

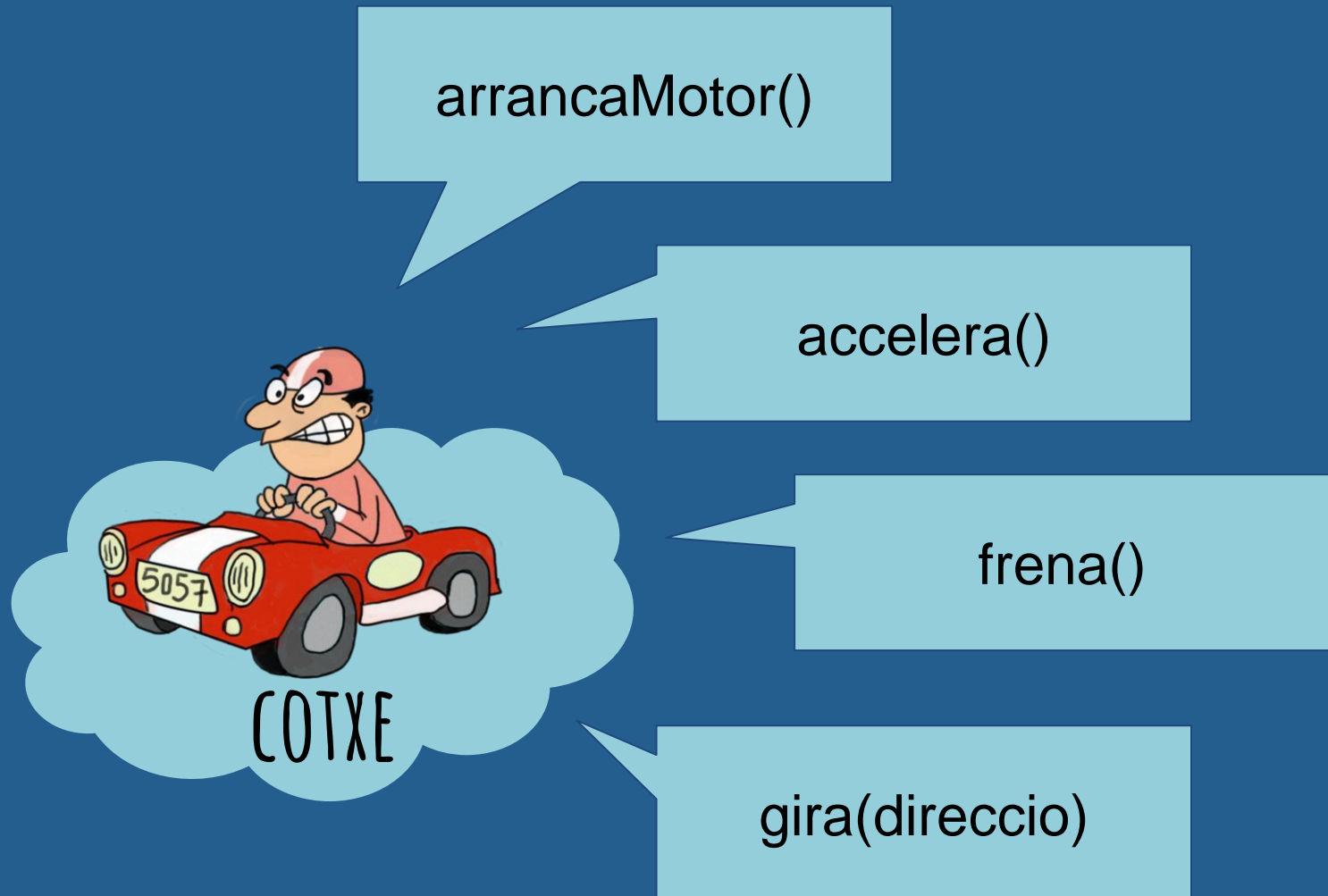
PROGRAMACIÓ ORIENTADA A OBJECTES EN JAVA

CLASSES I OBJECTES

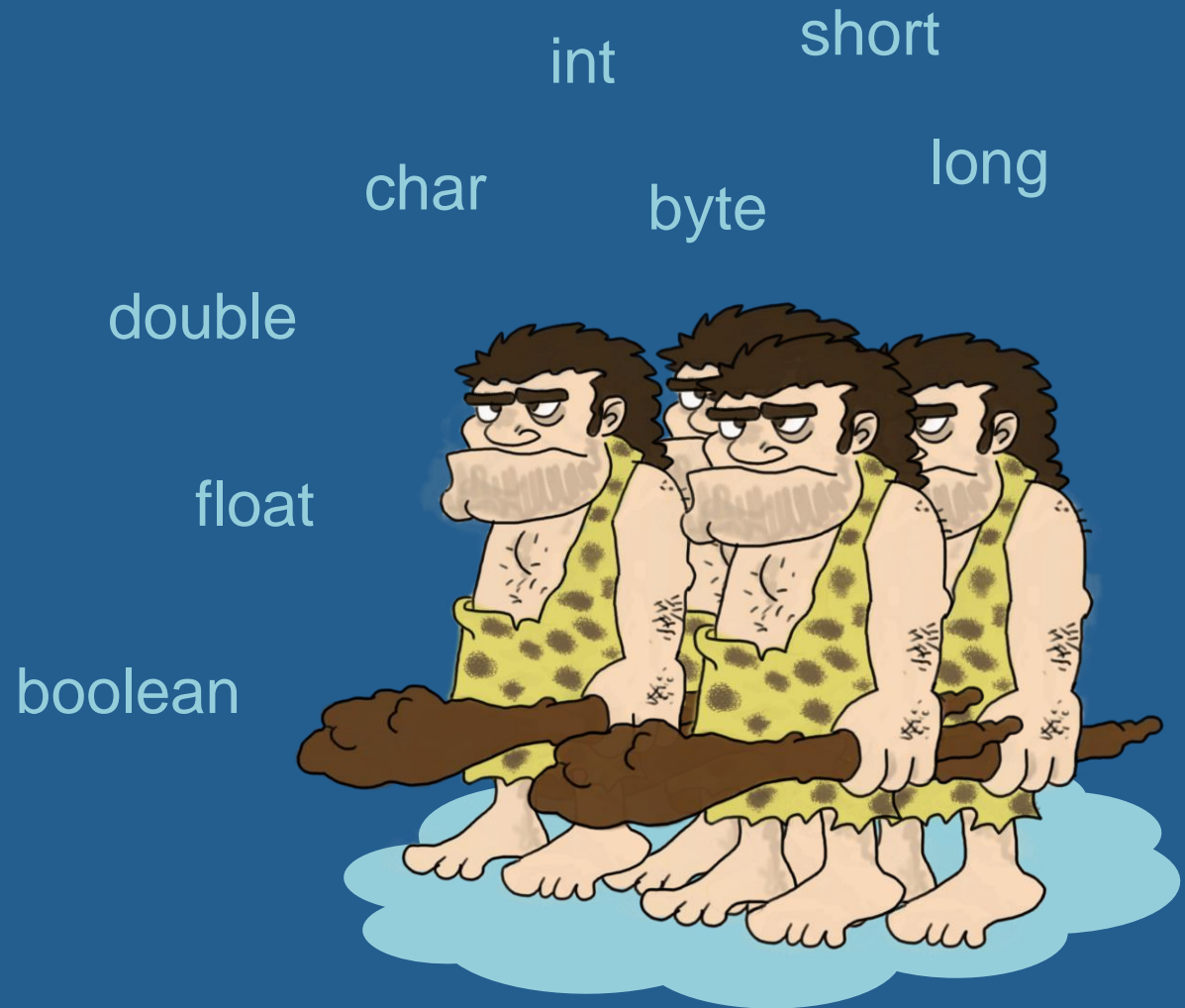


M03 PROGRAMACIÓ
UF4. NF1. CLASSES I
OBJECTES

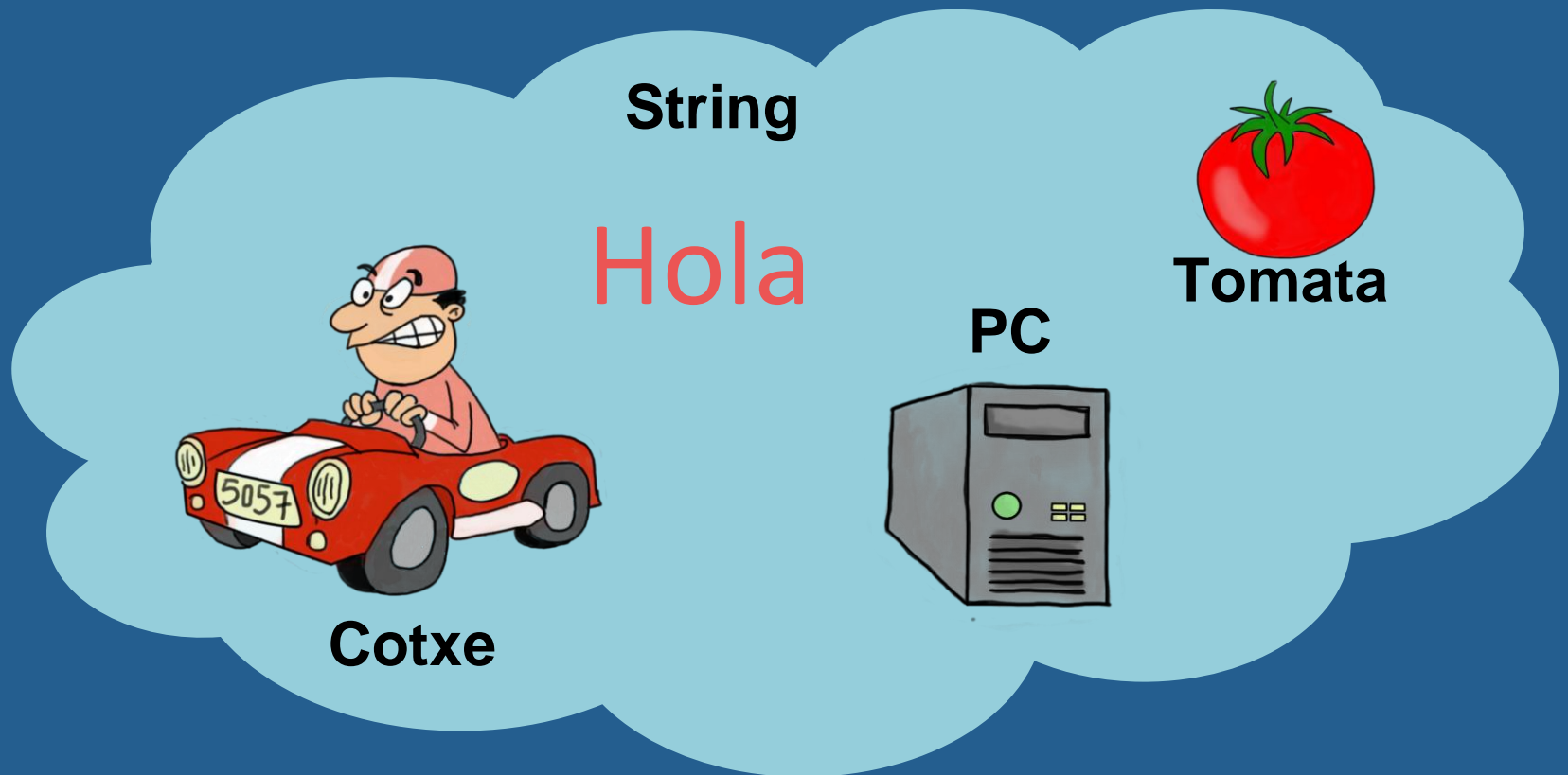
Java és un llenguatge de programació **orientat a objectes**



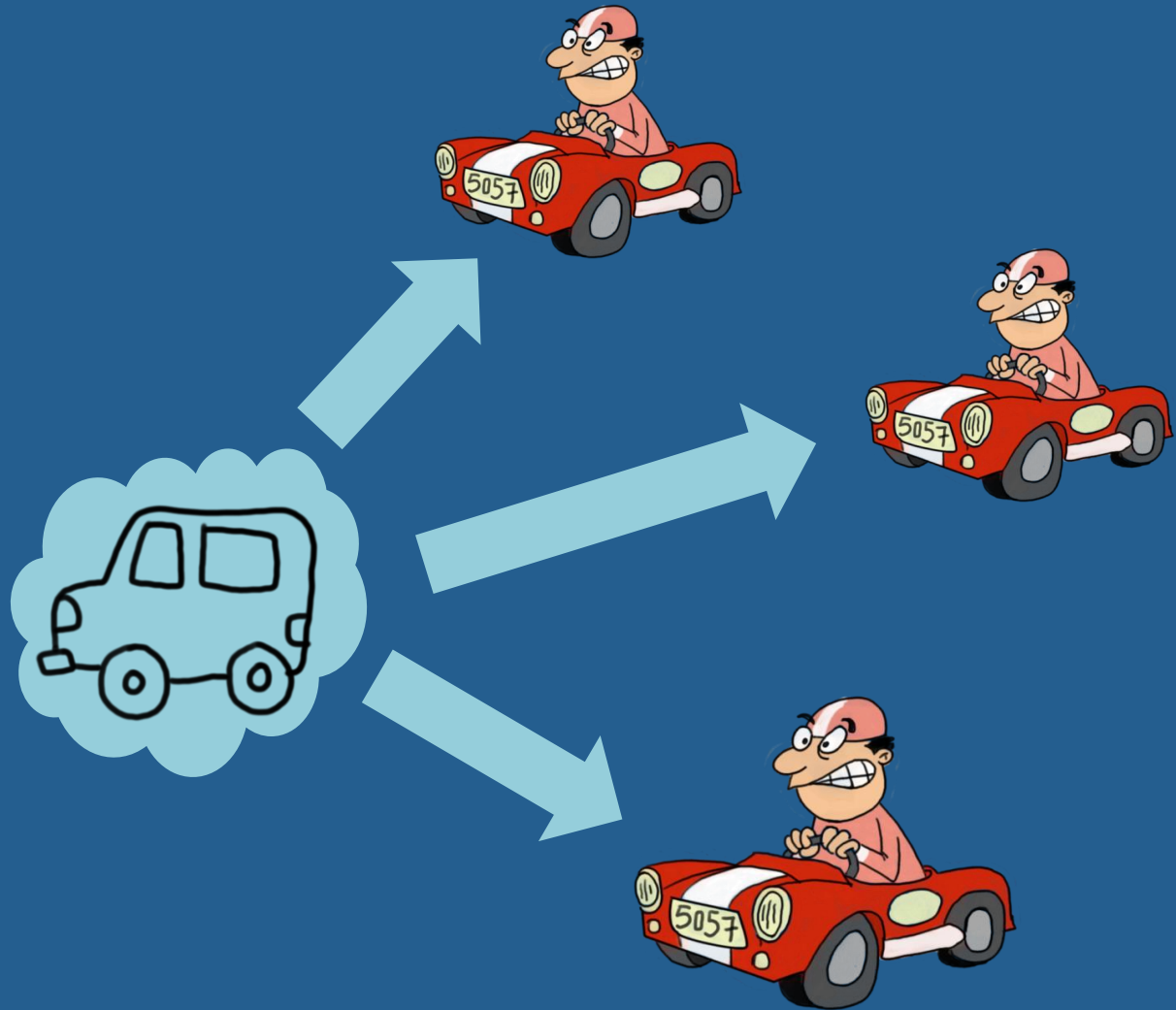
Els tipus primitius **NO**
són objectes



Tota la resta són
objectes

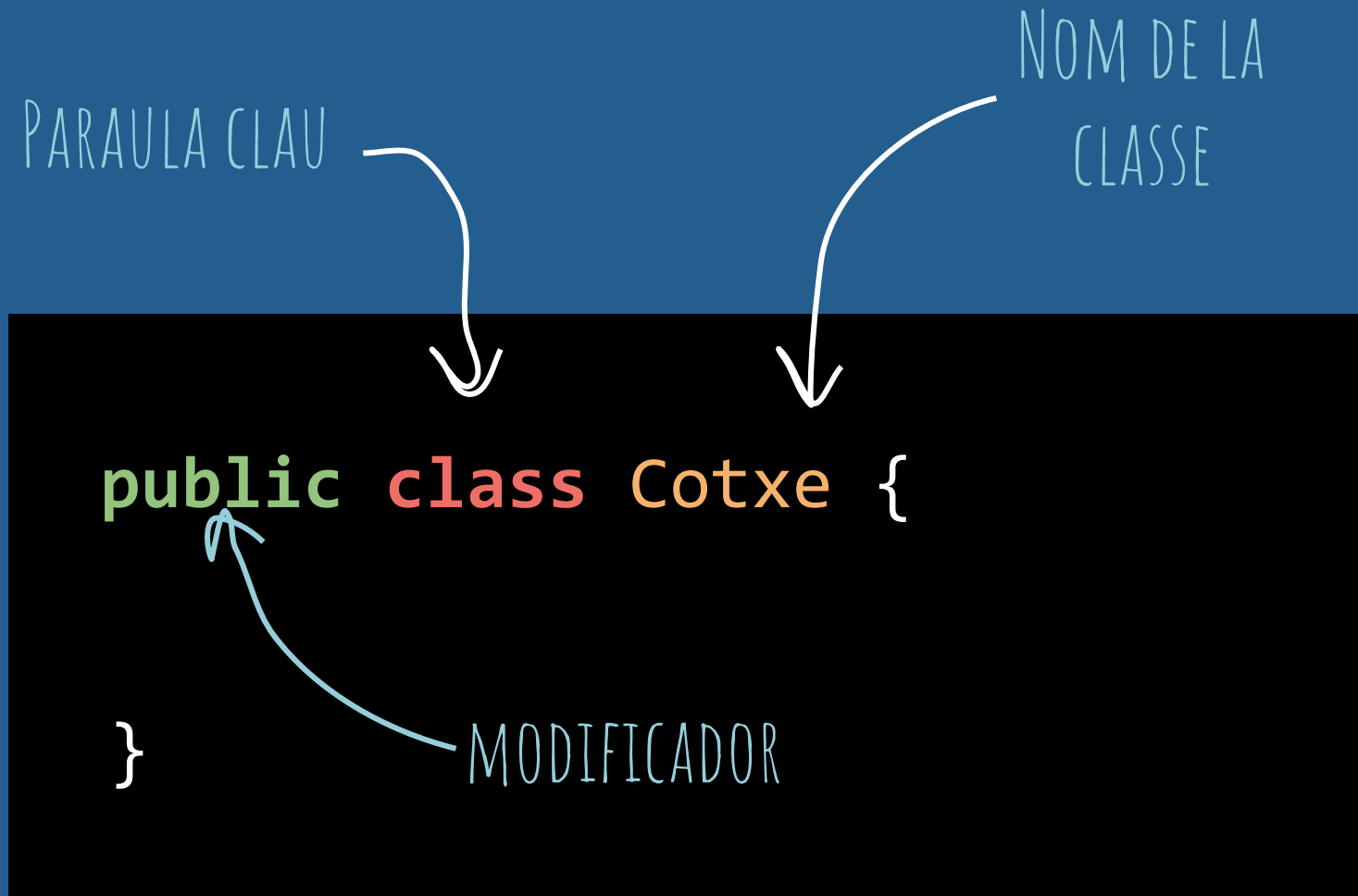


Els objectes es creen a partir de **classes**



DEFINICIÓ DE CLASSES EN JAVA

Les classes es defineixen amb **class**



Dins de les classes s'hi
defineixen **propietats** ...

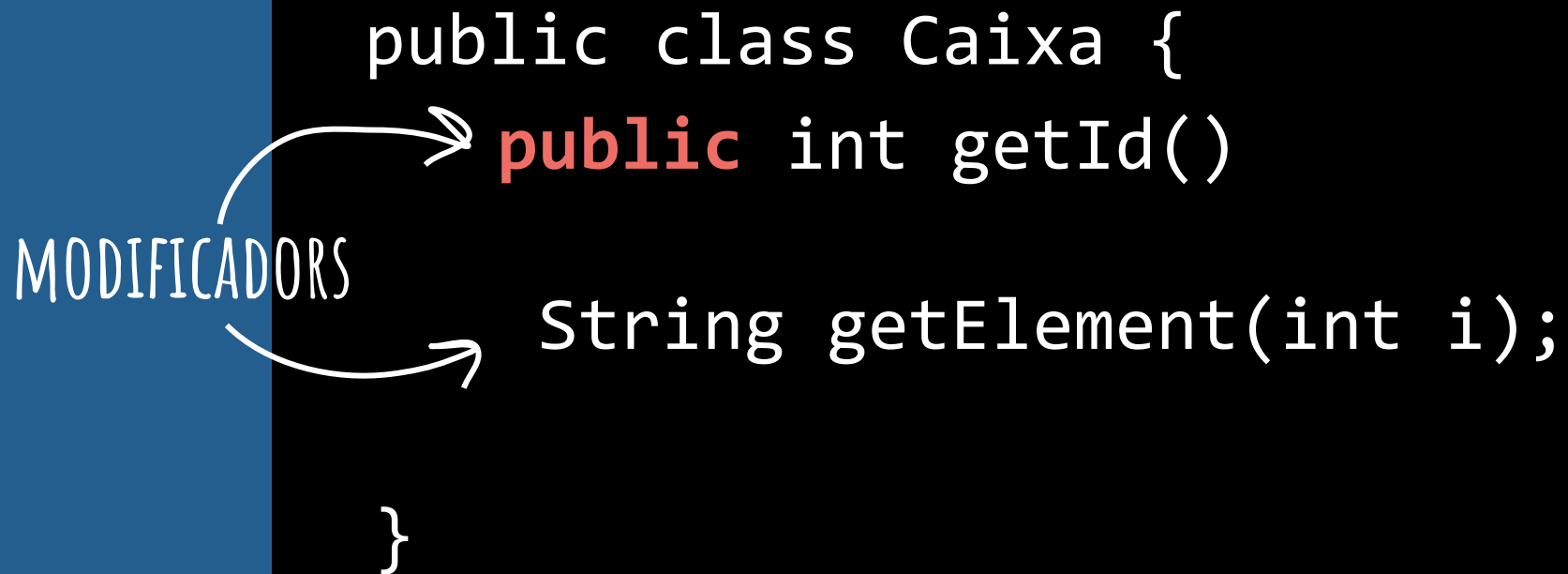
```
public class Cotxe {  
    private int id;  
    public String color;  
}
```

MODIFICADORS

DE QUAALSEVOL
TIPUS

The diagram illustrates the components of a class property definition in Java. The word 'MODIFICADORS' (Modifiers) is written in white, with two curved arrows pointing to the 'private' and 'public' keywords in the code. The phrase 'DE QUAALSEVOL TIPUS' (Of any type) is also written in white, with a curved arrow pointing to the 'String' type in the second property declaration.

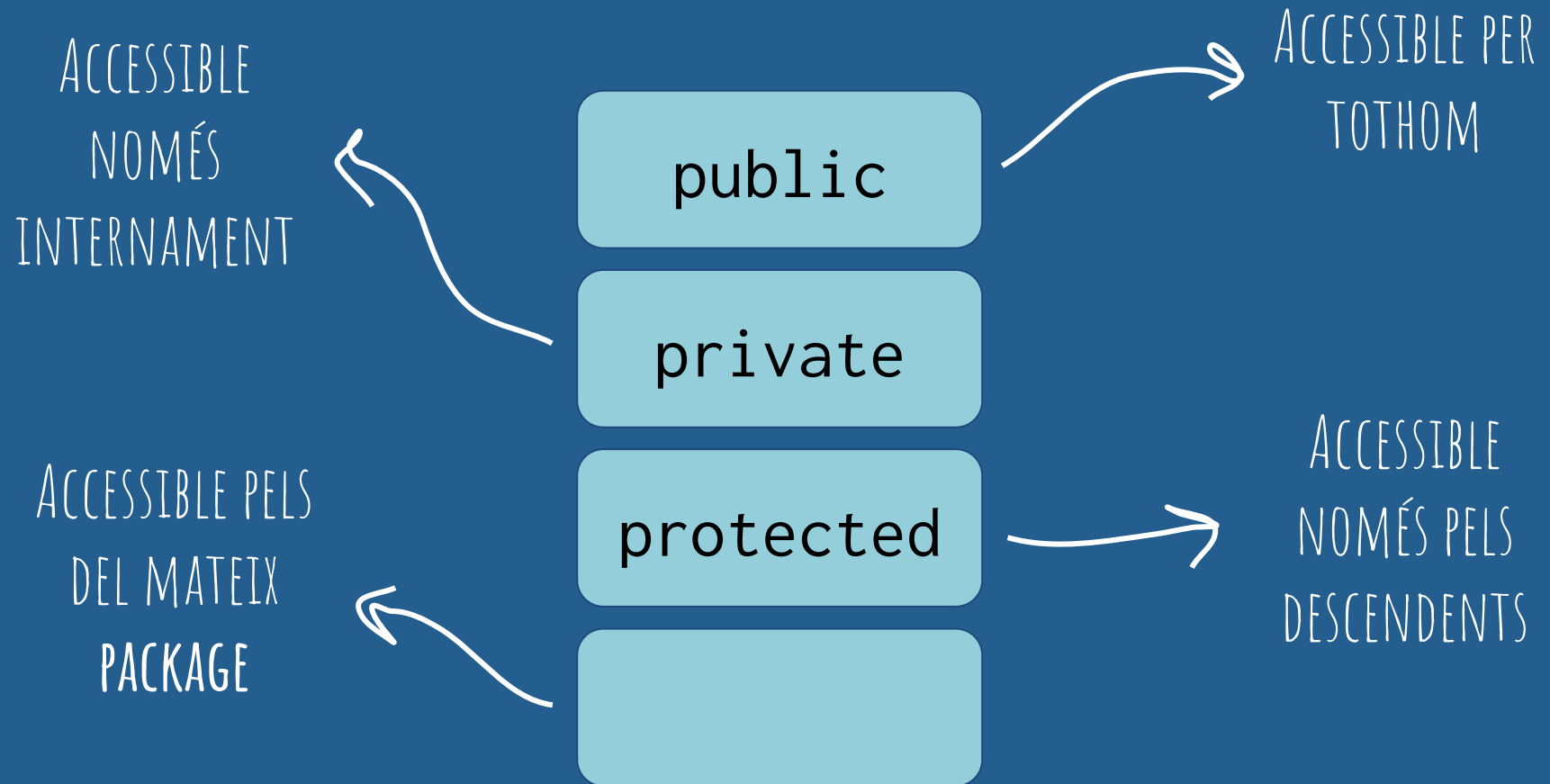
Les classes tenen **mètodes** ...



```
public class Caixa {  
    public int getId()  
    String getElement(int i);  
}
```

MODIFICADORS

Amb els modificadors s'indica **l'accés**
a mètodes i propietats



En Java són habituals els
getters i **setters** de les
propietats




ELS GETTERS RECUPEREN
INFORMACIÓ

ELS SETTERS POSEN
INFORMACIÓ




S'AFEGEIX EL NOM DE LA
PROPIETAT AL DARRERE

```
public class Cotxe {  
    private String color;  
  
    public String getColor() {  
        return color;  
    }  
    public void setColor(String c) {  
        color = c;  
    }  
}
```

A diagram consisting of two white arrows on a black background. The first arrow starts at the text 'S'AFEGEIX EL NOM DE LA PROPIETAT AL DARRERE' and points to the 'color' parameter in the 'setColor' method signature. The second arrow starts at the same text and points to the 'color' variable in the 'return color;' statement within the 'getColor' method.

ELS GETTERS EN BOOLEANS ES
COMENCEN AMB **is**

```
public class Cotxe {  
    private boolean vehicle;  
  
    public boolean isVehicle() {  
        return vehicle;  
    }  
}
```



PER QUÈ NO ACCEDIR
DIRECTAMENT A LES
PROPIETATS EN COMPTES
DE FER GETTERS I
SETTERS?



Els getters i setters permeten **tenir control** sobre els valors

```
private int any;  
public void setAny(int nou) {  
    if (nou < 2016) {  
        any = nou;  
    }  
}
```

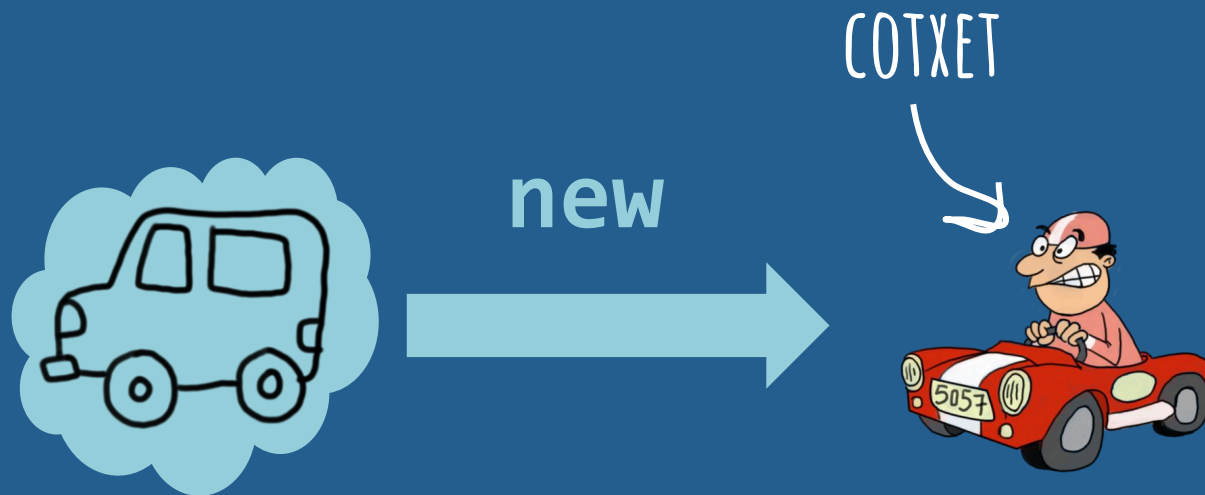
←
L'ACCÉS DIRECTE A LA
PROPIETAT NO
PERMET FER
COMPROVACIONS

Accedir directament a les propietats
trencaria amb l'encapsulació



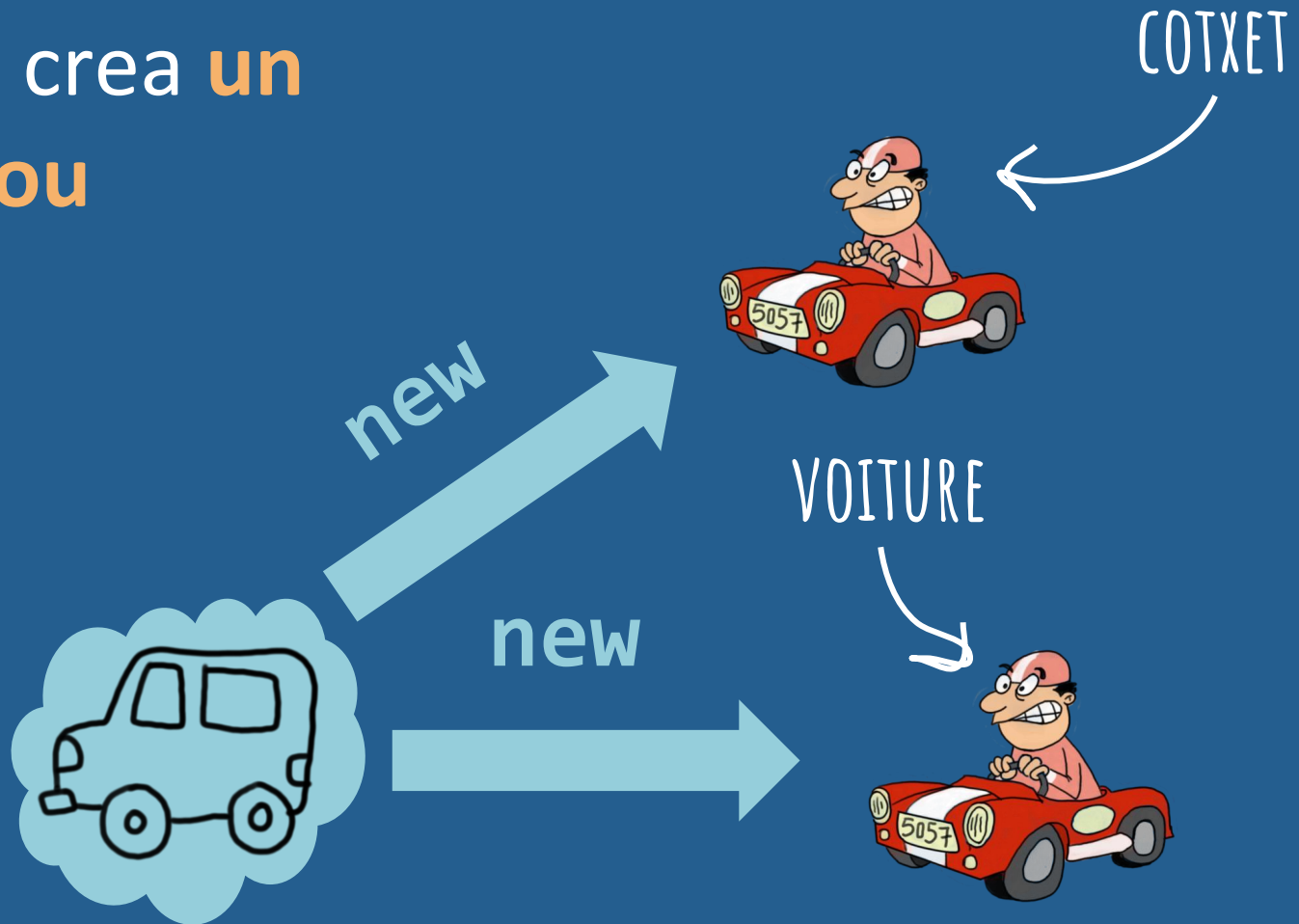
CREACIÓ D'OBJECTES

Els objectes es creen
amb **new**



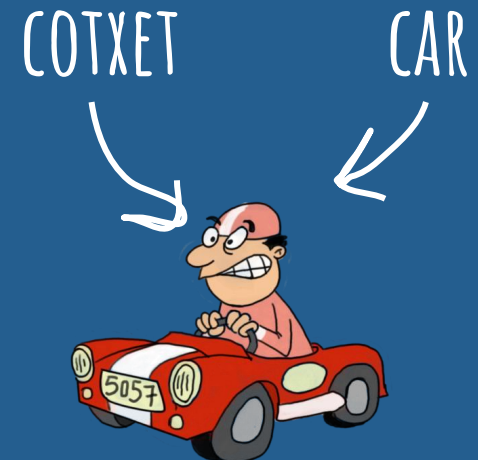
```
Cotxe cotxet = new Cotxe();
```

Cada new crea **un**
objecte nou



```
Cotxe cotxet = new Cotxe();  
Cotxe voiture = new Cotxe();
```

L'assignació **NO** crea
objectes nous



EL MATEIX
OBJECTE TÉ DOS
NOMS!

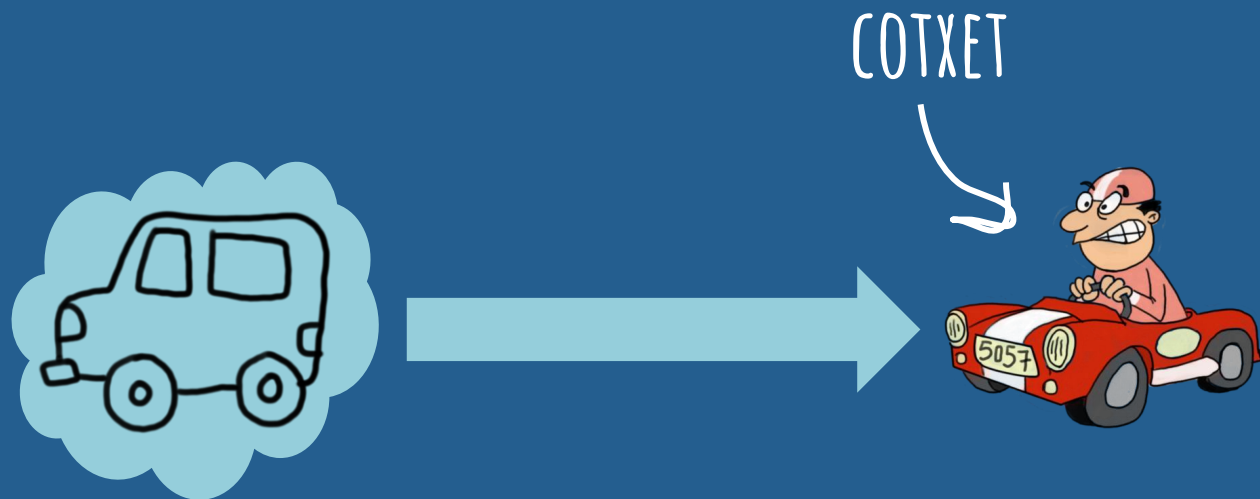
```
Cotxe cotxet = new Cotxe();  
Cotxe car = cotxet
```

Inicialment les variables estan
buides



```
Cotxe cotxet;
```

instanceOf permet saber el tipus
d'una variable en temps d'execució



```
if (cotxet instance of Cotxe) {  
    System.out.println("Cotxe");  
}
```

Es poden cridar als membres d'una classe amb el '.'

```
class Cotxe {  
    public void arrancaMotor() {  
        ...  
    };  
}
```

```
Cotxe cotxet = new Cotxe();  
cotxet.arrancaMotor();
```

Els objectes poden fer referència a ells mateixos amb **this**

COTXET



COTXE

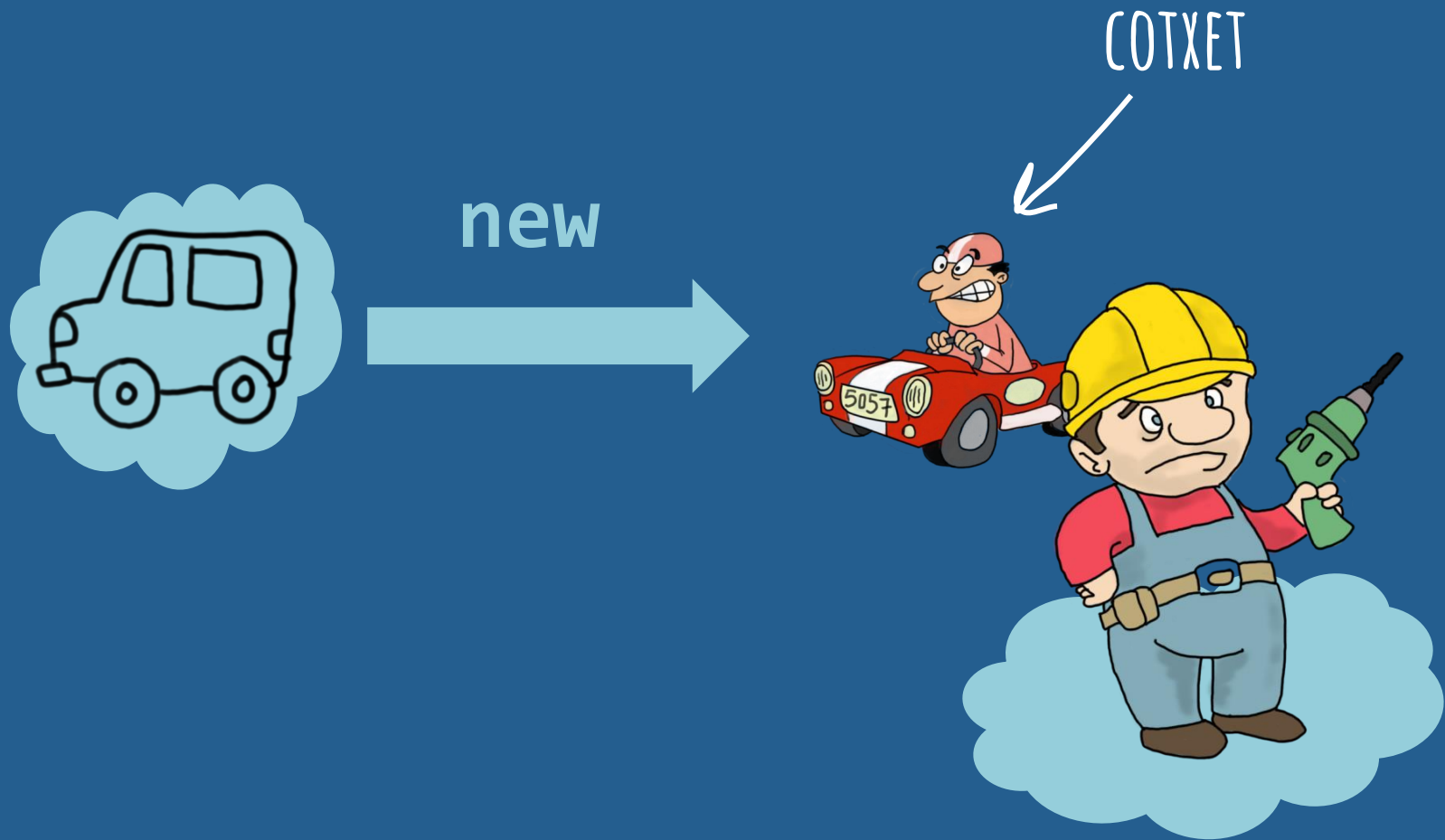
```
this.arrancaMotor();
```

Brum,
Brum,
Brum

CONSTRUCTORS I

DESTRUCTORS

Al crear un objecte s'executa
el **constructor**



Els constructors tenen el
nom de la classe



```
class Cotxe {  
  
    public Cotxe() {  
  
    }  
  
}
```

NO PODEN
RETORNAR RES

Si no se'n defineix cap Java en crea
un **per defecte**



```
class Cotxe {  
  
}
```

```
public Cotxe() {  
  
}
```



Hi poden haver tants
constructors com calgui

SEMPRE QUE NO TINGUIN EL
MATEIX NÚMERO PARAMETRES
I AQUESTS DEL MATEIX TIPUS



```
class Cotxe {
```

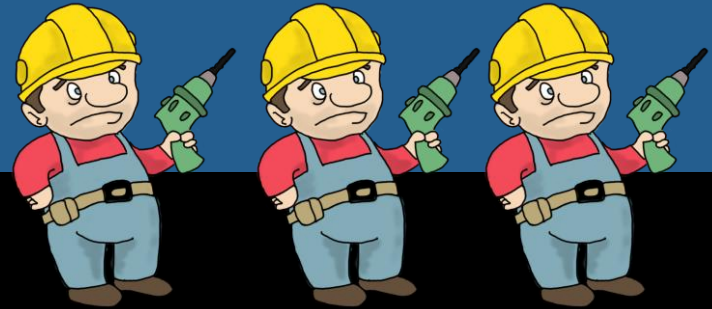
```
    public Cotxe() { ... };
```

```
    public Cotxe(int rodes);
```

```
    public Cotxe(String marca);
```

```
    public Cotxe(int x, int y);
```

```
}
```



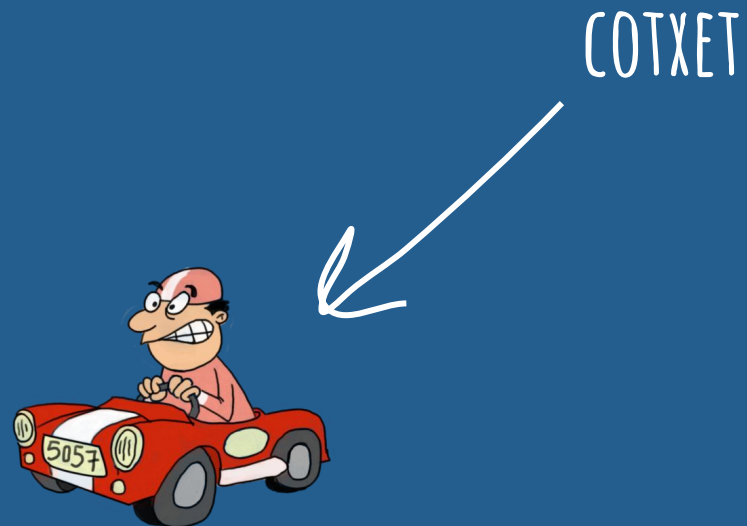
NO LI CALEN
DESTRUCTORS PERQUÈ TÉ
UN RECOLLIDOR DE
BASURA



ES POT FER FINALIZE()



Els objectes es destrueixen quan
no s'hi fa referència



```
Cotxe cotxet = new Cotxe();
```


Els objectes es destrueixen quan no s'hi fa referència



COTXET



NULL

```
Cotxe cotxet = new Cotxe();  
cotxet = null;
```

Els objectes es destrueixen quan no s'hi fa referència



COTXET

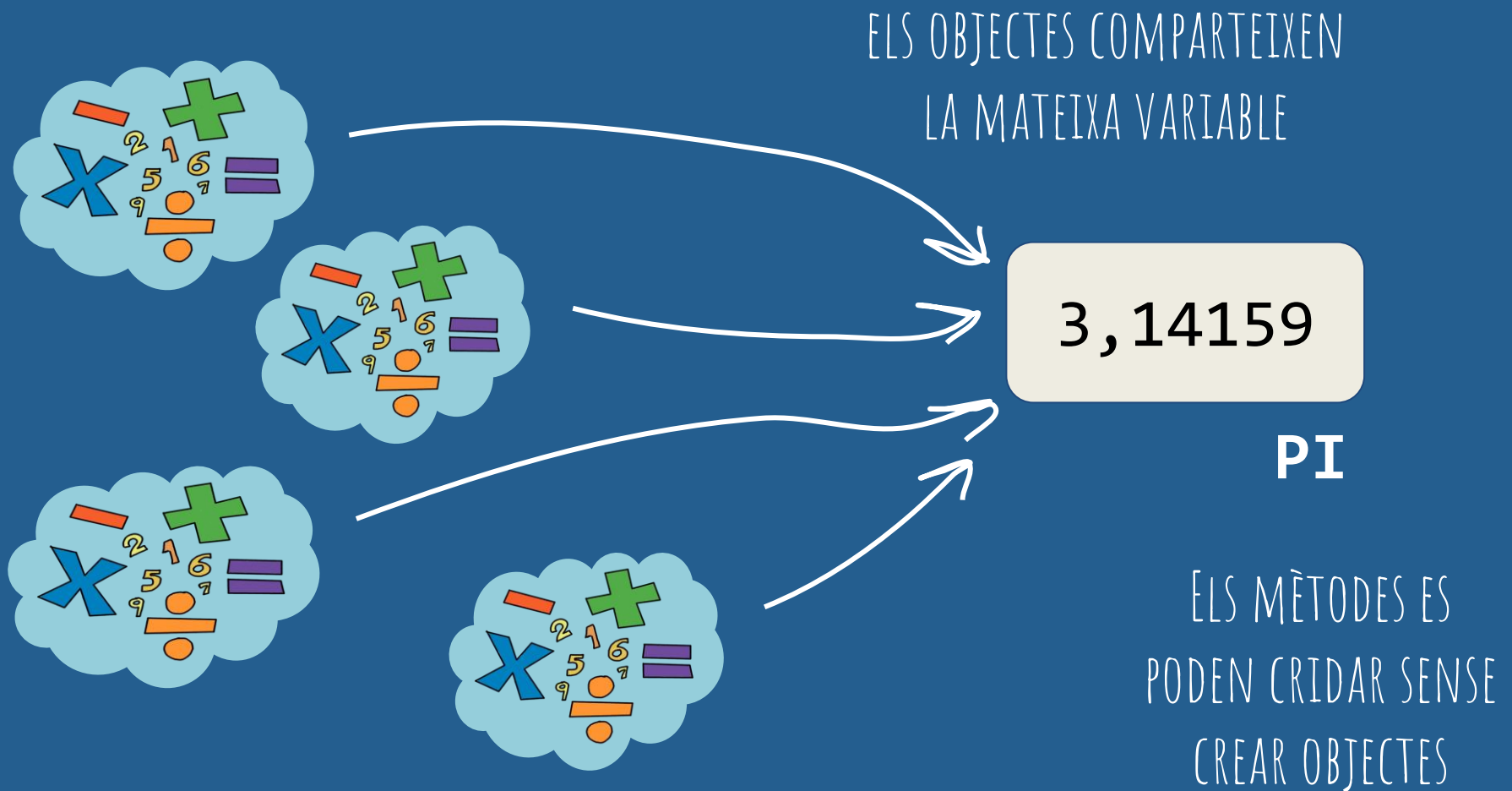


NULL

```
Cotxe cotxet = new Cotxe();  
cotxet = null;
```

MEMBRES DE LA CLASSE

Amb Java es poden fer **variables** i **mètodes de classe**



El modificador **static** es el que defineix membres de classe

```
class Matematiques {  
    public static int PI = 3,14159;  
    ...  
}
```

Tots els objectes tindran la mateixa propietat

S'hi pot accedir des d'altres classes afegint-hi el **nom de la classe**

```
int area = Matematiques.PI *  
          radi * radi;
```

Els **mètodes static** poden ser executats sense crear cap objecte

```
class Matematiques {  
    public static mult(int x, int y){  
        return x*y;  
    }  
}
```

```
int x = Matematiques.mult(3,4);
```

EL MODIFICADOR FINAL

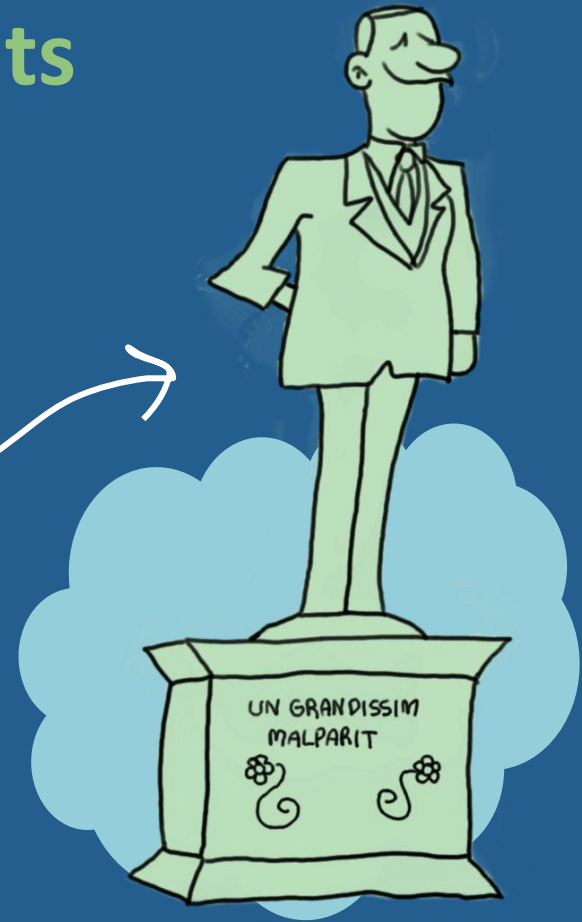
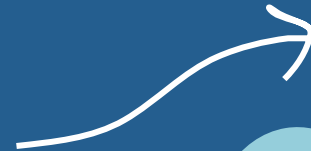
final es fa servir per indicar que
no es pot canviar

És per
sempre



És la forma de definir **propietats constants**

NO ES PODEN
MODIFICAR

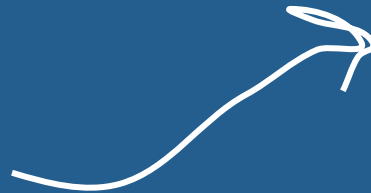


```
public static final int X = 3;
```

De marcar mètodes que **no
poden ser sobreescrits**

Servida!

NO ES PODEN
CANVIAR



I classes que **no poden** **ser esteses**

NO POT CREAR
DESCENDENTS

VULL CONVERTIR-TE EN UN
COTXE DE CARRERES



Ho sento, sóc
final

Versió 2 - 2019



ORIGINAL: XAVIER SALA PUJOLAR - ACTUALITZACIÓ: PEDRO DURÁN DÍEZ

Les imatges són de OpenClipart o meves