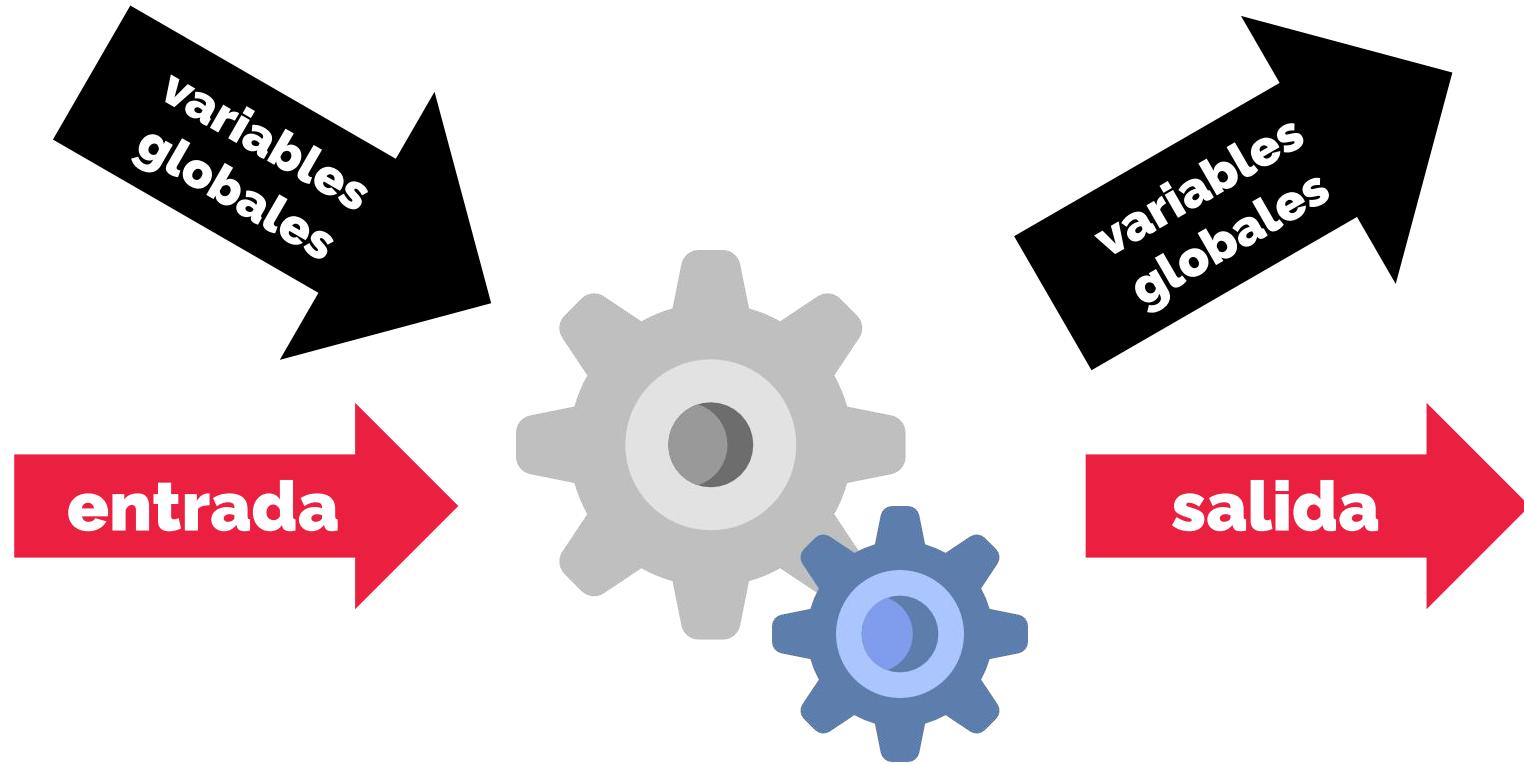


**El debugging es  
mucho más  
complejo**



# **Los tests son mucho más difíciles**

**El estado anterior de la variable  
global es difícil de predecir**



Se pierde la **delimitación** de los argumentos y retorno

**UNRN**

Universidad Nacional  
de Río Negro

**Por eso se  
desaconseja su  
uso**

**(pero, no es una cuestión de que no se use)**

---

Y...

**Que tiene que ver  
esto con Java y  
Programación  
Orientada a  
Objetos**

---

# La Orientación a objetos



**Es una forma de  
asociar código con  
datos**

**Es una forma de  
asociar código con  
datos **bajo control****

**Que *técnicamente***  
**es como usar**  
**variables globales**  
**pero sin las contras**  
**mas grandes**

# ya que

# un objeto es



# juntos

---

# **En términos prácticos**

## **Auto**

### **estado**

fabricante

modelo

año

Toyota Corolla GR 2024

Ford Focus 2022 TdCi

### **comportamiento**

Tocar bocina

Encender

---

# ¿Que se describen en las columnas?

## Auto

### estado

fabricante

modelo

año

### comportamiento

Tocar bocina

Encender

Toyota Corolla GR 2024

Ford Focus 2022 TdCi

---

**la descripción**

**Auto**

**uno en particular**

**estado**

fabricante

modelo

año

**comportamiento**

Tocar bocina

Encender

Toyota Corolla GR 2024

Ford Focus 2022 TdCi

---

**clase**

**Auto**

**objeto**

**estado**

fabricante

modelo

año

**comportamiento**

Tocar bocina

Encender

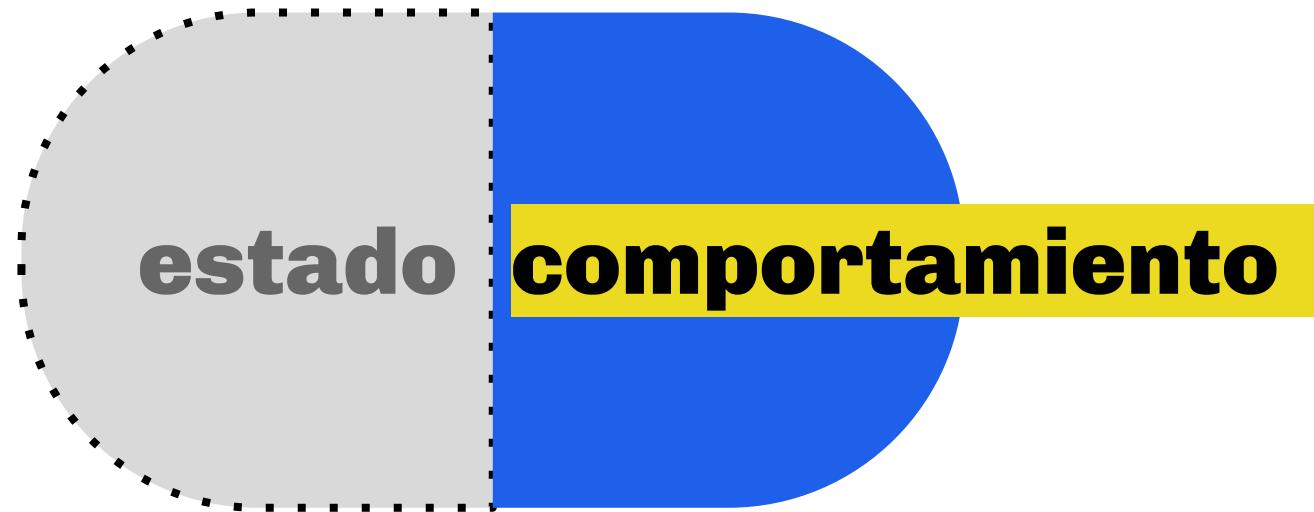
Toyota Corolla GR 2024

Ford Focus 2022 TdCi

Cada objeto es  
independiente  
entre sí

*aunque se comporten igual*

# en definitiva una clase es



## el molde del objeto

**comportamiento**

**estado**

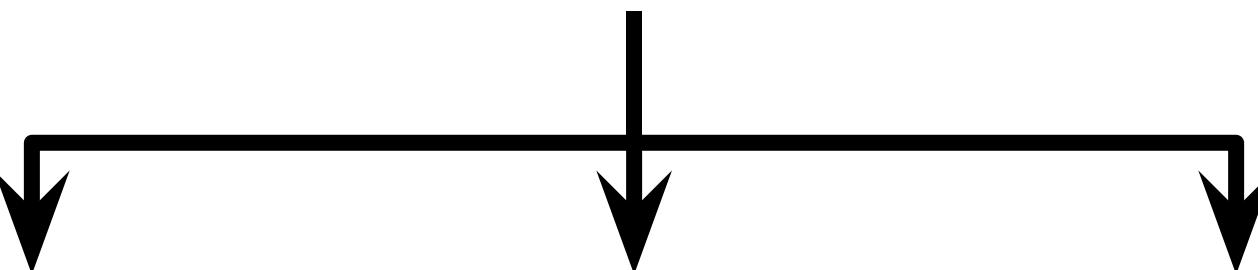
**comportamiento**

**o**

**estado**

**estado**

**estado**

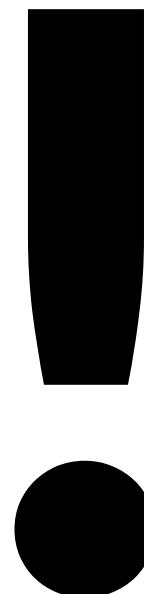


**El comportamiento es el mismo para cada instancia de la clase**

Ese estado es  
compartido **solo**  
por una **instancia\***  
de la clase



# No por todo el programa



Pero cuidado, que  
es **clase** y que es  
**objeto** depende del  
contexto

---

# **Esta es la clave de la Orientación a objetos**

**Controlar el acceso a los datos**

## método

1

**El código de una clase, llamar a uno se denomina *invocar* o pasar un mensaje.**

## Interfaz

2

**El conjunto de métodos de una clase, esta determina el comportamiento del objeto**

## atributo

3

**La información definida en una clase y que recibe valores para transformarse en el estado de un objeto**

## estado

# 4

### Combinación específica de valores en los atributos

# Construcción

5

**Parte del comportamiento que toma una clase y le da valores a los atributos.**

**La *instanciación* de una clase.**

## Destrucción

6

**Parte del comportamiento\* que  
consiste en las acciones para  
liquidar el objeto al final de su  
vida útil.**

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

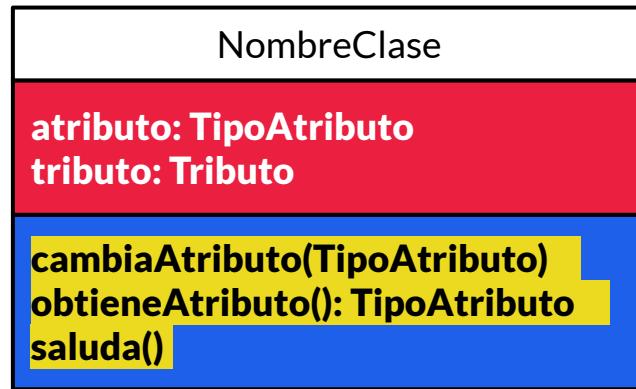
**¿Preguntas?**



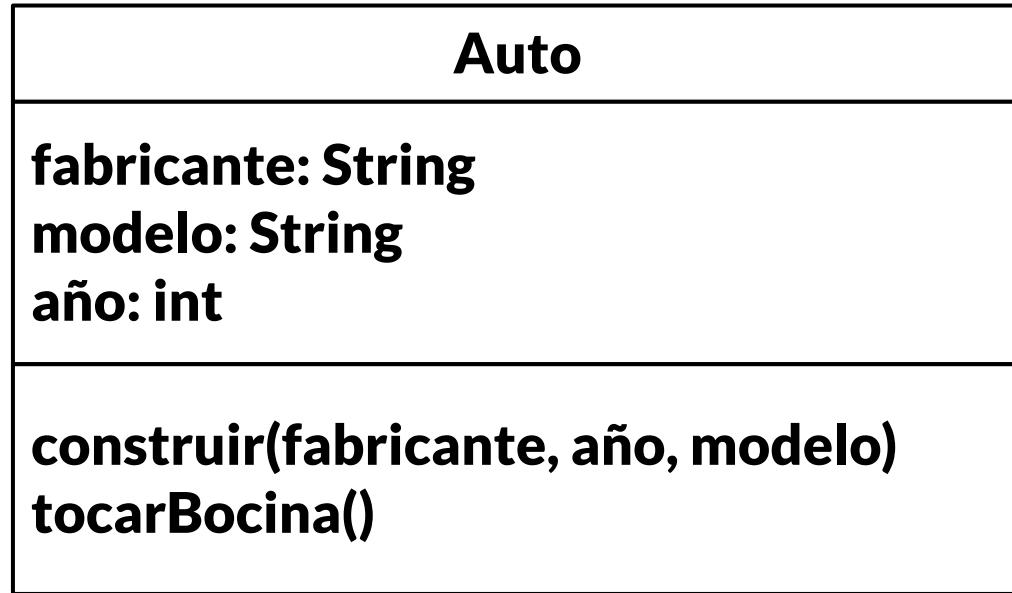
# UML

**unified modelling language**  
***gradual***

# Diagrama de clases



# **El auto como Clase en UML**



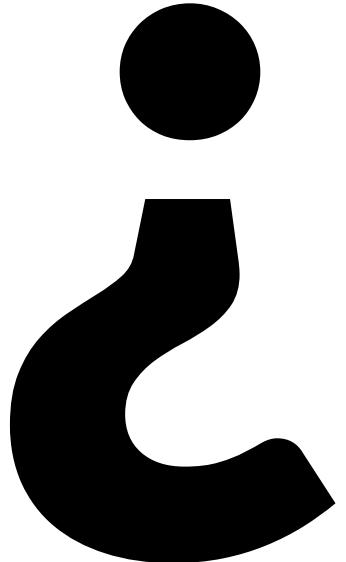
**Antes de seguir  
tenemos que  
hablar sobre un  
tema**

# Otro ejemplo

**Persona**

**nombre: String**  
**apellido: String**

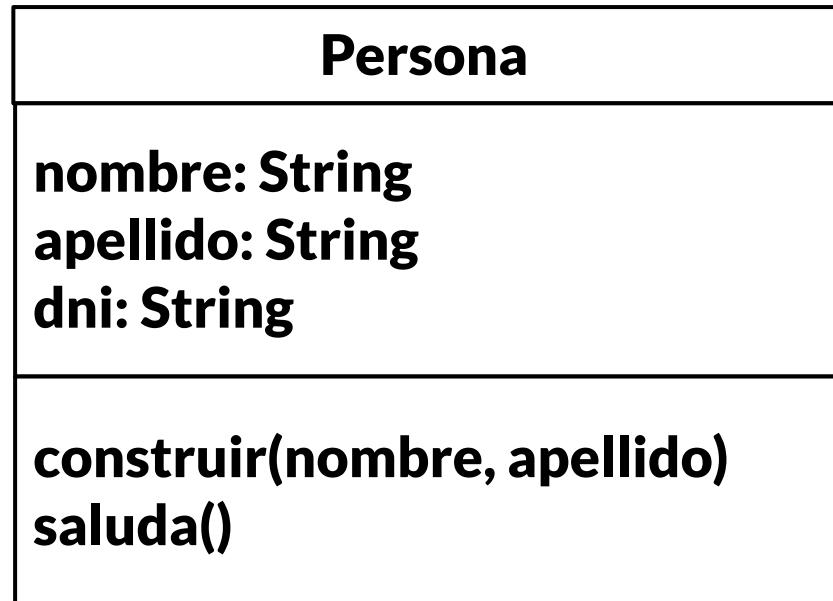
**construir(nombre, apellido)**  
**saluda()**



**Que hace que  
dos personas  
diferentes,  
*diferentes***



# Falta información



7

## Identidad\*

**La combinación de estado que hace único a un objeto**

# Pensemos algunas clases

**Indiquen:**  
**Nombre**  
**Atributos**  
**Métodos**

Un ejemplo

Auto

atributos

cilindrada motor  
cantidad de cambios  
potencia de frenos

conductor  
tonoBocina

métodos

acelerar  
frenar  
sonarBocina

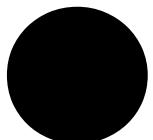
[10 minutos]

---

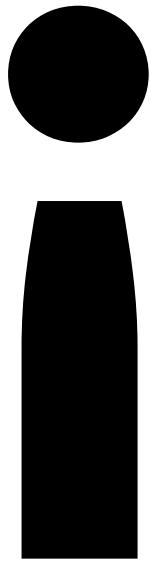
# volvemos en

# 5'

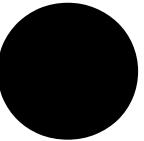
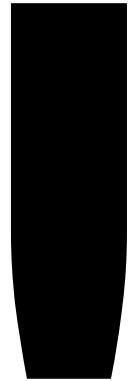
# Las primeras abstracciones de la realidad

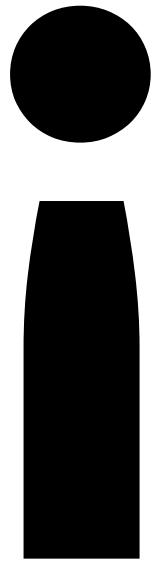


**Quien quiere  
pasar a contar  
en que pensó**

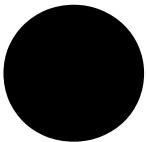


# Fantastico





**El análisis y la  
abstracción  
hecha es crítico**



**La identidad  
puede estar dada  
por cualquier  
parte del estado**

El  
comportamiento  
es ***escencial***



**¿Preguntas?**

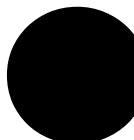


# Siendo un Auto

**Auto**

motor  
cambios  
frenos  
conductor

acelerar()  
frenar()  
tocarBocina()



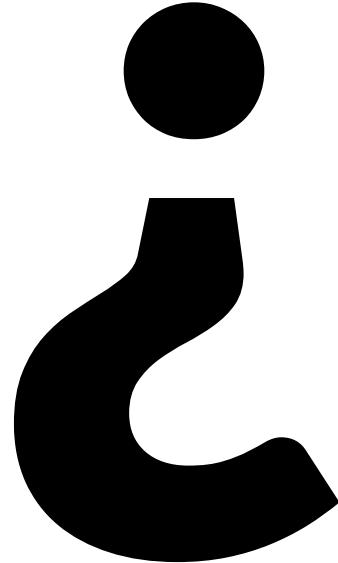
# Cómo sería una Bicicleta

# Bici

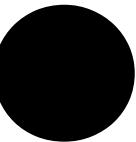
**Bici**

cambios  
conductor

acelerar()  
frenar()  
tocarBocina()



# Y una Moto?



# Moto

**Moto**

motor  
cambios  
frenos  
conductor

acelerar()  
frenar()  
tocarBocina()

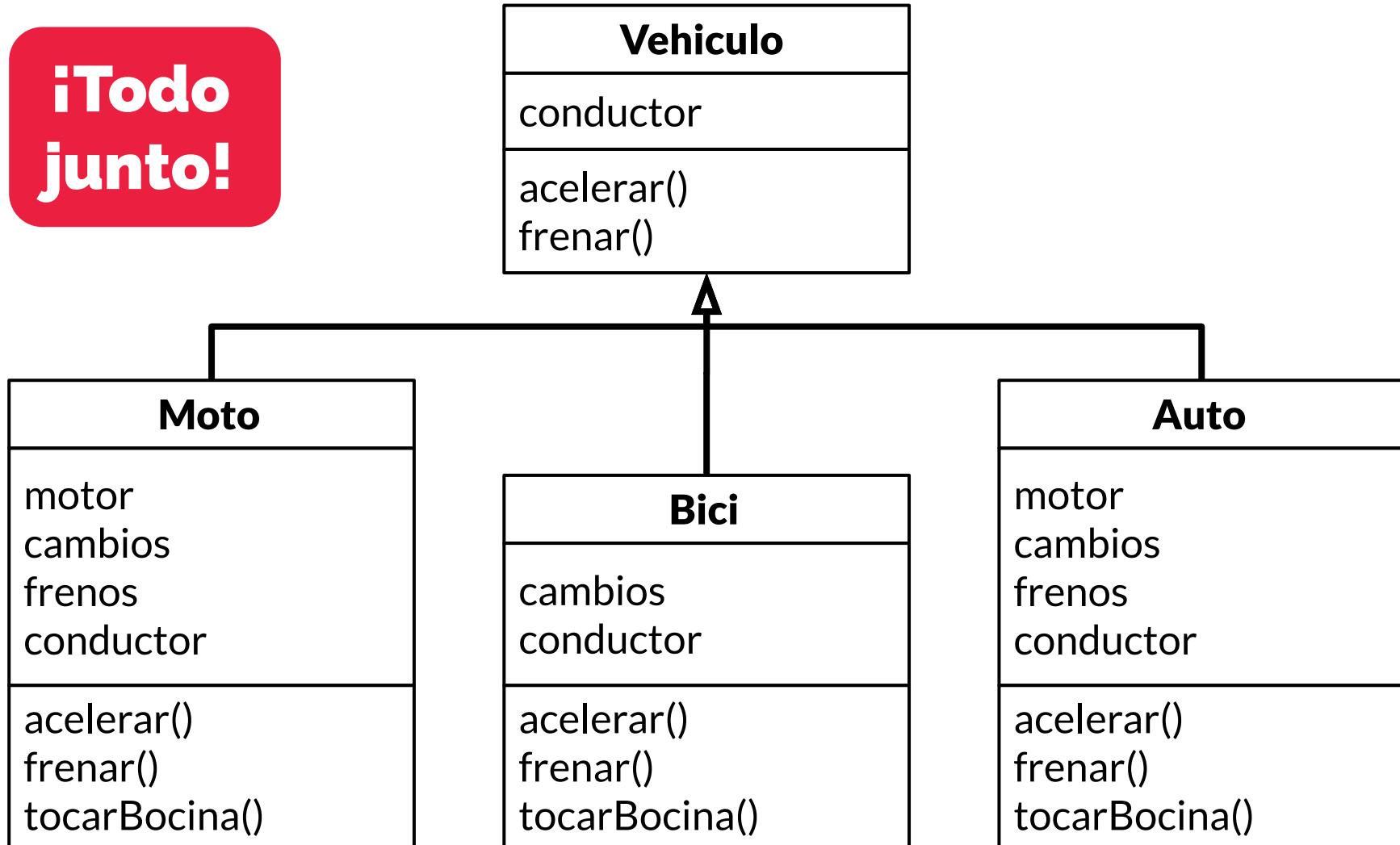
# Vehículo

**Vehiculo**

conductor

acelerar()  
frenar()

# ¡Todo junto!



**El paso de Auto a  
Vehículo es una  
generalización**

# 8

## Generalización

**Pasar de una clase más cercana  
a la realidad a un concepto más  
abstracto**

# **El paso de Vehículo a Bici es una especialización**

## Especialización

9

**Pasar de una clase más  
abstracta a una cercana a la  
realidad**

## Auto

motor  
cambios  
frenos  
conductor

acelerar()  
frenar()  
tocarBocina()



## Camión

motor  
cambios  
frenos  
conductor  
carga

acelerar()  
frenar()  
tocarBocina()  
cargar()

**Ambos  
conceptos  
forman**

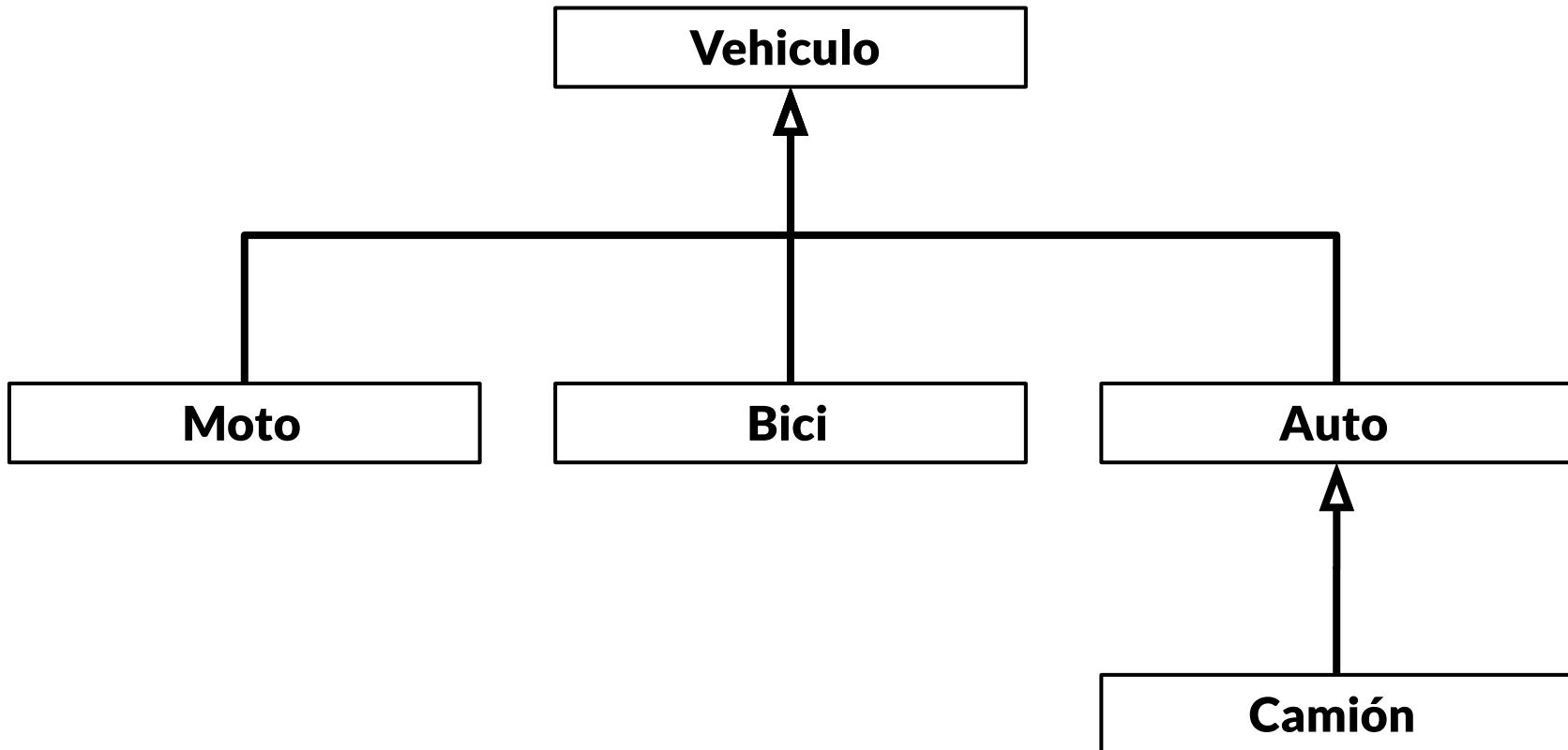
1

## Herencia

**Es la relación entre clases de un mismo tipo.**

- La clase que hereda especializa a la madre.
- La clase madre es la generalización de la hija.

0



# Herencia

El comportamiento y atributos\* definidos por el parent, son utilizables por los hijos

# Herencia II

[19:11] Santiago Tosoni

Los autos son vehículos, pero no todos los  
vehículos son autos

---

# Auto [Versión 1]

atributos

motor

cambios

frenos

conductor

métodos

acelerar

frenar

¡Queda mucho afuera!

## Concepto clave

1

# Abstracción

es el proceso de identificar las características esenciales de un objeto y representarlas de manera simplificada. Dejando de lado detalles que no tengan que ver con el objetivo.

1

Identificar lo  
relevante a la tarea  
es crítico



**Encontrar la  
abstracción  
correcta requiere  
practica**



**¿Preguntas?**



# ¿De qué *tipo* es el motor?

**Auto**

motor  
cambios  
frenos  
conductor

acelerar()  
frenar()  
tocarBocina()

1

## Composición

**es una relación de tipo  
"todo/parte", en donde la parte  
no puede existir por su cuenta.**

2

auto-motor

# ¿De qué *tipo* es el conductor?

**Auto**

motor  
cambios  
frenos  
conductor

acelerar()  
frenar()  
tocarBocina()

1

## Agregación

**es una relación en la que la clase contenida en otra, puede existir de manera separada.**

3

auto-conductor

# Notación UML para relaciones

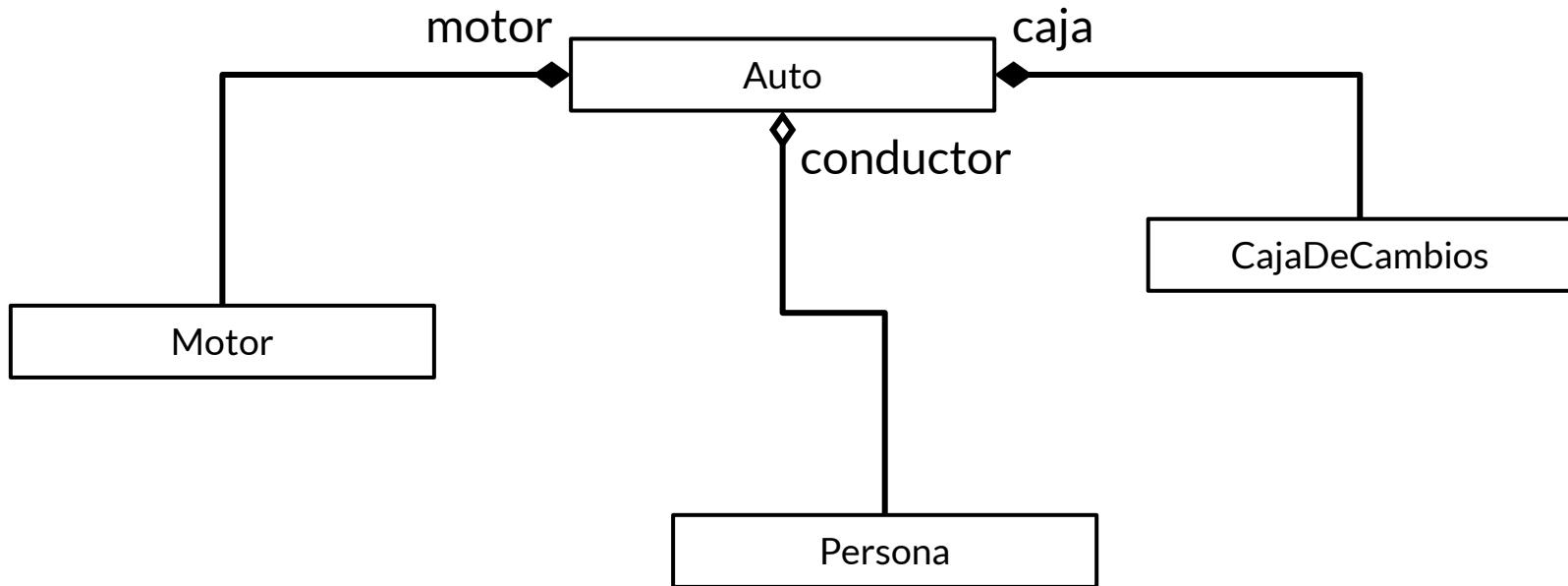
Composición



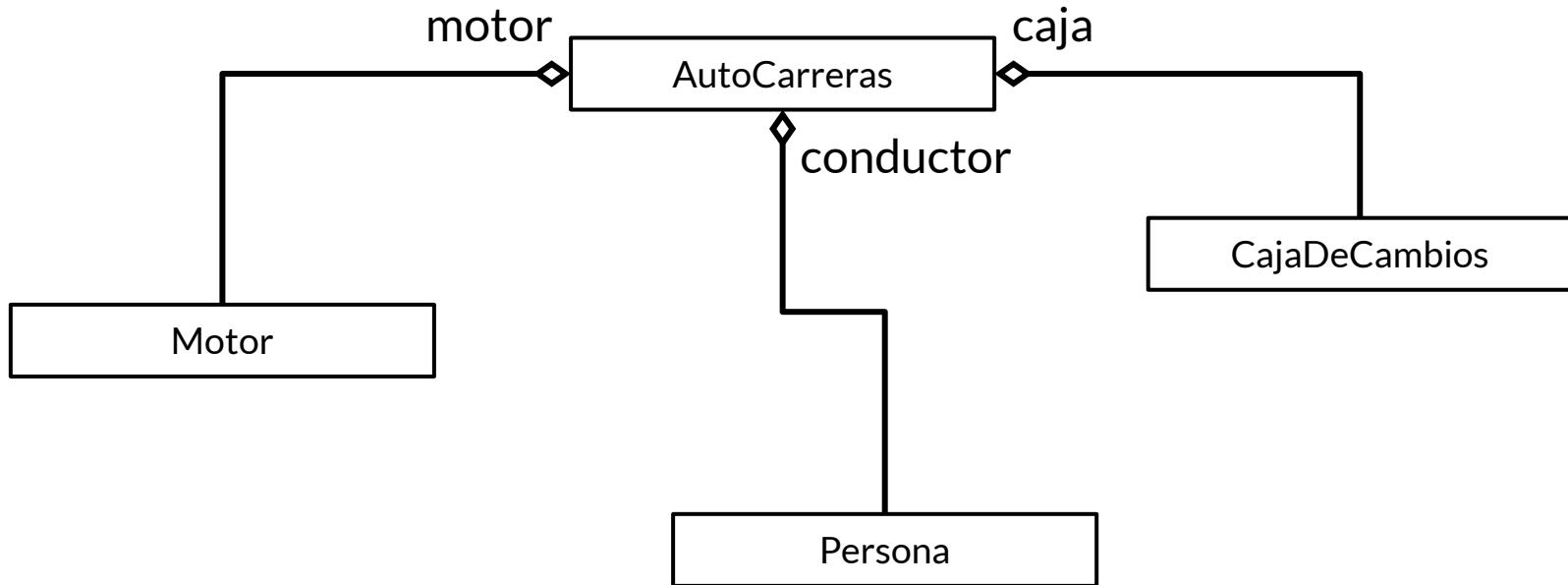
Agregación



# UML para Auto



# UML para Auto de carreras



A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

**¿Preguntas?**



Para seguir  
ejercitando

**La POO es una  
forma de  
representar la  
realidad**

**Es muy importante  
tener en cuenta el  
contexto**

---

# No es lo mismo

# **Una moto para un taller**

# Que una moto para su usuario

# Tren

- locomotora
- vagones
- ruta
- conductor

**¿Cuántos hay de cada una?**

# Tren

- 1 o más locomotoras
- 0 o más vagones
- 1 ruta
- 1 conductor

# **Esto establece la**

Establece la

# cardinalidad

Cuantas veces aparece un objeto dentro de otro

1

## Cardinalidad

Cuántas instancias de un objeto  
son contenidas por otro.  
Si no se indica, es una.

1

0..1

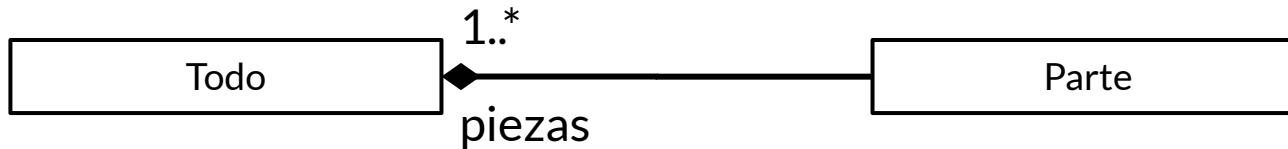
0..\*

1..\*

4

# Notación UML con cardinalidad

Composición



Agregación



# Definida como

## minimo .. maximo

0 .. 1

0 .. \*

1 .. \*

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

**¿Preguntas?**



# TP5

## Análisis de Clases y Objetos

—

Es formato taller,  
tiene limitaciones  
de tiempo

*adicionales*

**unrn.edu.ar**

**UNRN**

Universidad Nacional  
de Río Negro



| unrnionegro