

GRAU D'ENGINYERIA INFORMÀTICA

# **PROGRAMACIÓ II**

## **CURS 12-13**

**Bloc 2:**

## **Programació Orientada a Objectes (1)**

**Laura Igual**

Departament de Matemàtica Aplicada i Anàlisi

Facultat de Matemàtiques

Universitat de Barcelona

# Índex Bloc 2:

## Programació orientada a objectes

- Abstracció en el desenvolupament del *software*
- Conceptes fonamentals: classes i objectes
- Característiques de l'orientació a objectes
- Ús de classes i objectes
- Constructors i destructors
- Encapsulació
- Herència i jerarquia de classes
- Polimorfisme
- Lligadures
- Interfícies
- col·leccions

# **ABSTRACCIÓ EN EL DESENVOLUPAMENT DEL *SOFTWARE***

# Abstracció en el desenvolupament de software

- Abstracció:  
Extracció de les característiques essencials d'un objecte i dels seus comportaments.
- Una vegada s'han identificat els objectes, identificar les seves relacions en el món real.

# Abstracció en el desenvolupament de software

- **Orientació a Objectes (OO)** consisteix en organitzar el software com una col·lecció discreta d'entitats que incorporen:
  - les dades
  - el comportament d'aquestes dades.
- **Classes:** entitats en les que la OO estructura el software en dades i el conjunt d'operacions associades a aquestes dades.
- Un **programa** és un conjunt **d'objectes** (que pertanyen a diferents classes) que **interactuen entre si** per tal de resoldre el problema.

# Programació Orientada a Objectes

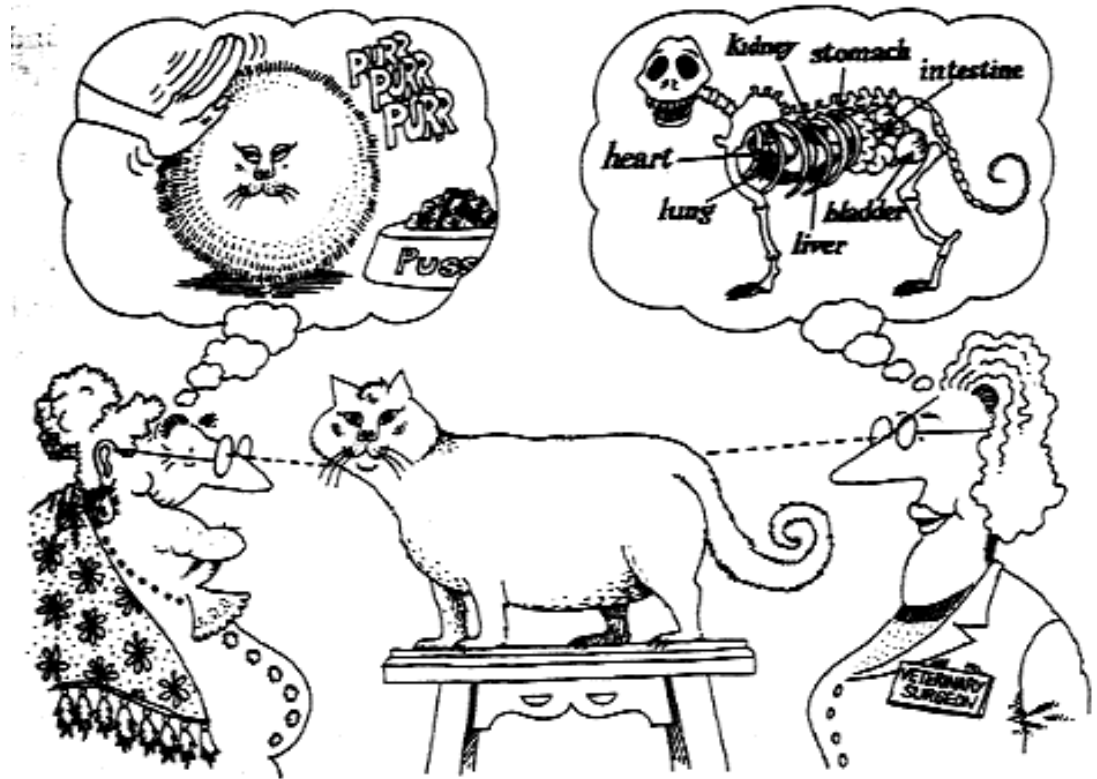
- La OO és una filosofia, no està lligada a cap llenguatge de programació.
- Java és un llenguatge de programació orientat a objectes (POO).

# **CARACTERÍSTIQUES DE L'ORIENTACIÓ A OBJECTES**

# Característiques de la OO

## Abstracció

- Consisteix en agafar una informació i extreure'n les característiques més representatives



Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

Figura extreta de la pàgina 39 del llibre: **“Object-Oriented Analysis and Design with Applications”**. Grady Booch. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.



# Característiques de la OO

## Encapsulament

- Amaga a l'usuari la implementació interna de l'objecte

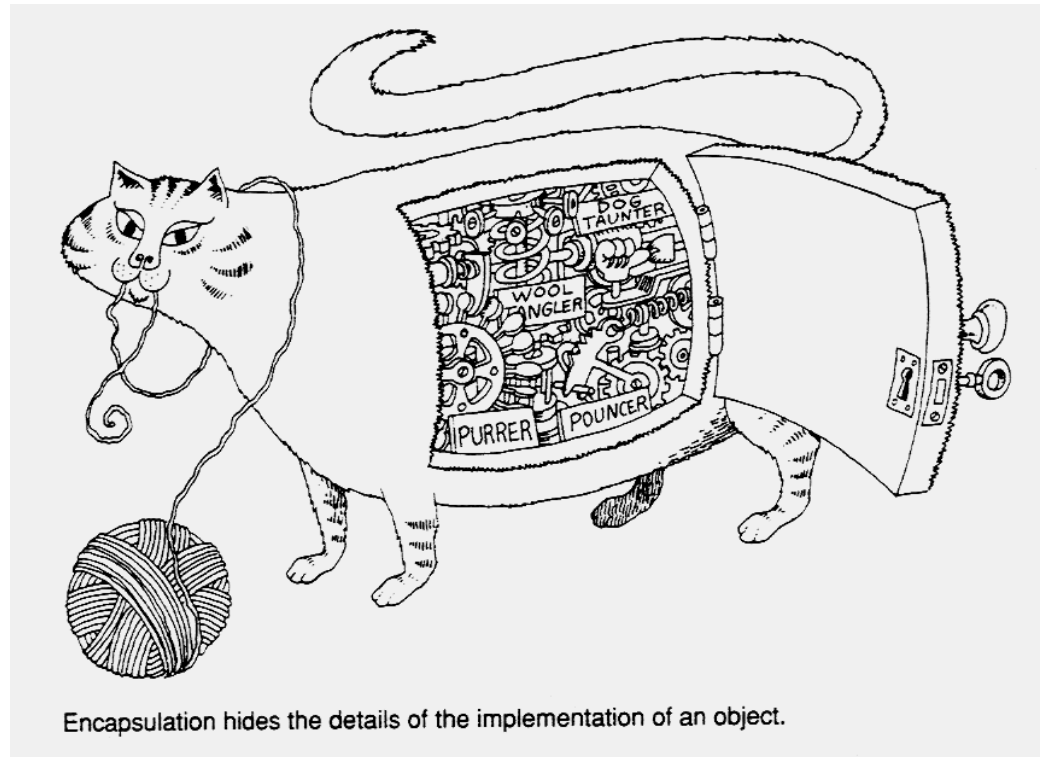


Figura extreta de la pàgina 46 del llibre: **“Object-Oriented Analysis and Design with Applications”**. Grady Booch. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

# Característiques de la OO

## Herència

- Defineix una relació entre classes.
- Una classe (**superclasse**) defineix un conjunt de propietats comuns a altres classes (**subclasses**).
- Les classes es poden organitzar en jerarquies

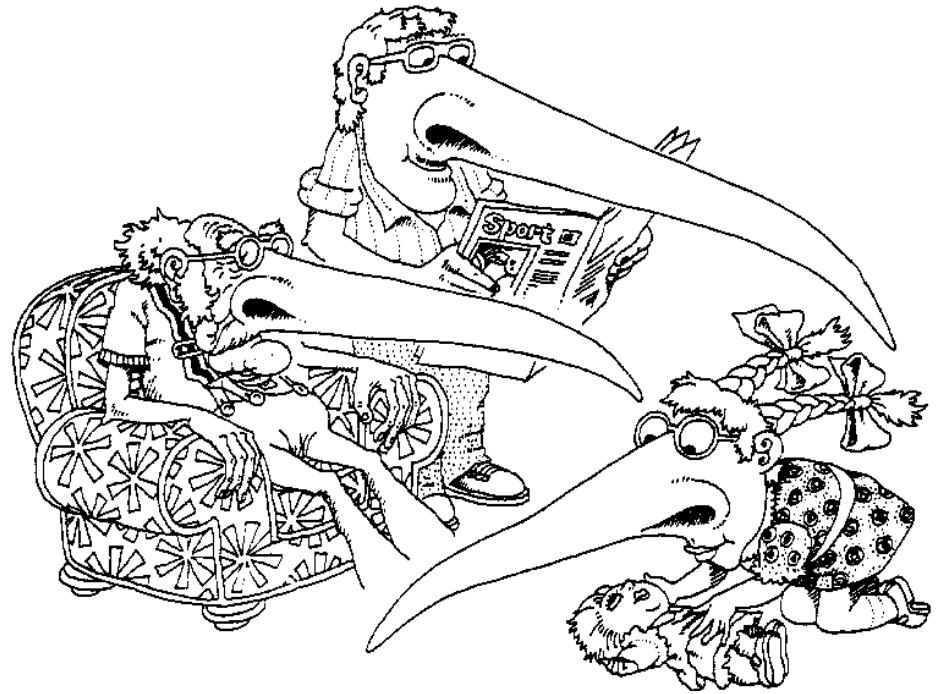


Figura extreta de la pàgina 109 del llibre: “**Object-Oriented Analysis and Design with Applications**”. Grady Booch. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

# **CONCEPTES FONAMENTALS: CLASSES I OBJECTES**

# Objectes

- Els objectes es poden usar per representar entitats del mon real
- Un objecte està format per:
  - Un conjunt de dades (**atributs o estats**)
  - Un conjunt d'operacions (**mètodes o comportament**)
- **El comportament d'un objecte pot modificar el seu estat**
- Cada objecte té un identificador únic pel qual pot ser referenciat.
- L'usuari no gestiona l'objecte directament → fa una petició a l'objecte d'un servei que ofereix ell mateix.



Serveis:

- Engregar
- Apagar
- Canviar d'emisora
- Pujar el volum
- Disminuir el volum

# Primer és l'objecte i després la classe

- **Classe:** descriu un grup d'entitats amb característiques comunes
- **Objecte:** descriu un membre concret del grup.

## Cotxe



### Classe cotxe:

Patró que defineix atributs i mètodes comuns a tots els exemples de **cotxes**.

Classe cotxe
marca
model
color
número portes

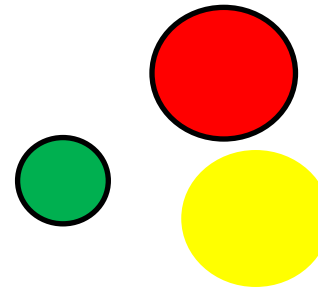
# Els papers de la classe i l'objecte

- Una **classe** és una abstracció d'un cert concepte.
- Un **objecte** es defineix mitjançant una classe.
- Es diu que un objecte és una **instància** d'una classe.

La **classe** representa un concepte:  
Figura, Bicicleta,...

Un **objecte** representa la materialització d'aquest concepte.: cercle de radi 2, bicicleta de color vermell,...

Objectes de cerlces



Info  
abstracta  
→

Cercle
radi color
dibuixa borra mou

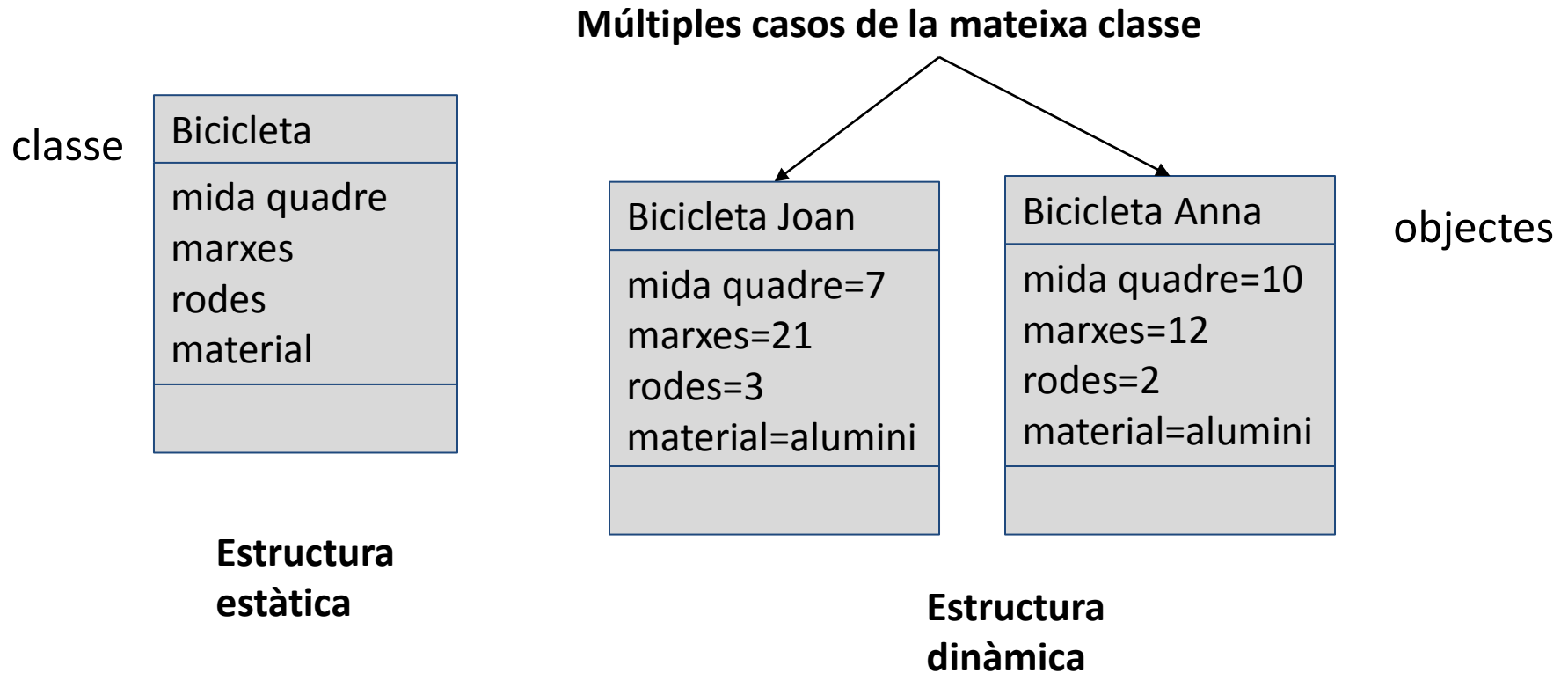
Objectes de bicicletes



Info  
abstracta  
→

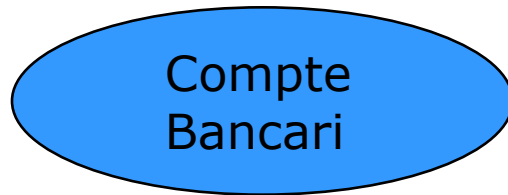
Bicicleta
tamany rodes material color
moure reparar

# Objectes i classes



# Objectes i classes

**Una classe  
(el concepte)**



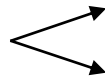
**Un objecte  
(la materialització,  
la realització)**

Compte bancari de Joan  
Saldo: 5,257€

Compte bancari de Blas  
Saldo: 1,245,069€

Compte bancari de Maria  
Saldo: 16,833€

**Múltiples objectes  
de la mateixa classe**



- L'estat d'un compte de banc inclou el seu **Saldo**
- Els comportaments associats amb un compte de banc inclouen la capacitat de fer ingressos i extraccions
- El comportament d'un objecte pot, per tant modificar el seu estat

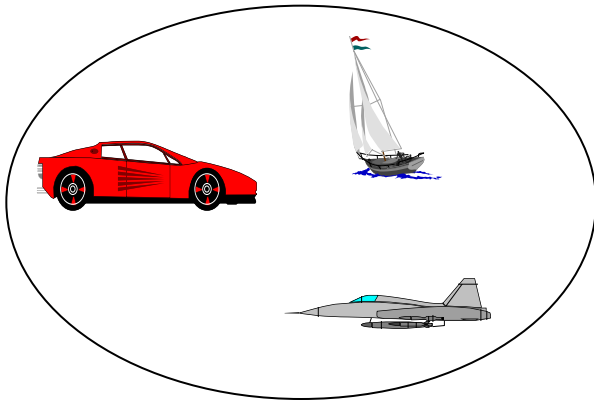


# Exemple

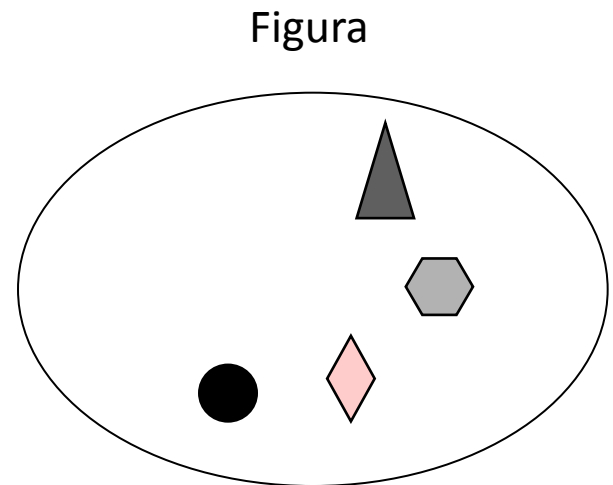
- Classe: Persona
- Objectes: Maria, Joan,...

Classe Persona
nom cognoms sexe data naixement nacionalitat dni
edat

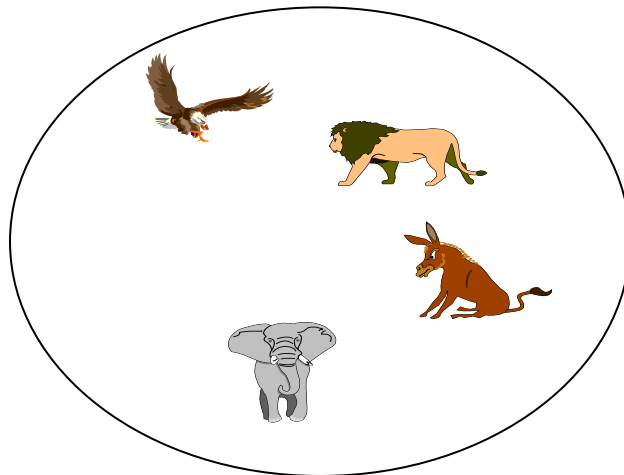
# Exemplos de classes



Vehicle



Figura



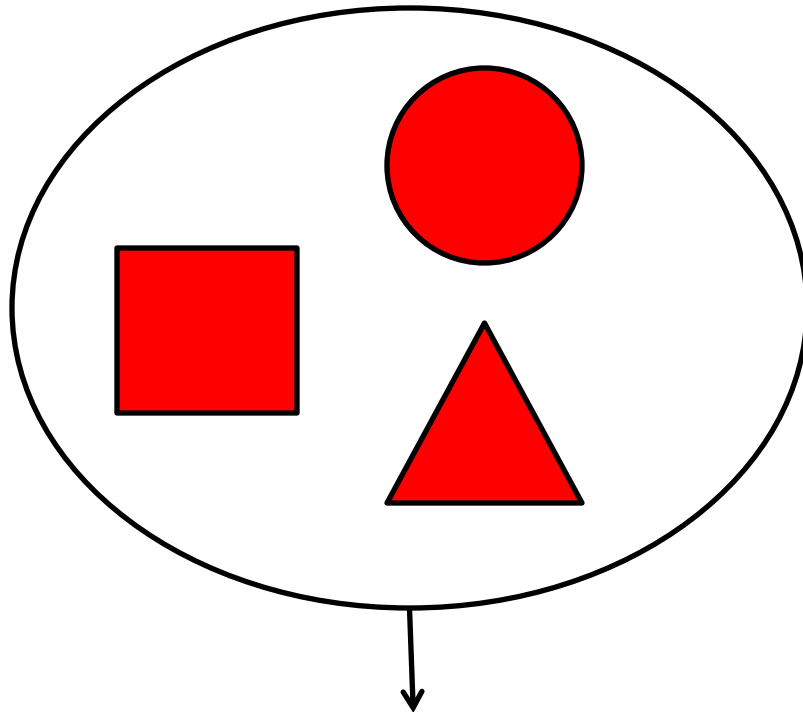
Animal

# Classes

- Una classe és un plànol (o pla d'obra) d'un objecte
  - La classe utilitza els mètodes per definir els comportaments de l'objecte.
  - Es poden crear múltiples objectes d'una mateixa classe.
  - Els objectes comparteixen el nom dels atributs i les operacions, però cada objecte té un valor concret per cada atribut.
  - La classe que conté el mètode main d'un programa JAVA representa el programa complet.

# Exemple

- Herència → Jerarquia de classes
- Classe abstracta: Figura geomètrica
- Classe: Triangle, quadrat, cercle, ...



Figures geomètriques → Dibuixar

Figura geomètrica
color posició a pantalla àrea perímetre
calcula àrea calcula perímetre retorna color assigna color

Quadrat
color posició a pantalla àrea perímetre dimensió costats

Circumferència
color posició a pantalla àrea perímetre Radi

Triangle
color posició a pantalla àrea perímetre dimensió 3costats

# ÚS DE CLASSES I OBJECTES

# Ús de classes

```
public class MiClase {  
    int i;  
    public MiClase() {  
        i = 10;  
    }  
    public void suma_a_i( int j ) {  
        i = i + j;  
    }  
}
```

Nom de la classe

atribut

Constructor


Mètode o servei

# Ús de classes

```
public static void main(String[] args) {  
    MiClase mc;  
    mc = new MiClase();  
    mc.i++;  
    mc.suma_a_i(10);  
  
    System.out.println(mc.i);  
}
```



crea una instància de la classe



Crida a un mètode de l'objecte

# Ús de classes

- La **sobrecàrrega** és la capacitat de poder associar més d'un significat a un mateix identificador que apareix dins d'un programa.
- Es pot produir sobrecàrrega en:
  - Els noms dels mètodes
  - Els operadors



# Ús de classes

- Sobrecàrrega

```
public class MiClase {  
    int i;  
    public MiClase() {  
        i = 10;  
    }  
    public MiClase(int i) {  
        this.i = i;  
        // i = valor  
    }  
    public void suma_a_i( int j ) {  
        i = i + j;  
    }  
}
```

La paraula **this** és una referència al objecte (l'argument implícit) sobre el que s'està aplicant el mètode.

**this.i** es refereix a la variable membre, mentres que **i** és l'argument del mètode.

# Ús de classes

## Creant Objectes

- Una variable conté un tipus primitiu o una **referència a un objecte** (reference)
- Un nom de classe pot utilitzar-se com a tipus per declarar una variable que referència a un objecte  
`String title;`  
`MiClasse unExemple;`
- En aquesta declaració **no** es crea cap objecte
- Una variable que referència un objecte conté l'adreça de l'objecte
- L'objecte en si mateix s'ha de declarar de forma separada

# Ús de classes

## Creant Objectes II

- Usarem l'operador new per crear un objecte  
`title = new String("Java Software");`



Crida al constructor de la classe String, que és un mètode especial que prepara l'objecte

# Ús de classes

## Creant Objectes III

Tres passos: declaració, creació i assignació:

```
Persona laPersona = new Persona();
```

```
class Persona {  
    public String nom;  
}
```

1

Declaració d'una  
variable de  
referència

2

Creació d'un objecte

3

Assignació el nou objecte  
Persona a la variable  
laPersona.

```
laPersona.setNom("Lluís")
```

laPersona

Persona

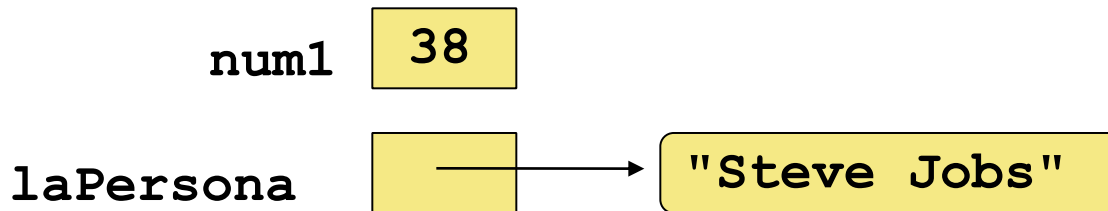
Lluís

Persona

# Ús de classes

## Referències

- Una variable de tipus primitiu conté el valor però una variable objecte conté l'adreça de l'objecte
- Exemples :



# Ús de classes

## Assignació

- De tipus primitius: Exemple:

Abans

num1	38
num2	96

`num2 = num1;`

Després

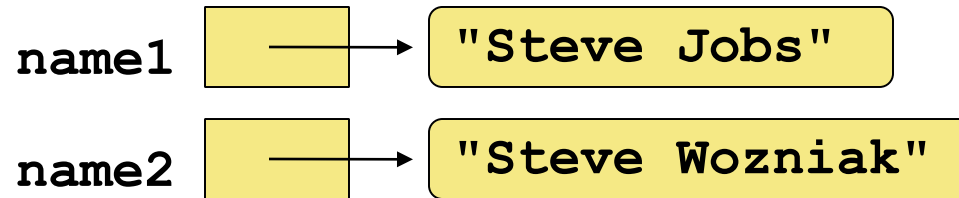
num1	38
num2	38

# Ús de classes

## Assignació de referències

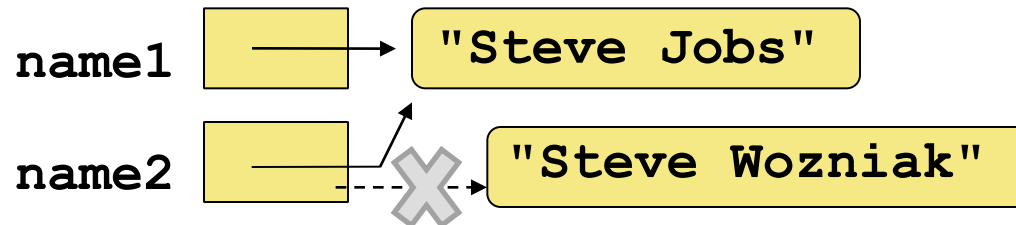
- Per referències a objectes, l'assignació còpia l'adreça

Abans



```
name2 = name1;
```

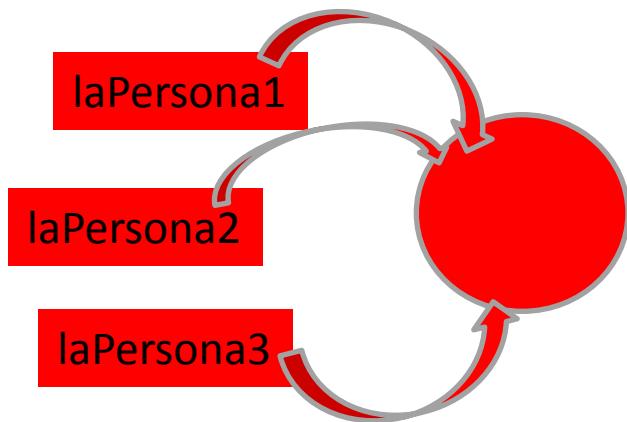
Després



# Ús de classes

## Aliases

- Dos o més referències que es refereixen al mateix objecte s'anomenen **aliases**
- Canviant un objecte a través d'una referència, canvia tots els seus aliases perquè realment només hi ha un objecte.



```
Persona laPersona1 = new Persona(20);  
Persona laPersona2;  
Persona laPersona3;  
laPersona2 = laPersona1;  
laPersona3 = laPersona1;  
laPersona1.edat = 21;  
System.out.println(laPersona3.nom "té" +  
laPersona3.edat + "anys");
```



# Ús de classes: Aliases

```
public class Aliases {  
    public static void main(String[] args) {  
        Persona x = new Persona();  
        Persona y = new Persona();  
        x.nom = "Joan";  
        y.nom = "Lluís";  
        Persona z;  
        z = x;  
        x = y;  
  
        x.nom = "Marc";  
        System.out.println("El nom en l'objecte referenciat per x és:" + x.nom);  
        System.out.println("El nom en l'objecte referenciat per y és:" + y.nom);  
        System.out.println("El nom en l'objecte referenciat per z és:" + z.nom);  
    }  
}
```

```
class Persona {  
    public String nom;  
}
```

→ Sortida per pantalla:

Marc  
Marc  
Joan

# Exercici 1: Assignació de referències

```
public class Aliases {  
    public static void main(String[]args) {  
        Persona x = new Persona();  
        x.edat = 23;  
        Persona y = new Persona();  
        y.edat = 25;  
        Persona z;  
        z = y;  
        y = x;  
        x = z;  
  
        x.edat = 26;  
        System.out.println("L'edat en l'objecte referenciat per x és:" + x.edat);  
        System.out.println("L'edat en l'objecte referenciat per y és:" + y.edat);  
        System.out.println("L'edat en l'objecte referenciat per z és:" + z.edat);  
    }  
}
```

```
class Persona {  
    public String nom;  
    public int edat;  
}
```

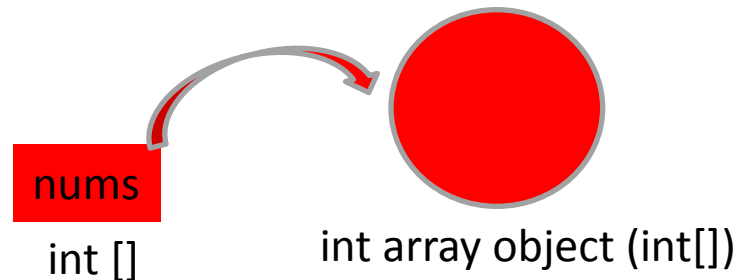
→ Sortida per pantalla?

# Ús de classes

- Un array també és un objecte

```
int [] nums;  
nums = new int[7];
```

```
nums[0]=6;  
nums[1]=19;  
nums[2]=2;  
nums[3]=32;  
nums[4]=5;  
nums[5]=15;  
nums[6]=11;
```



- L'array pot contenir primitives o objectes.

# Exemple

Implementar el joc d'endevinar el número que la màquina a pensat

Tres classes:

- **GuessGame** – joc d'endevinar
- **Player** – jugador
- **GameLauncher** – llançador del joc

```

public class GuessGame {
    Player p1;
    Player p2;
    Player p3;
    public void startGame() {
        p1 = new Player();
        p2 = new Player();
        p3 = new Player();
        int guessp1 = 0;
        int guessp2 = 0;
        int guessp3 = 0;
        boolean p1isRight = false;
        boolean p2isRight = false;
        boolean p3isRight = false;
        int targetNumber = (int) (Math.random() * 10);
        System.out.println("I'm thinking of a number between 0 and
            9...");
        while(true) {
            System.out.println("Number to guess is " +
                targetNumber);
            p1.guess();
            p2.guess();
            p3.guess();
            guessp1 = p1.number;
            System.out.println("Player one guessed " + guessp1);
            guessp2 = p2.number;
            System.out.println("Player two guessed " + guessp2);
            guessp3 = p3.number;
            System.out.println("Player three guessed " + guessp3);

```

3 variables d'instancia  
pels 3 jugadors

```

        if (guessp1 == targetNumber) {
            p1isRight = true;
        }
        if (guessp2 == targetNumber) {
            p2isRight = true;
        }
        if (guessp3 == targetNumber) {
            p3isRight = true;
        }
        if (p1isRight || p2isRight || p3isRight) {
            System.out.println("Player one got it right? " +
                p1isRight);
            System.out.println("Player two got it right? " +
                p2isRight);
            System.out.println("Player three got it right? " +
                p3isRight);
            System.out.println("Game is over.");
            break; // game over, so break out of the loop
        } else {
            // we must keep going because nobody got it right!
            System.out.println("Players will have to try
                again.");
        } // end if/else
    } // end loop while
} // end method
} // end class

```

# Example

```
public class Player {  
    int number = 0; // where the guess goes  
    public void guess() {  
        number = (int) (Math.random() * 10);  
        System.out.println("I'm guessing "+ number);  
    }  
}  
  
public class GameLauncher {  
    public static void main (String[] args) {  
        GuessGame game = new GuessGame();  
        game.startGame();  
    }  
}
```

Sortida per pantalla:

```
%java GameLauncher
```

I'm thinking of a number between 0 and 9...

Number to guess is 7

I'm guessing 1

I'm guessing 9

I'm guessing 9

Player one guessed 1

Player two guessed 9

Player three guessed 9

Players will have to try again.

Number to guess is 7

I'm guessing 3

I'm guessing 0

I'm guessing 9

Player one guessed 3

Player two guessed 0

Player three guessed 9

Players will have to try again.

Number to guess is 7

I'm guessing 7

I'm guessing 5

I'm guessing 0

Player one guessed 7

Player two guessed 5

Player three guessed 0

We have a winner!

Player one got it right? true

Player two got it right? false

Player three got it right? false

Game is over.

# Example

# Referències

- Bertrand Meyer, “**Construcción de software orientado a objetos**”, Prentice Hall, 1998.
- “Software Architecture and UML” de Grady Booch (Rational Software). Presentació P. Letelier.
- Bert Bates, Kathy Sierra. **Head First Java**. O’Reilly Media, 2005.