

udo.

# Programación II

Bernarda Sandoval

# Tema: Diseño de clases

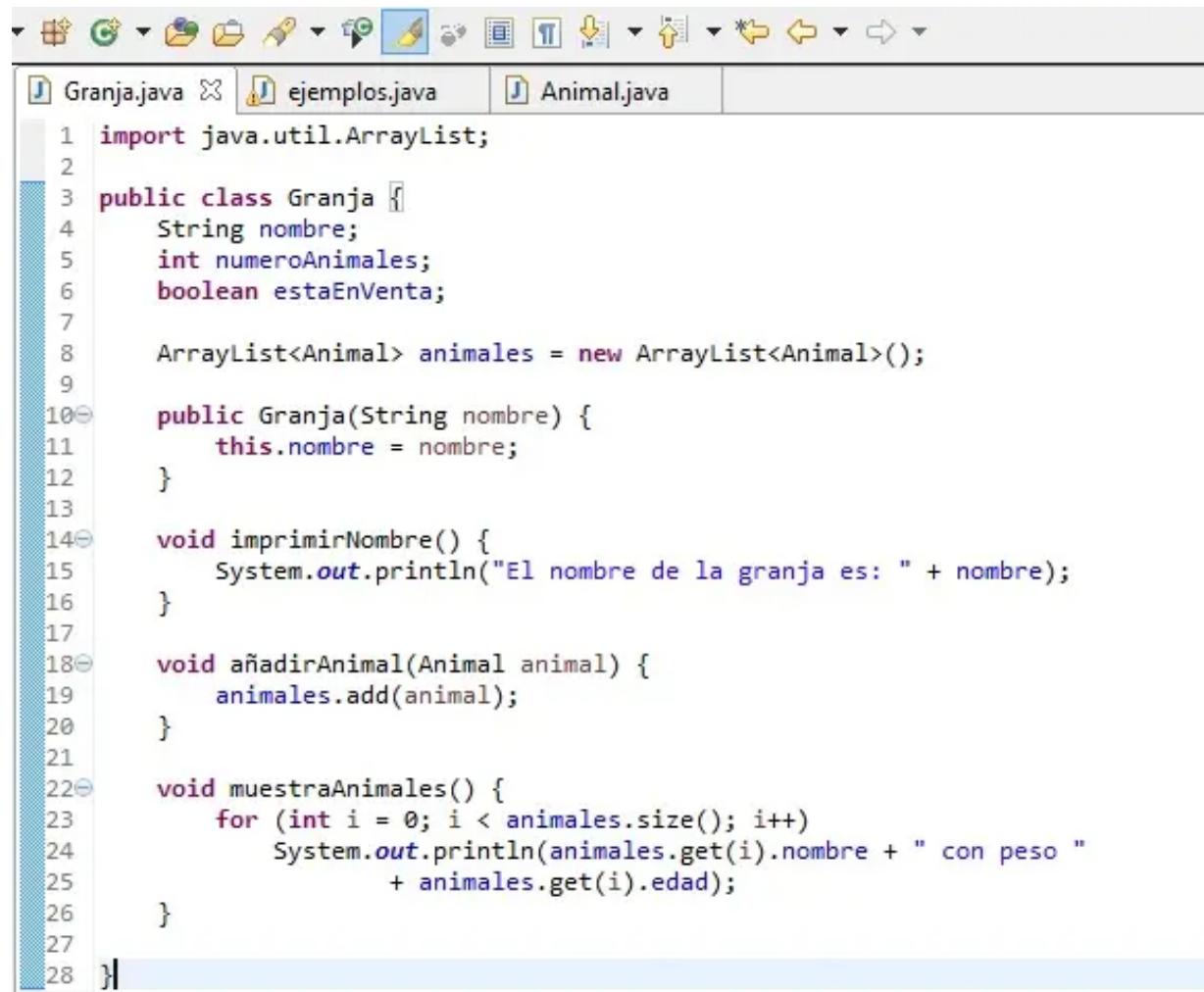
RDA

- Explica los principios y características propias de un paradigma de programación.

# Objetivo del tema

- Identificar la diferencia entre la POO y programación estructurada
- Conocer los pilares de la POO
- Identificar la importación de la POO

# Programación orientada a objetos



The screenshot shows a Java Integrated Development Environment (IDE) interface. The title bar has tabs for "Granja.java", "ejemplos.java", and "Animal.java". The main window displays the code for the "Granja" class. The code uses Java 8 features like lambdas and the Stream API. It includes methods for adding animals to a list and printing the list.

```
1 import java.util.ArrayList;
2
3 public class Granja {
4     String nombre;
5     int numeroAnimales;
6     boolean estaEnVenta;
7
8     ArrayList<Animal> animales = new ArrayList<Animal>();
9
10    public Granja(String nombre) {
11        this.nombre = nombre;
12    }
13
14    void imprimirNombre() {
15        System.out.println("El nombre de la granja es: " + nombre);
16    }
17
18    void añadirAnimal(Animal animal) {
19        animales.add(animal);
20    }
21
22    void muestraAnimales() {
23        for (int i = 0; i < animales.size(); i++)
24            System.out.println(animales.get(i).nombre + " con peso "
25                                + animales.get(i).edad);
26    }
27
28 }
```



EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Programación orientada a objetos

The screenshot shows a Java IDE interface with two tabs open: 'Granja.java' and 'ejemplos.java'. The 'Granja.java' tab contains the following code:

```
1 import java.util.ArrayList;
2
3 public class Granja {
4     String nombre;
5     int numeroAnimales;
6     boolean estaEnVenta;
7
8     ArrayList<Animal> animales = new ArrayList<Animal>();
9
10    public Granja(String nombre) {
11        this.nombre = nombre;
12    }
13}
```

The screenshot shows a Java IDE interface with one tab open: 'Mesa.java'. The code defines a class 'Mesa' with a private attribute 'color' and a public method 'getColor()' which returns the color. There is also a comment for a constructor.

```
1 package clases;
2
3 /**
4  * @author Pablo Ruiz Soria
5  */
6 public class Mesa {
7     private String color;
8
9     /**
10      * @return the color
11     */
12     public String getColor() {
13         return color;
14     }
15
16     /**
17      * @param color the color to set
18     */
19 }
```

The screenshot shows a Java IDE interface with one tab open: 'Coche.java'. The code defines a class 'Coche' with attributes 'marca', 'modelo', and 'motor'. It has a constructor that takes 'marca', 'modelo', and 'motor' parameters and initializes them. There is also a comment for a constructor.

```
1 /**
2  * @author Pablo Ruiz Soria
3  */
4 public class Coche {
5     String marca;
6     String modelo;
7     Motor motor;
8
9     public Coche(){
10
11     }
12
13     public Coche(String marca, String modelo, Motor motor){
14         this.marca = marca;
15         this.modelo = modelo;
16         this.motor = motor;
17     }
18 }
```

The screenshot shows a Java IDE interface with one tab open: 'UniversityStudent.java'. The code defines a class 'UniversityStudent' with attributes 'id', 'name', 'gender', 'university', 'career', and 'numSubjects'. It has a constructor that takes 'id', 'name', 'gender', 'university', 'career', and 'numSubjects' parameters and initializes them.

```
public class UniversityStudent {
    int id;
    String name;
    String gender;
    String university;
    String career;
    int numSubjects;
    public UniversityStudent(int id, String name, String gender,
                           String university, String career, int numSubjects) {
        this.id = id;
        this.name = name;
        this.gender = gender;
        this.university = university;
        this.career = career;
        this.numSubjects = numSubjects;
    }
}
```



EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Programación orientada a objetos

```
#include<stdio.h>
int main(){
int i,n;
float calif,suma,prom;

i=1;
suma=0;
printf("Programa que calcula el promedio de n \
calificaciones de un alumno, dadas por teclado.\n");
printf("Cuantas calificaciones va a proporcionar?");
scanf("%d",&n);
while(i<=n) {
    printf("Teclea la calificacion %d:",i);
    scanf("%f",&calif);
    if(calif>=0 && calif<=10){
        suma=suma+calif;
        i++;
    }
    else
        printf("calificacion no valida\n");
}
if(n>0){
    prom=(float) suma/n;
    printf("El promedio es: %.2f\n",prom);
}
system("pause");
return 0;
}
```

```
public class UniversityStudent {
    int id;
    String name;
    String gender;
    String university;
    String career;
    int numSubjects;
    public UniversityStudent(int id, String name, String gender,
                           String university, String career, int numSubjects) {
        this.id = id;
        this.name = name;
        this.gender = gender;
        this.university = university;
        this.career = career;
        this.numSubjects = numSubjects;
    }
}
```



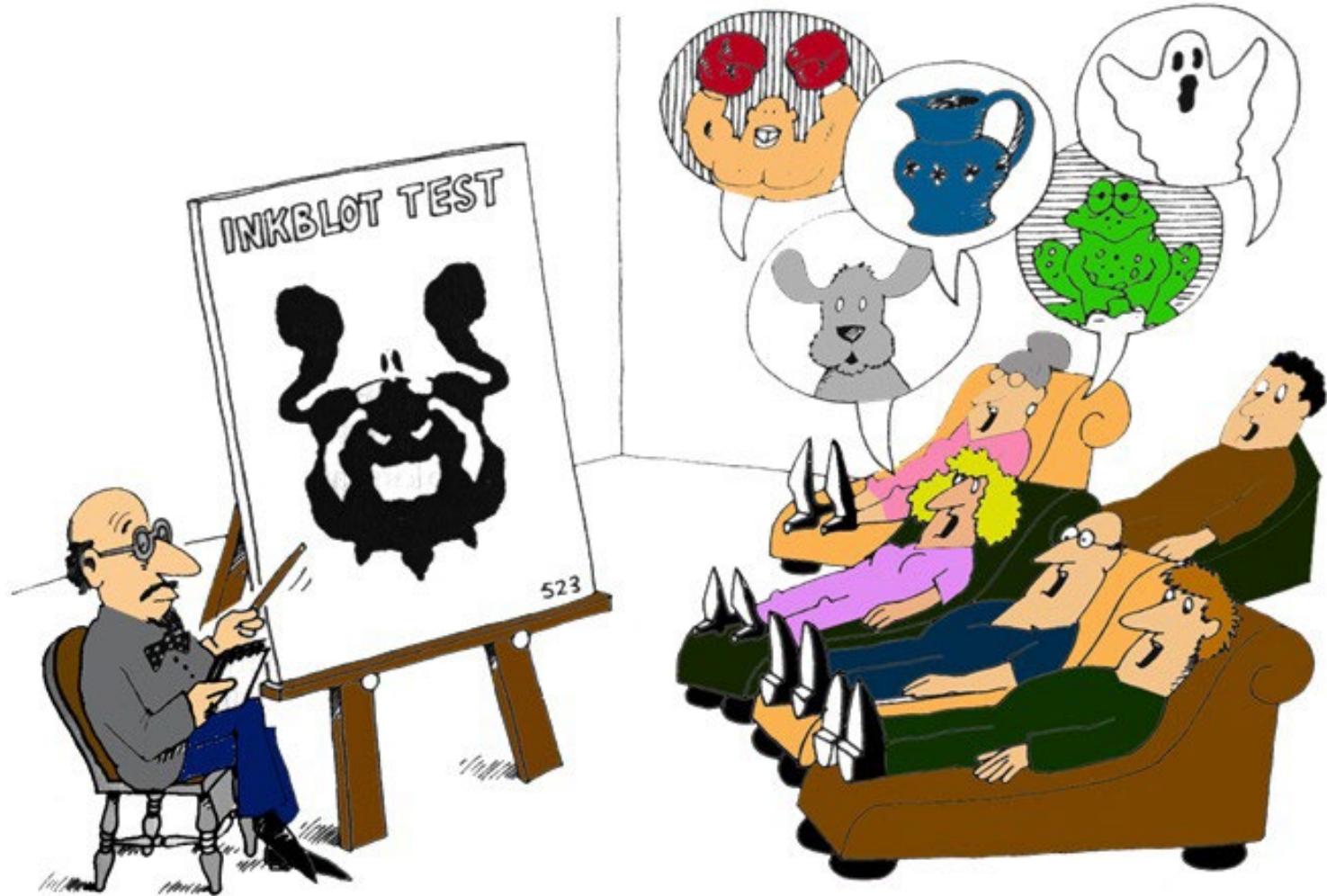
# Pilares de Poo

  
**EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE**



UNIVERSIDAD DE LAS AMÉRICAS © 2016

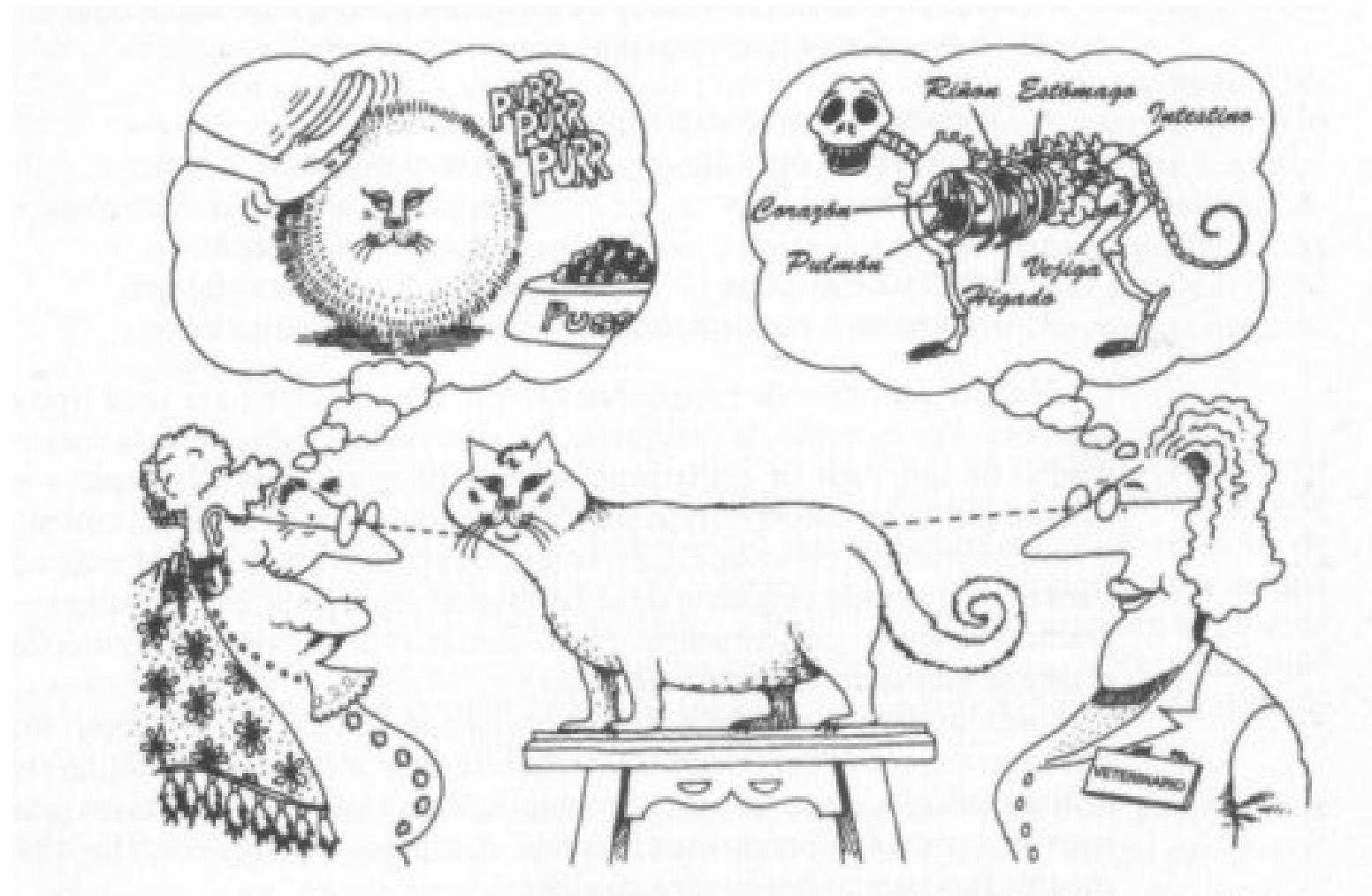
# Abstracción



udo

EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Abstracción



*udo*

EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

*udo*

UNIVERSIDAD DE LAS AMÉRICAS © 2016

# Pilares de la POO

## Abstracción

- Permite identificar las características y comportamientos de un objeto, con los cuales se construirá la clase (plantilla).
- Capacidad de obtener y aislar toda la información y cualidades de un objeto que nos parezcan relevantes
- La clase abstrae todo lo que representa un objeto, tomando solamente lo que nos interesa, descartando todo lo demás.



EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Pilares de la POO

## Abstracción

- ¿Qué características podemos abstraer de los automóviles? O lo que es lo mismo ¿Qué características semejantes tienen todos los automóviles?
- Todos tendrán una marca, un modelo, número de chasis, precio, puertas, ventanas, etc.
- Y en cuanto a su comportamiento todos los automóviles podrán acelerar, frenar, retroceder, etc.

# Ejemplos de sillas



*udla*

EL MUNDO  
NECESITA GENTE  
**QUE AME**  
LO QUE HACE

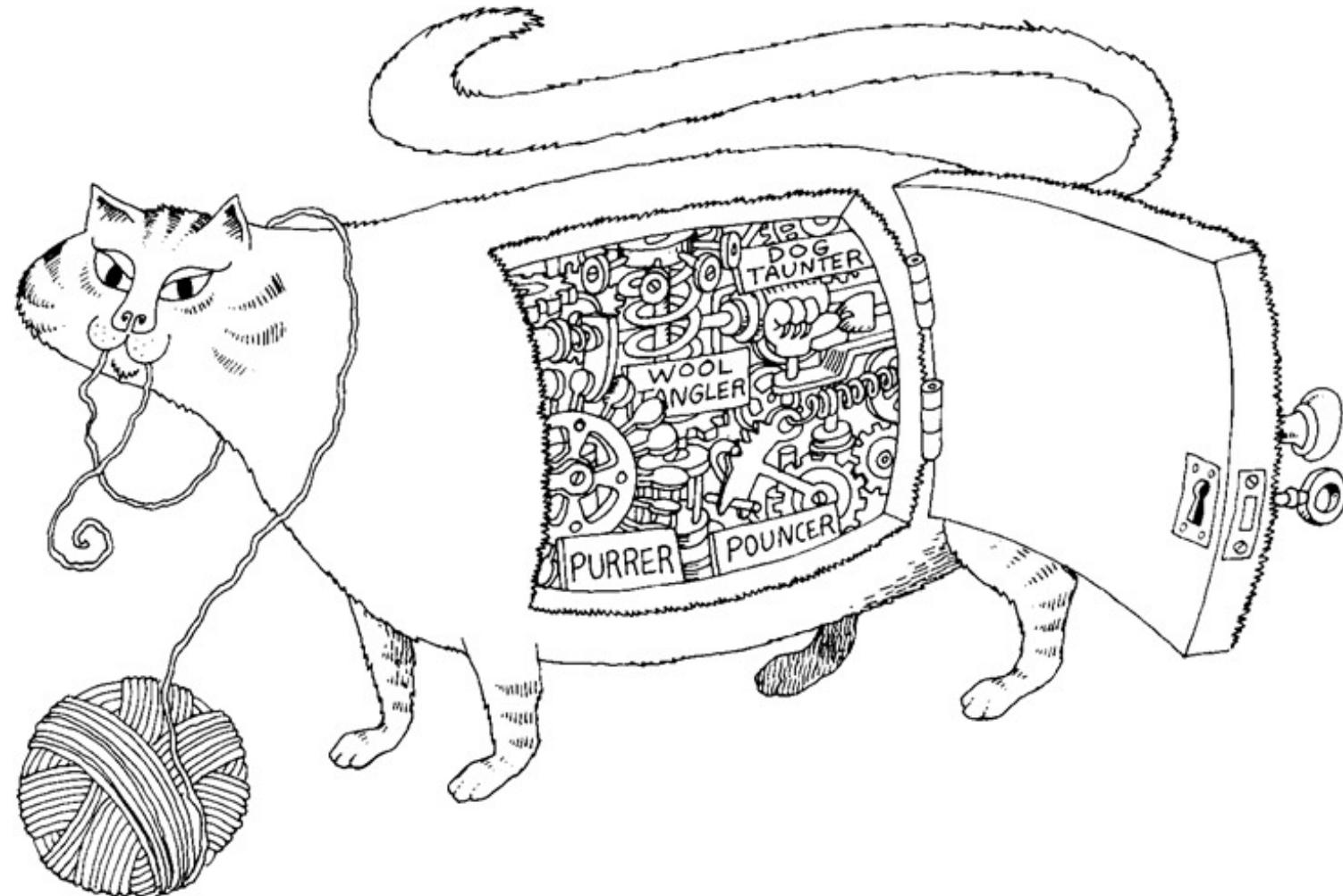
# Ejemplos de carros



*udla*

EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Encapsulamiento



*udoo*

EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

*udoo*

UNIVERSIDAD DE LAS AMÉRICAS © 2016

# Encapsulamiento



COURTESY: TESLA MOTORS, INC.



# Pilares de la POO

## Encapsulación

- La encapsulación consiste en formar un “paquete” con los atributos (variables) y el comportamiento (métodos) de un objeto
- Los métodos forman la membrana exterior de un objeto y “esconden” los detalles de implementación al usuario

# Pilares de la POO

## Encapsulación

- La encapsulación hace que un sistema sea más fácil de comprender y facilita el mantenimiento de una aplicación



EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Herencia



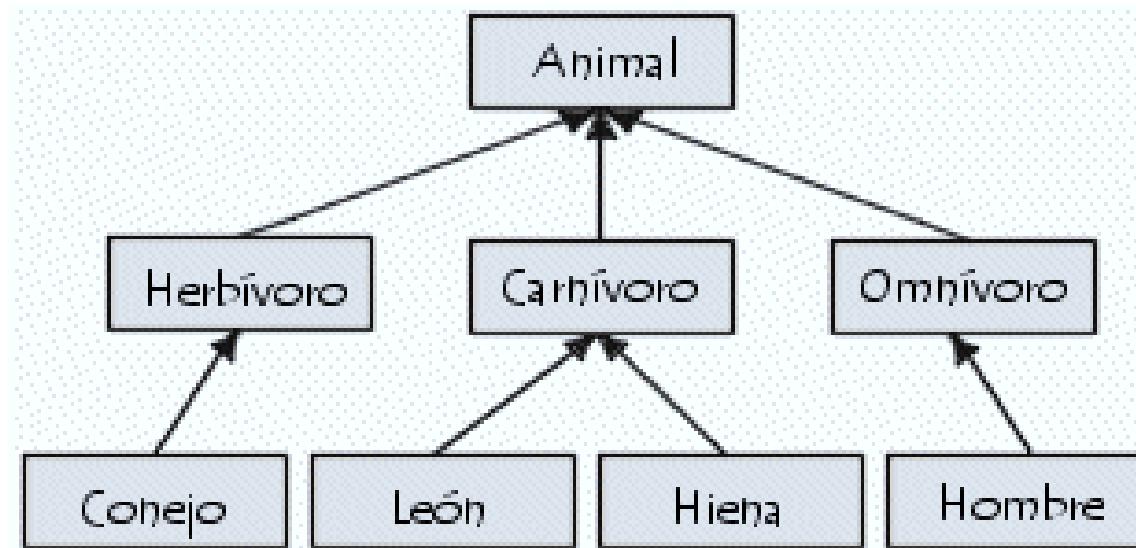
*udc*

EL MUNDO  
NECESITA GENTE  
**QUE AME**  
LO QUE HACE

*udc*

UNIVERSIDAD DE LAS AMÉRICAS © 2016

# Herencia

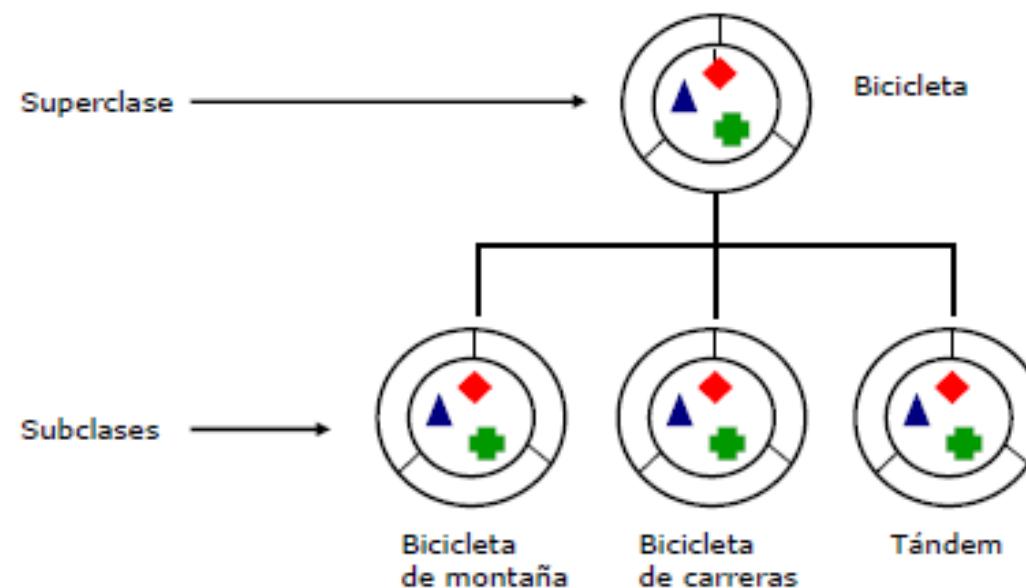


*udg.*

EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Pilares de la POO

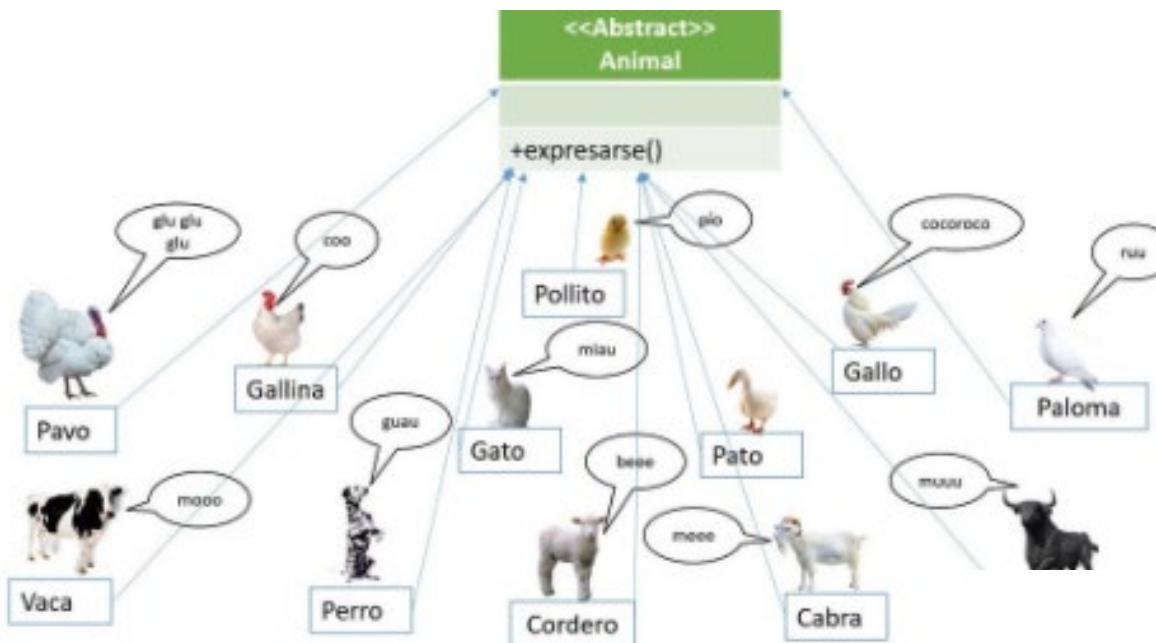
- Herencia
- La herencia es la capacidad de una clase para definirse en términos de otra clase y “heredar” atributos



*udg.*

EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Polimorfismo



## FORMA

- Area
- CalculaArea( )

## CIRCULO

- Area
- CalculaArea( )

## RECTANGULO

- Area
- CalculaArea( )

## CUADRADO

- Area
- CalculaArea( )

# Pilares de la POO

## Polimorfismo

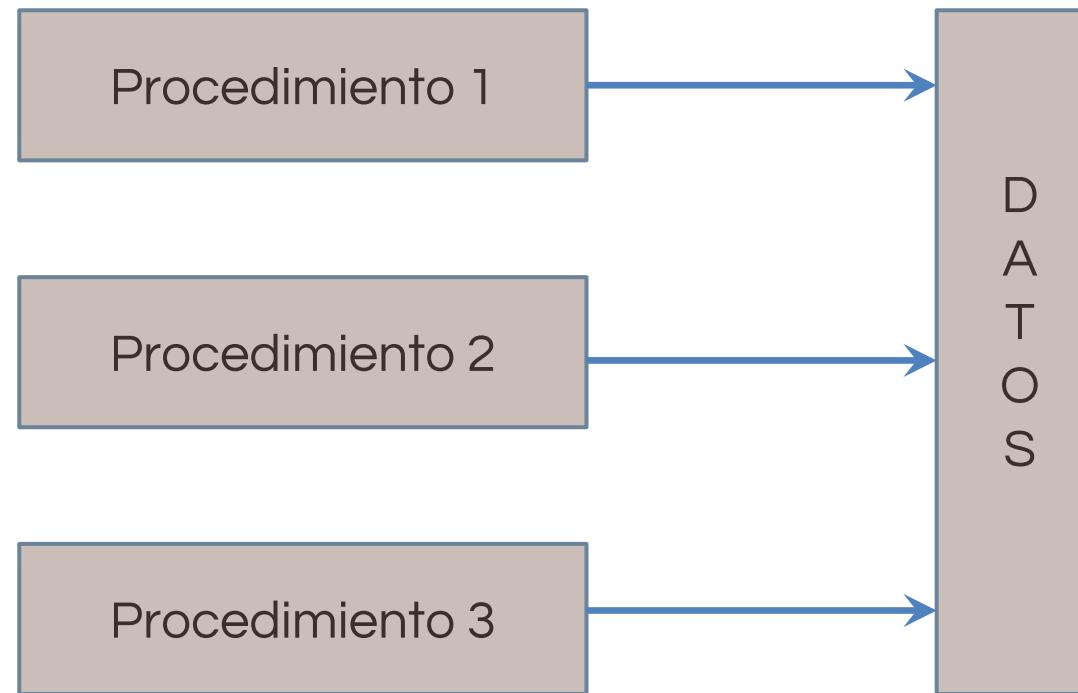
- El polimorfismo permite que distintos objetos pertenecientes a una misma clase “respondan” de diferentes formas a un mismo mensaje
- El polimorfismo permite modificar el comportamiento de un método en cada subclase. En este ejemplo, la superclase mascota tiene las subclases gato, pato y perro. El método “saludar” cada subclase es diferente



**En este ejemplo cada tipo de mascota “saluda” de forma distinta**

# Programación Orientada a Objetos frente a la programación tradicional

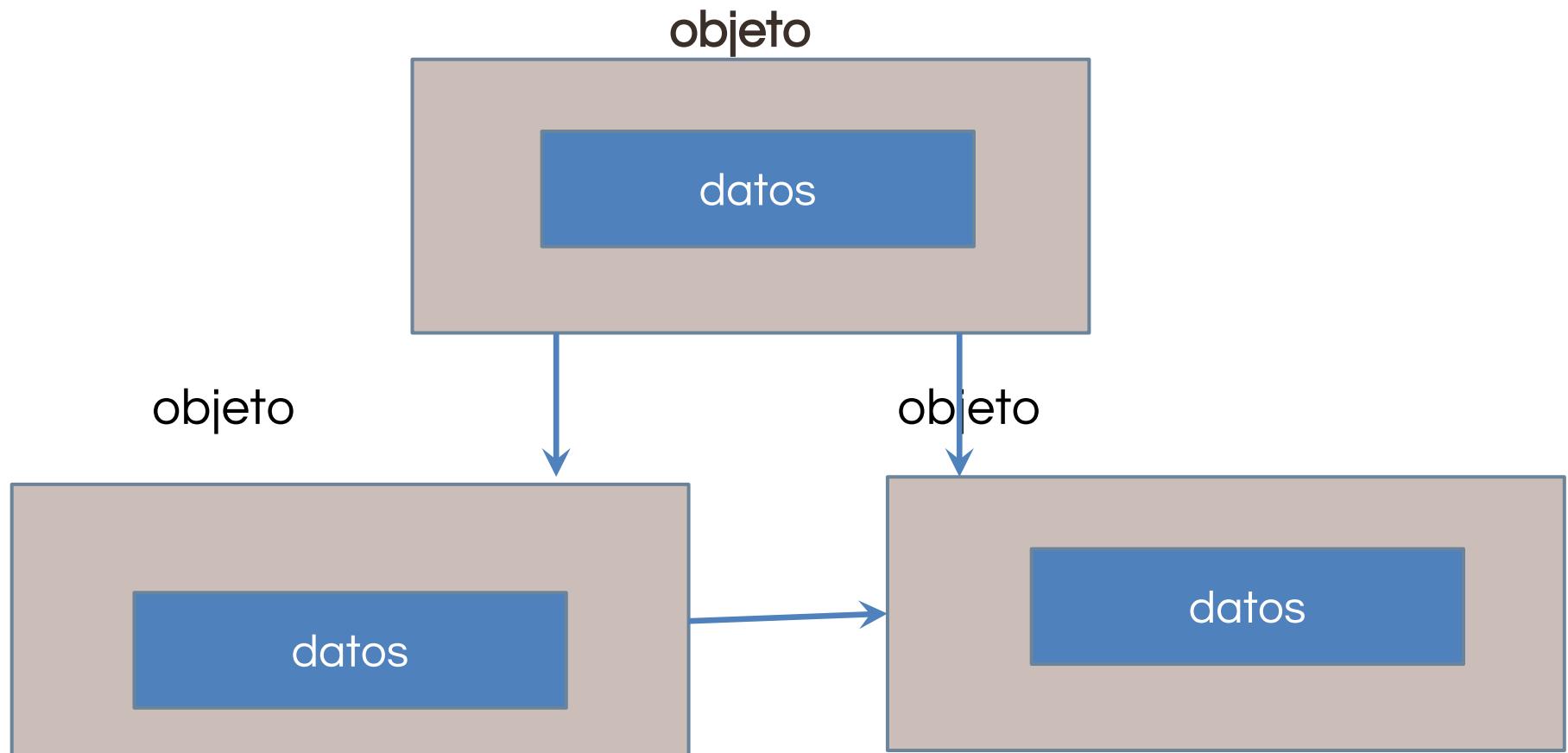
## Programación Tradicional



EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Programación Orientada a Objetos frente a la programación tradicional

## Programación Orientada a Objetos

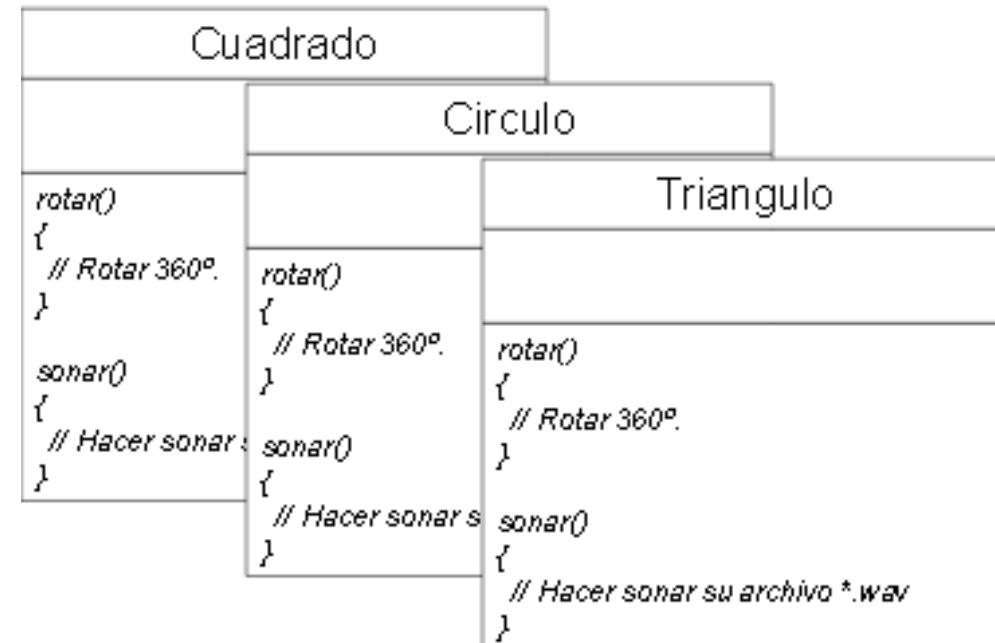


# Programación Orientada a Objetos frente a la programación tradicional

## Estructurada

```
rotar(numFigura){  
    // Dependiendo de  
    numFigura, rotar 360°  
    correctamente.  
}  
  
sonar(numFigura){  
    // Dependiendo de  
    numFigura, buscar el archivo  
    *.wav  
    // correcto y hacerlo sonar.  
}
```

## POO



# Programación Orientada a Objetos frente a la programación tradicional

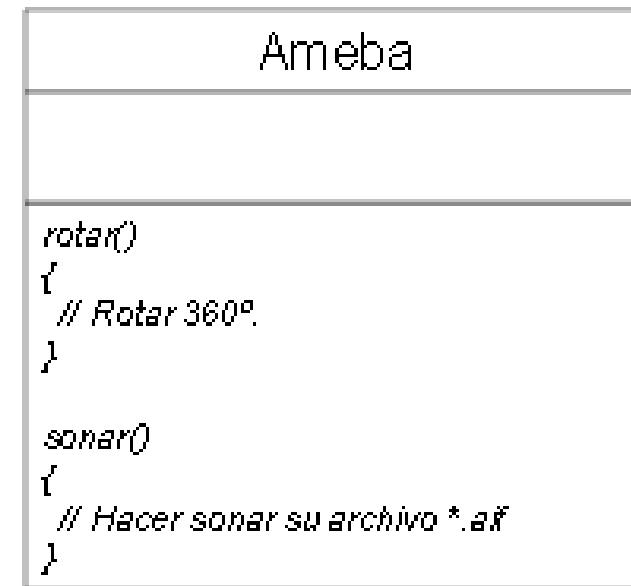
## Estructurada

```
rotar(numFigura){  
    // Dependiendo de  
    numFigura, rotar 360°  
    // correctamente.  
}  
sonar(numFigura){  
    if(es una amearchivoba)  
        // Buscar el *.aif y hacerlo  
        sonar.  
    else
```

```
        // Dependiendo de  
        numFigura, buscar el archivo  
        *.wav correcto y hacerlo  
        sonar.
```

```
}
```

## POO

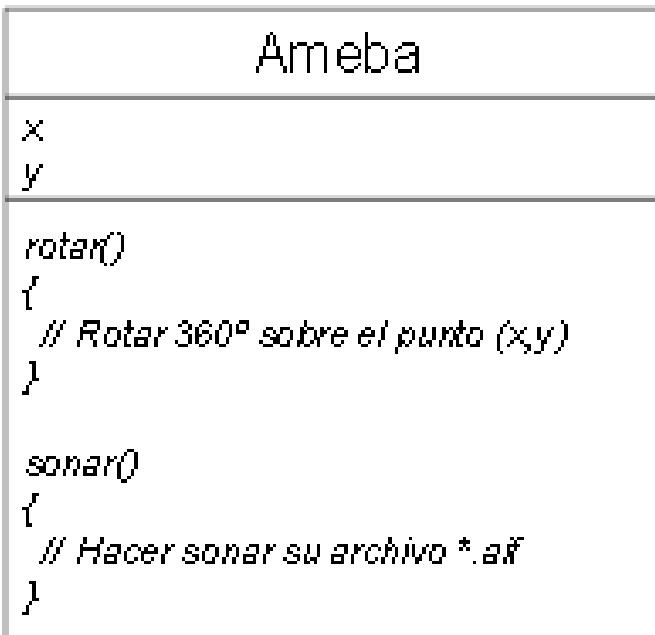


# Programación Orientada a Objetos frente a la programación tradicional

## Estructurada

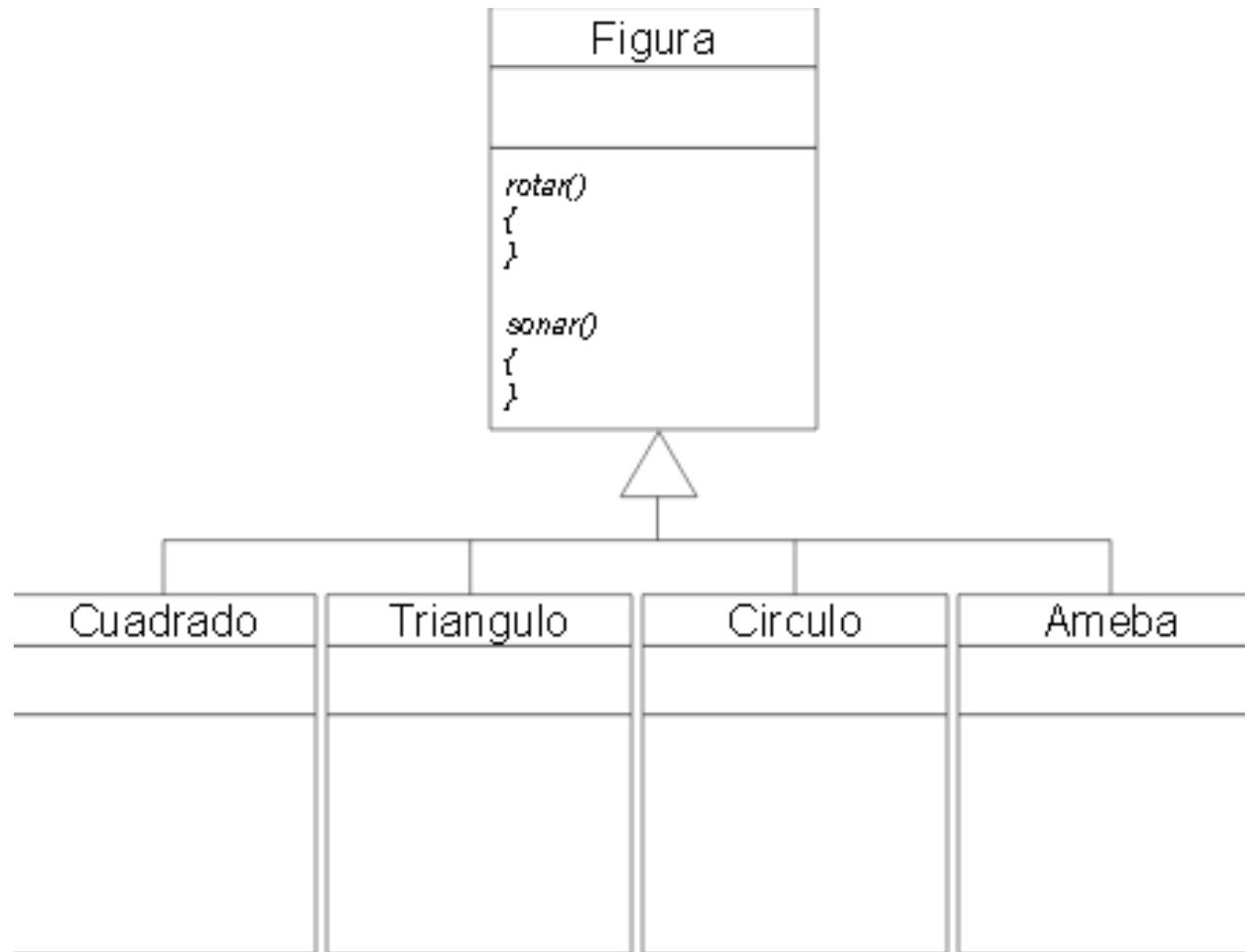
```
rotar(numFigura, x, y){  
    // Si numFigura no es del tipo  
    ameba.  
        // Calcular el punto  
        central basado en un  
        rectángulo,  
            // y rotar 360°.  
    // Si era una ameba.  
        // Usar x e y como el punto  
        de rotación y rotar 360°.  
}
```

## POO

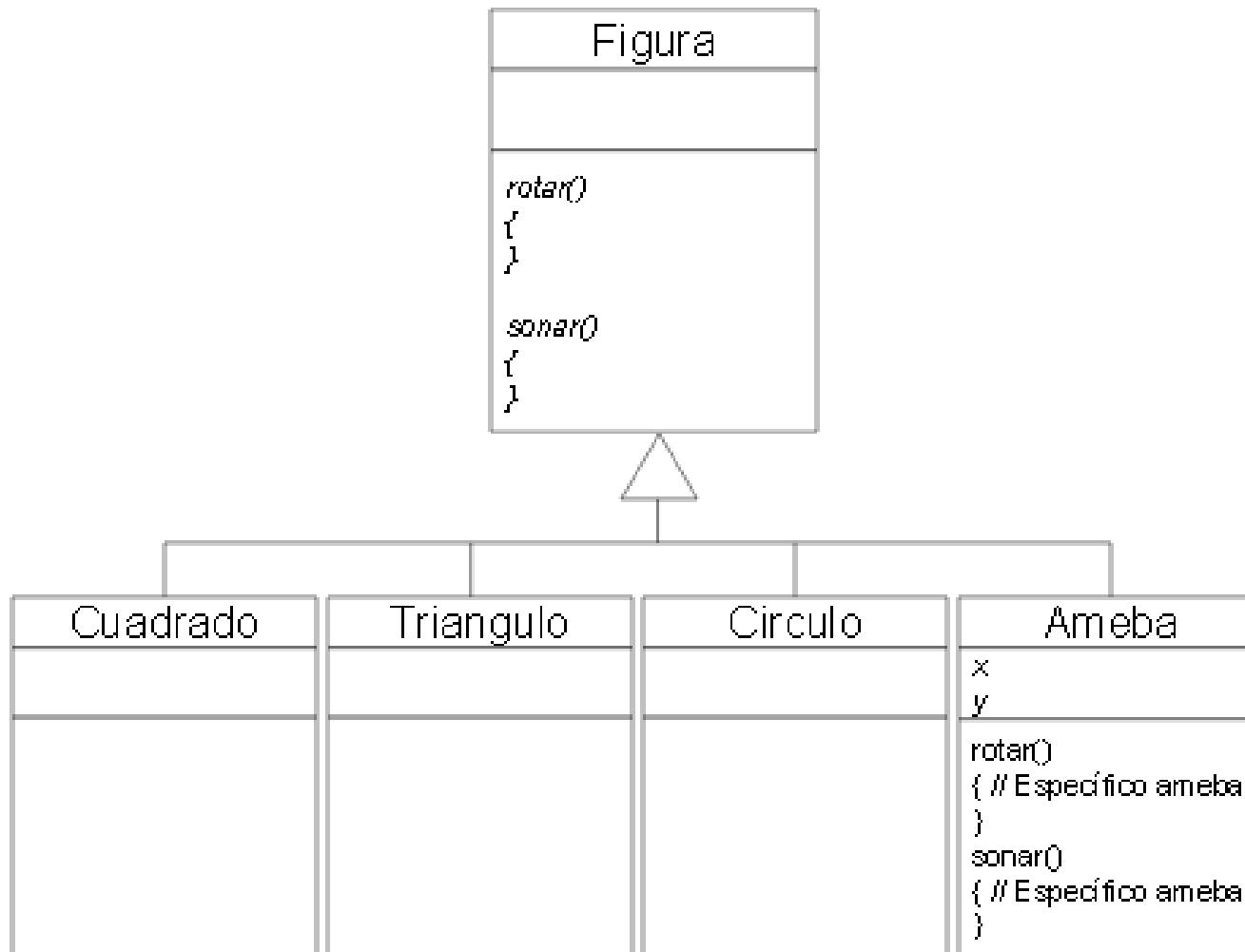


EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE

# Programación Orientada a Objetos frente a la programación tradicional



# Programación Orientada a Objetos frente a la programación tradicional



# ¿Por qué Orientación a Objetos (OO)?

- Fomenta la reutilización y extensión del código.
- Construcción de prototipos.
- Agiliza el desarrollo de software.
- Facilita el trabajo en equipo.
- Facilita el mantenimiento del software.
- POO proporciona conceptos y herramientas con las cuales se modela y representa el mundo real.

# ¿Por qué Orientación a Objetos (OO)?

- Las aplicaciones son más sencillas para los usuarios debido a que los datos innecesarios están ocultos.
- La productividad se incrementa debido a que puede reutilizar el código.
- Los sistemas son fáciles de mantener y se adaptan a las cambiantes necesidades de negocios.

The logo consists of a stylized, handwritten-style lowercase 'u' and 'd' followed by a diagonal slash and a lowercase 'l'.

*udl*

---

**EL MUNDO  
NECESITA GENTE  
QUE AME  
LO QUE HACE**



UNIVERSIDAD DE LAS AMÉRICAS © 2016