



**UF08.- POO (I)** 

LA COMUNITAT VALENCIANA

- Teoria -

#### PROGRAMACIÓ CFGS DAW

Joan V. Cassany

jv.cassanycoscolla@edu.gva.es





## 1. CLASSES I OBJECTES INSTANCIACIÓ

CLASSE





Cotxe cotxeMeu = new Cotxe

Cotxe cotxeSilvia = new Cotxe



**Classe: Cotxe** 

**Atributs**: color, rodes, matricula, marca ... **Mètodes**: *Cotxe*, posarColor, mostrarColor



**Objecte: cotxeMeu** 

Atributs:

color=Blau, rodes=4, matricula=1234 ...

Mètodes:

posarColor, mostrarColor

El mètode Cotxe és un constructor, necessari en totes les classes per a crear instàncies.

Ha de tindre el mateix nom que la classe.

Objecte: cotxeSilvia

Atributs:

Color=Verd, rodes=4, matricula=5678 ...

Mètodes:

posarColor, mostrarColor

## 2. CONSTRUCTORS BASICS







Cotxe cotxeMeu = new Cotxe



Cotxe cotxeSilvia = new Cotxe

## OBJECTE

#### **Classe: Cotxe**

```
public class Cotxe {
  int rodes, llarg, ample, motor, pes;
  String color, marca, model;
  // Mètode constructor amb el mateix nom que la classe
  Cotxe() {
    rodes=4; } }
```



```
public class ProgramaCotxes {
   public static void main (String[] args) {
      Cotxe cotxeMeu= new Cotxe ();
      System.out.println ("El cotxe Meu té " + cotxeMeu.rodes + " rodes");
      Cotxe cotxeSilvia= new Cotxe ();
      System.out.println ("El cotxe de Silvia té " + cotxeSilvia.rodes + " rodes");
    }
}
```

#### 3. CONSTRUCTORS **PARÀMETRES**

CLASSE





Cotxe cotxeMeu = new Cotxe



Cotxe cotxeSilvia = new Cotxe

## **OBJECTE**

#### **Classe: Cotxe**

```
public class Cotxe {
  int rodes, llarg, ample, motor, pes;
  String color, marca, model;
 // Mètode constructor amb el mateix nom que la classe
  Cotxe(String marca) {
    this.marca=marca;
    rodes=4;
```



```
public class ProgramaCotxes {
  public static void main (String[] args) {
    Cotxe cotxeMeu= new Cotxe ("Ford");
    System.out.println ("El cotxe Meu de la marca" + cotxeMeu.marca + " té ....);
    Cotxe cotxeSilvia= new Cotxe ("Seat");
    System.out.println ("El cotxe de Silvia de la marca" + cotxeMeu.marca + ...");
```

## 4. ENCAPSULAMENT VARIABLES NO ENCAPSULADES







Cotxe cotxeMeu = new Cotxe



Cotxe cotxeSilvia = new Cotxe



#### Classe: Cotxe

```
public class Cotxe {
  int rodes, llarg, ample, motor, pes;
  String color, marca, model;
  // Mètode constructor amb el mateix nom que la classe
  Cotxe(String marca) {
    this.marca=marca;
    rodes=4;
    }
}
```



```
public class ProgramaCotxes {
  public static void main (String[] args) {
    Cotxe cotxeMeu= new Cotxe ("Ford");
    System.out.println ("El cotxe Meu .... té " + cotxeMeu.rodes + " rodes");
    cotxeMeu.rodes=3; // Podem fer açò sense impediment
    System.out.println ("El cotxe Meu ... té " + cotxeSilvia.rodes + " rodes"); } }
```

## 4. ENCAPSULAMENT VARIABLES ENCAPSULADES







Cotxe cotxeMeu = new Cotxe



Cotxe cotxeSilvia = new Cotxe

# OBJECTE

#### Classe: Cotxe

```
public class Cotxe {
  private int rodes, llarg, ample, motor, pes;
  private String color, marca, model;
  // Mètode constructor amb el mateix nom que la classe
Cotxe(String marca) {
    this.marca=marca;
    rodes=4;  } }
```

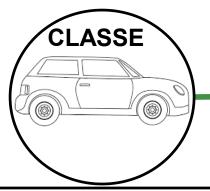


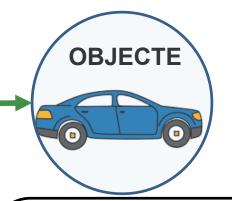
```
public class ProgramaCotxes {
   public static void main (String[] args) {
     .....
   // Les següents dos instruccions ara donaran error
   cotxeMeu.rodes=3;
   System.out.println ("El cotxe Meu ... té " + cotxeSilvia.rodes + " rodes");   } }
```

### 4. ENCAPSULAMENT MÈTODES SETTER I GETTER











#### Classe: Cotxe

```
public class Cotxe {
    private int rodes, llarg, ample, motor, pes;
    private String color, marca, model;
    // Mètode constructor amb el mateix nom que la classe
    Cotxe(String marca) {
        this.marca=marca;
        rodes=4;
        }
    // Mètode setter
    public void estableix_color(String color){
        this.color=color;
     }
    // Mètode getter
    public String mostra_color () {
        return "Aquest cotxe té color" + color;
     }
}
```

```
public class ProgramaCotxes {
   public static void main (String[] args) {

   Cotxe cotxeMeu= new Cotxe ("Ford");
   cotxeMeu.estableix_color("blau");

   System.out.println cotxeMeu.mostra_color());

   Cotxe cotxeSilvia= new Cotxe ("Seat");
   cotxeSilvia.estableix_color("verd");
   System.out.println cotxeSilvia.mostra_color());
}
```



### 4. ENCAPSULAMENT ÀMBIT DE VARIABLES I MÈTODES





En definir els elements d'una classe (variables o mètodes), es poden especificar els seus àmbits (scope) de visibilitat o accessibilitat amb les paraules reservades **public** (públic), **protected** (protegit) i **private** (privat).

	En la mateixa classe	En el mateix paquet	En una subclasse	Fora del paquet
private	~	×	×	×
protected	~	<b>~</b>	~	×
public	<b>~</b>	<b>~</b>	<b>~</b>	<b>~</b>
sin especificar	<b>/</b>	<b>*</b>	×	×

- Un atribut private només és accessible des de la mateixa classe
- A un mètode protected es podrà accedir des de la mateixa classe on estigua definit, des d'un altre fitxer dins del mateix paquet o des d'una subclasse.
- Com a regla general, se solen declarar private els atributs o variables d'instància i public els mètodes.



