Programación II

Bernarda Sandoval

Tema: Diseño de clases

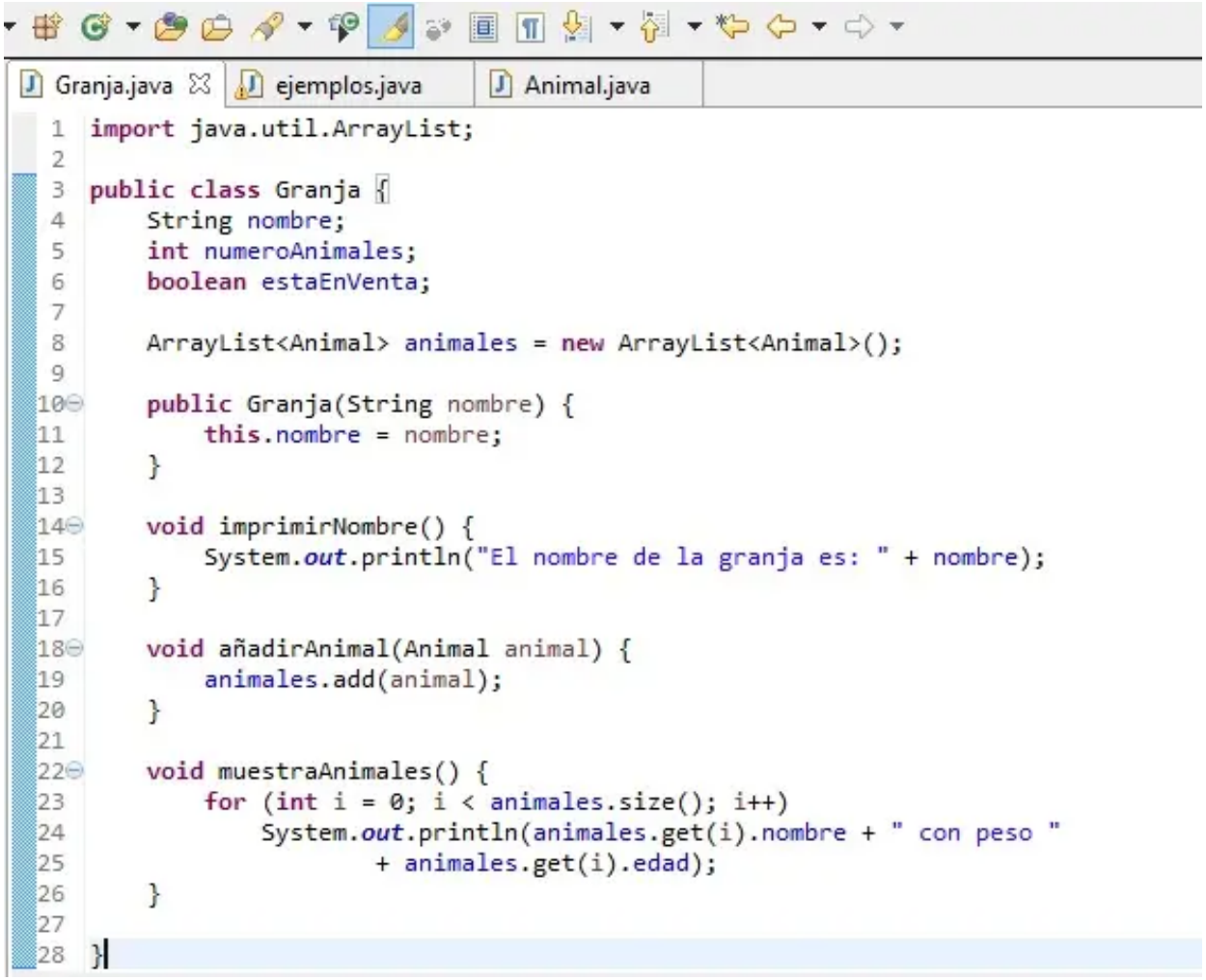
RDA

- 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



EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

Programación orientada a objetos

Granja.java | ejemplos.java | Animal.java

```java
import java.util.ArrayList;

public class Granja {
    String nombre;
    int numeroAnimales;
    boolean estaEnVenta;

    ArrayList<Animal> animales = new ArrayList<Animal>();

    public Granja(String nombre) {
        this.nombre = nombre;
    }
```

```java
package clases;
/**
 * @author Pablo Ruiz Soria
 */
public class Mesa {
    private String color;

    /**
     * @return the color
     */
    public String getColor() {
        return color;
    }

    /**
     * @param color the color to set
     */
```

```java
/**
 * @author Pablo Ruiz Soria
 */
public class Coche {
    String marca;
    String modelo;
    Motor motor;

    public Coche(){

    }

    public Coche(String marca, String modelo, Motor motor){
        this.marca = marca;
        this.modelo = modelo;
        this.motor = motor;
    }
}
```

```java
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

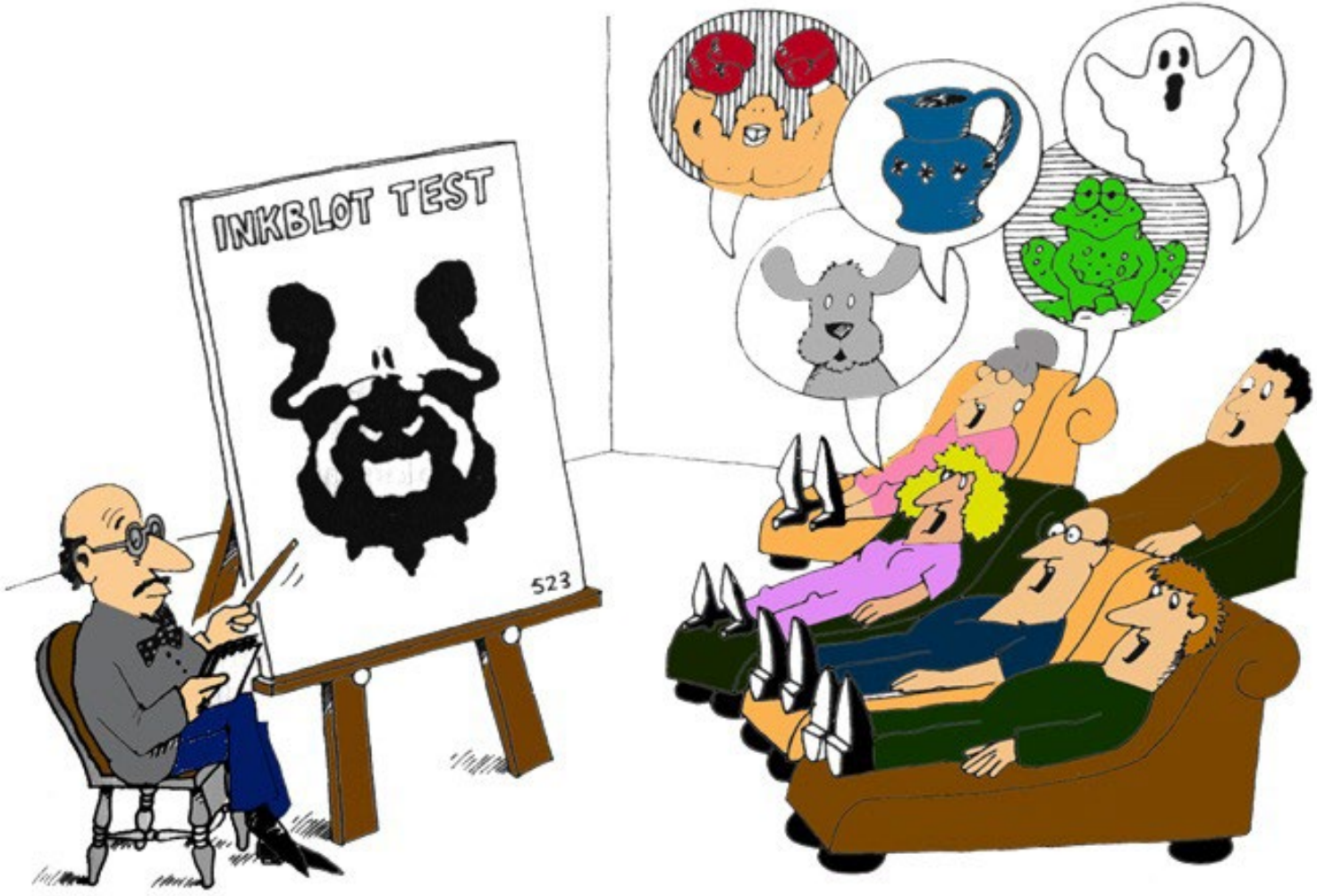
```c
#include<stdio.h>
int main(){
int i,n;
float calif,suma,prom;

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;
}
```

```java
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

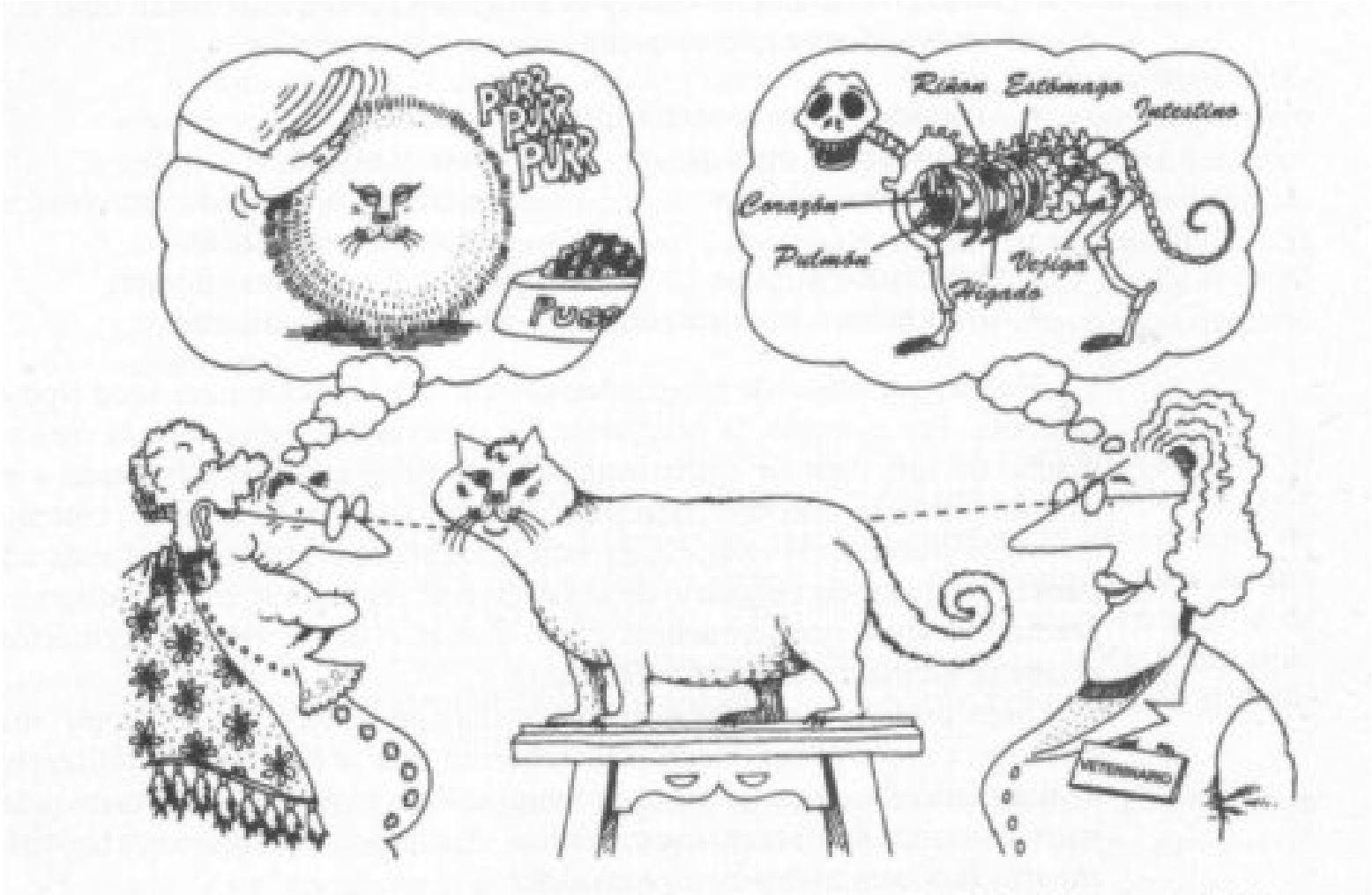
Pilares de POO

Abstracción



EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

Abstracción



EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

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.

Pilares de la POO

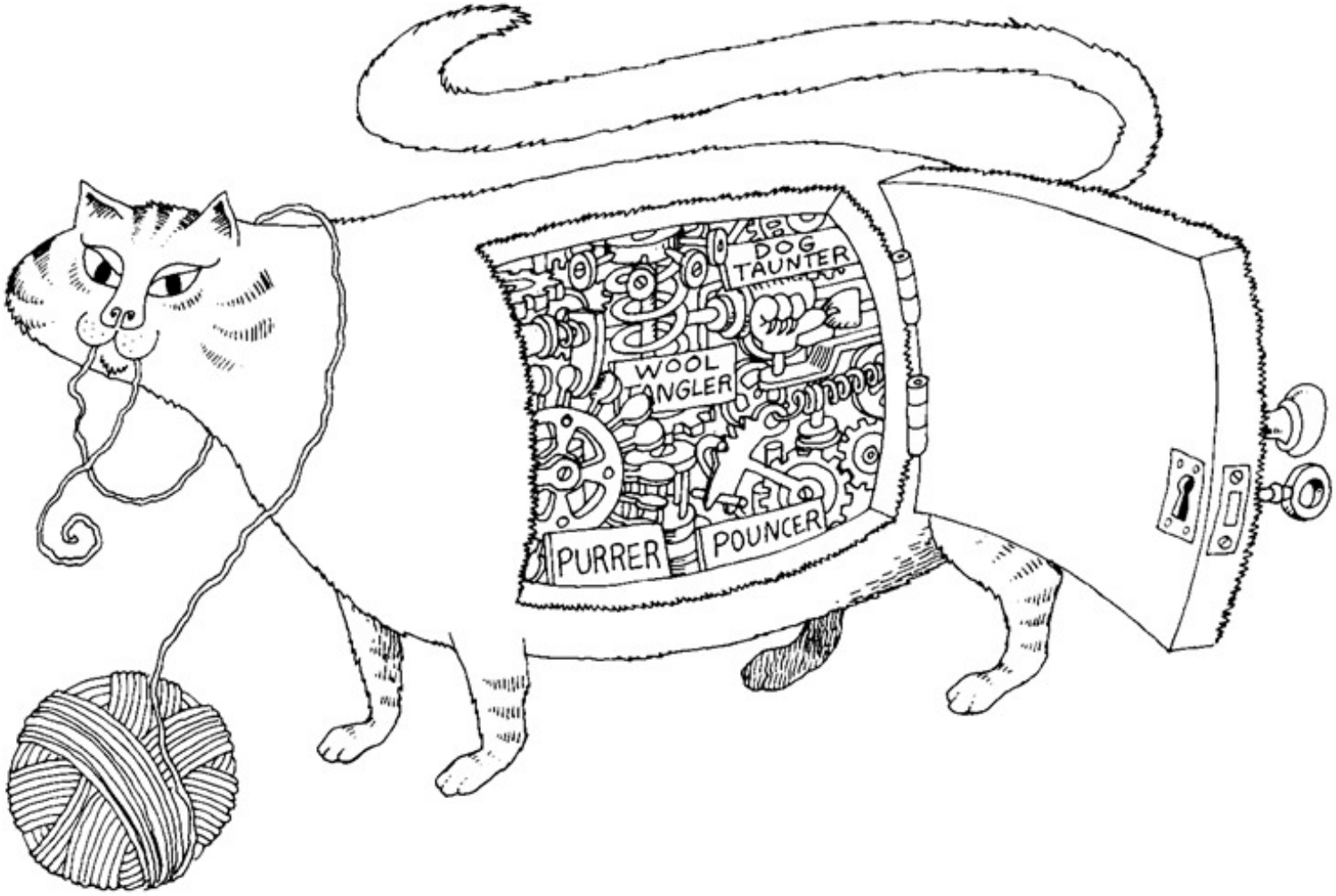
Abstracción

- ¿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

Ejemplos de carros

Encapsulamiento



EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

Encapsulamiento

Pilares de la POO

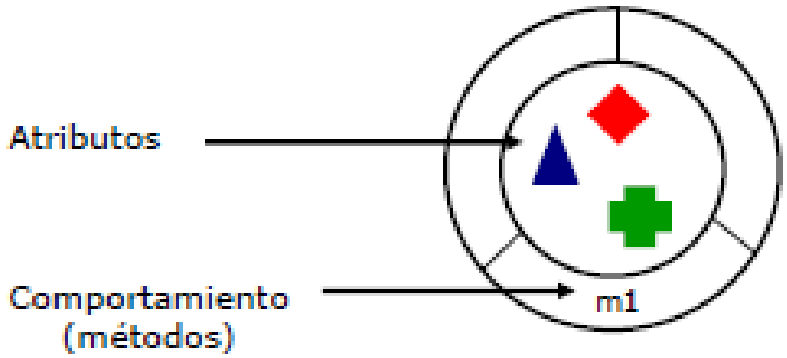
Encapsulación

- 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

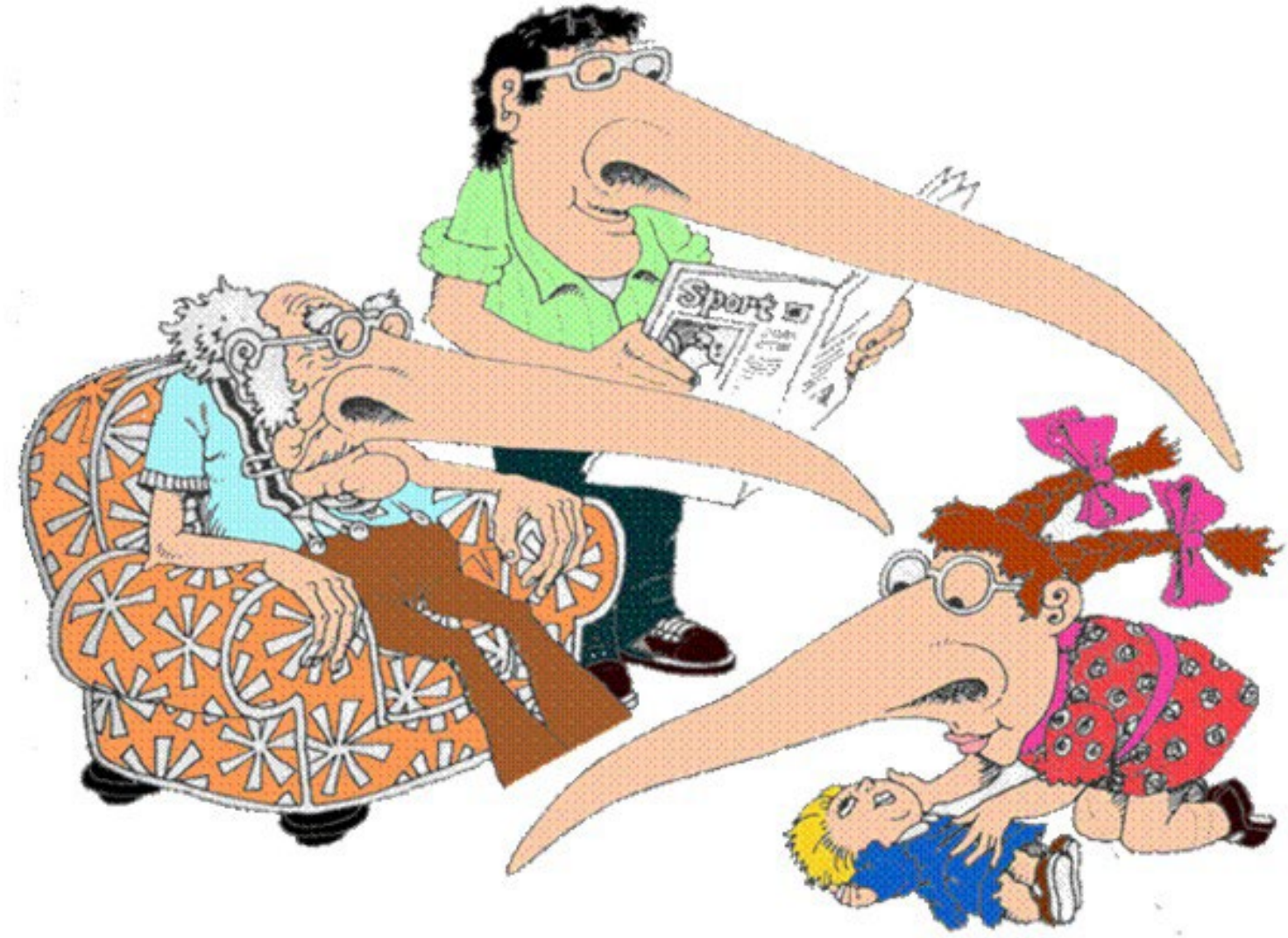
Pilares de la POO

Encapsulación

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

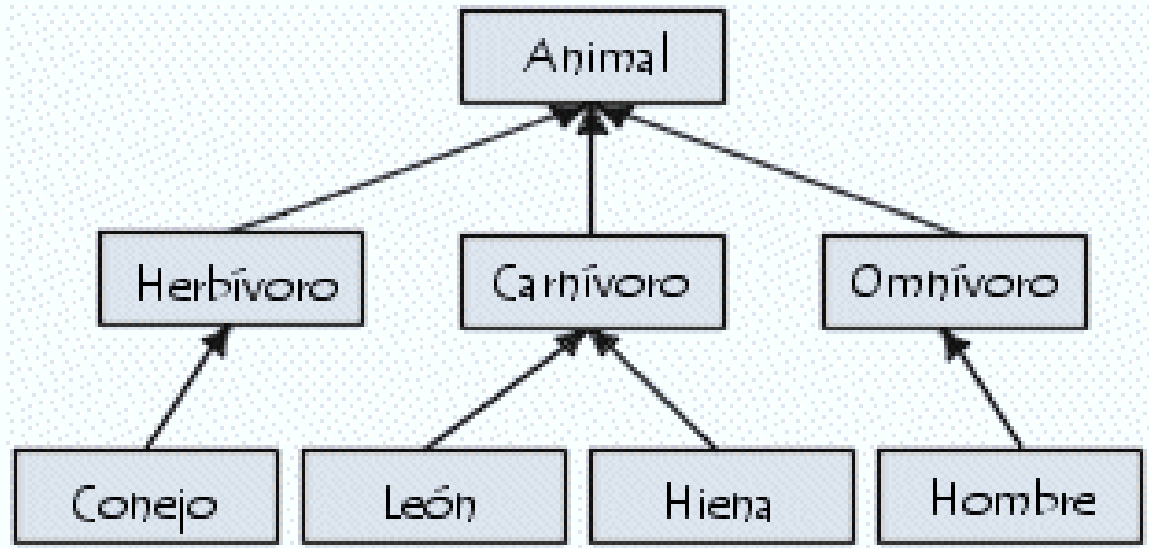


Herencia



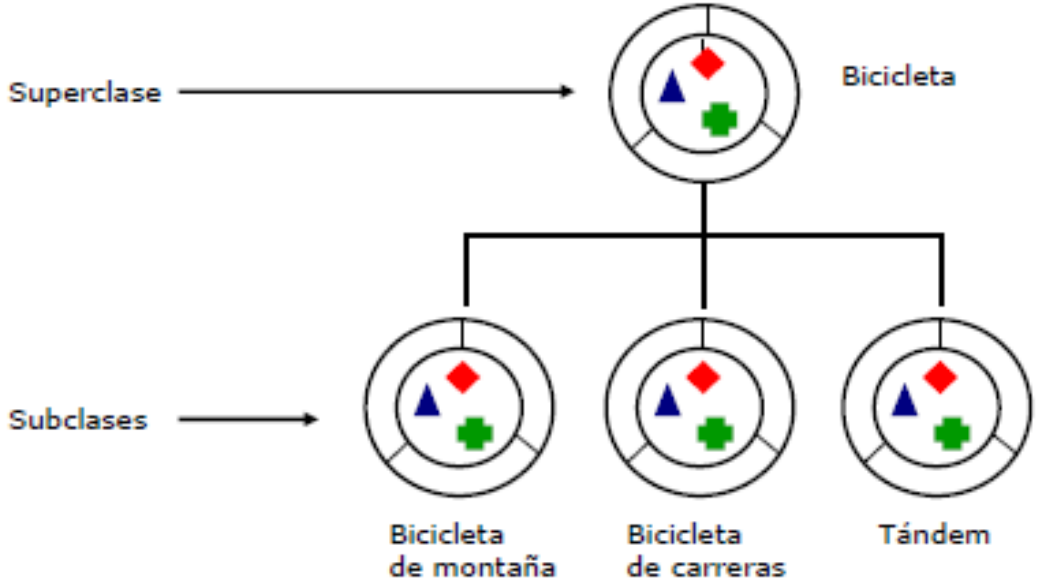
EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

Herencia

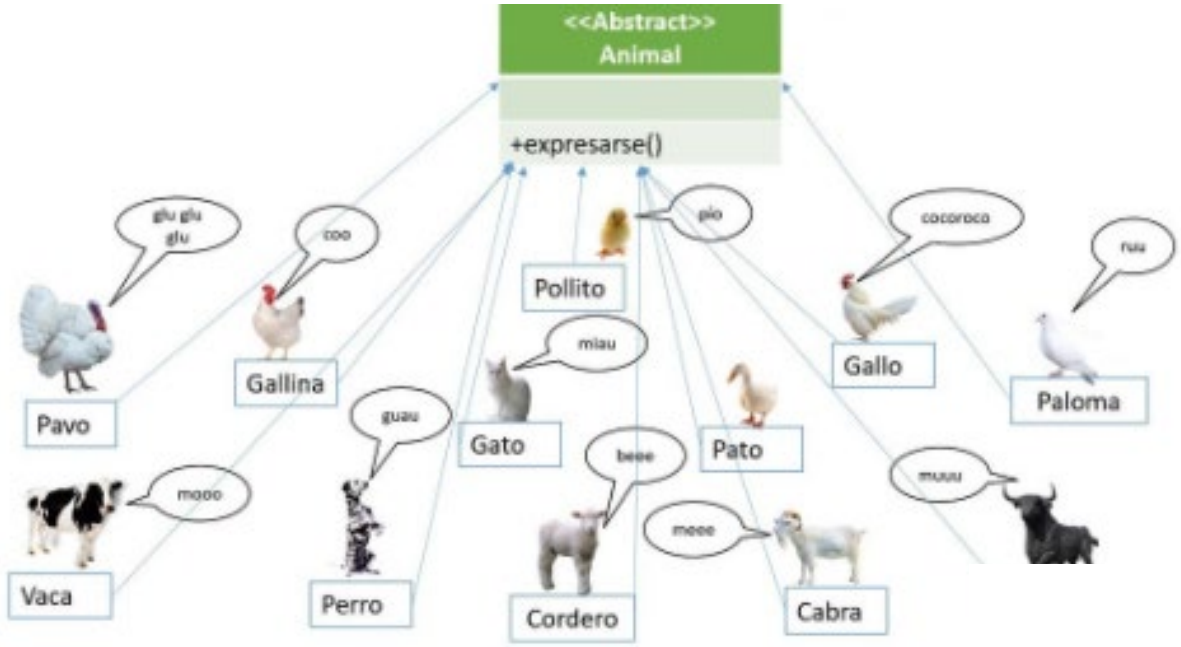


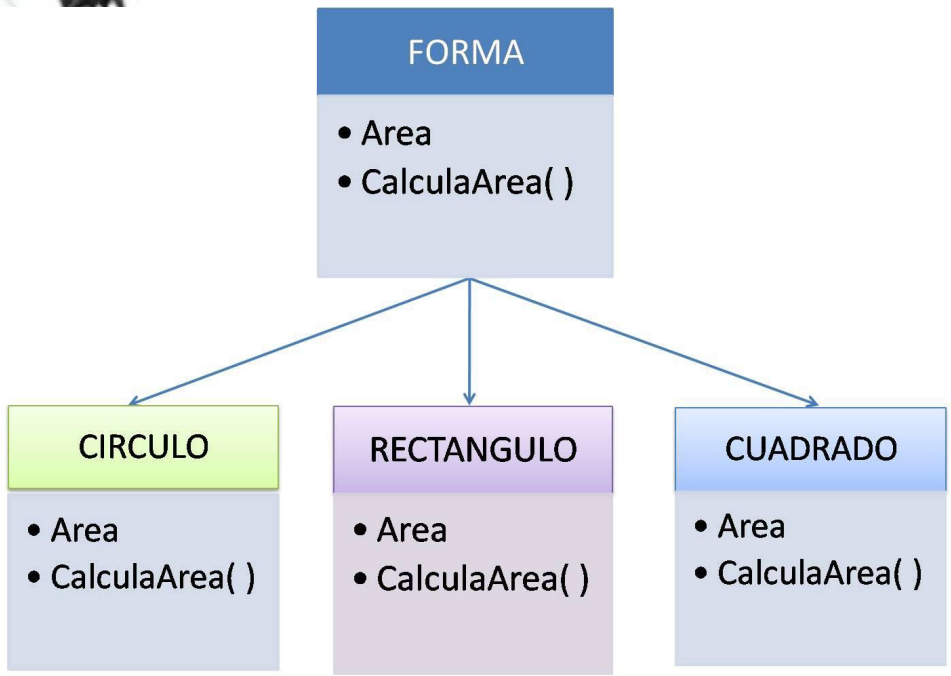
Pilares de la POO

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



Polimorfismo



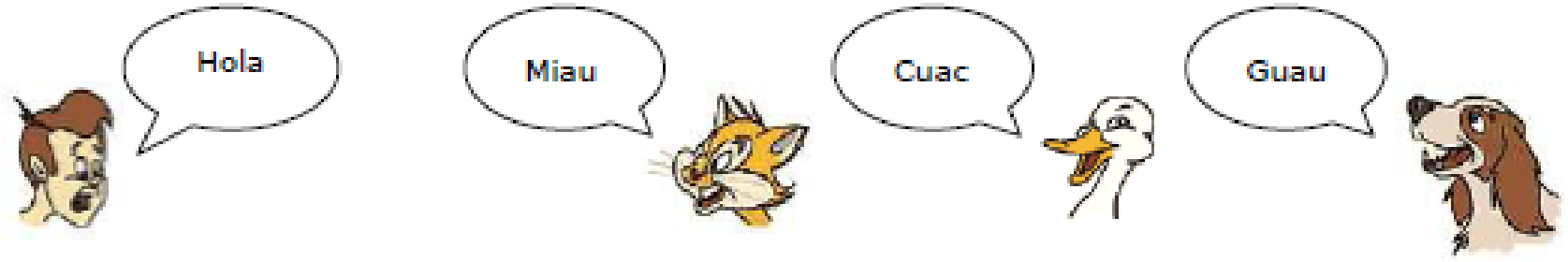


EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

Pilares de la POO

Polimorfismo

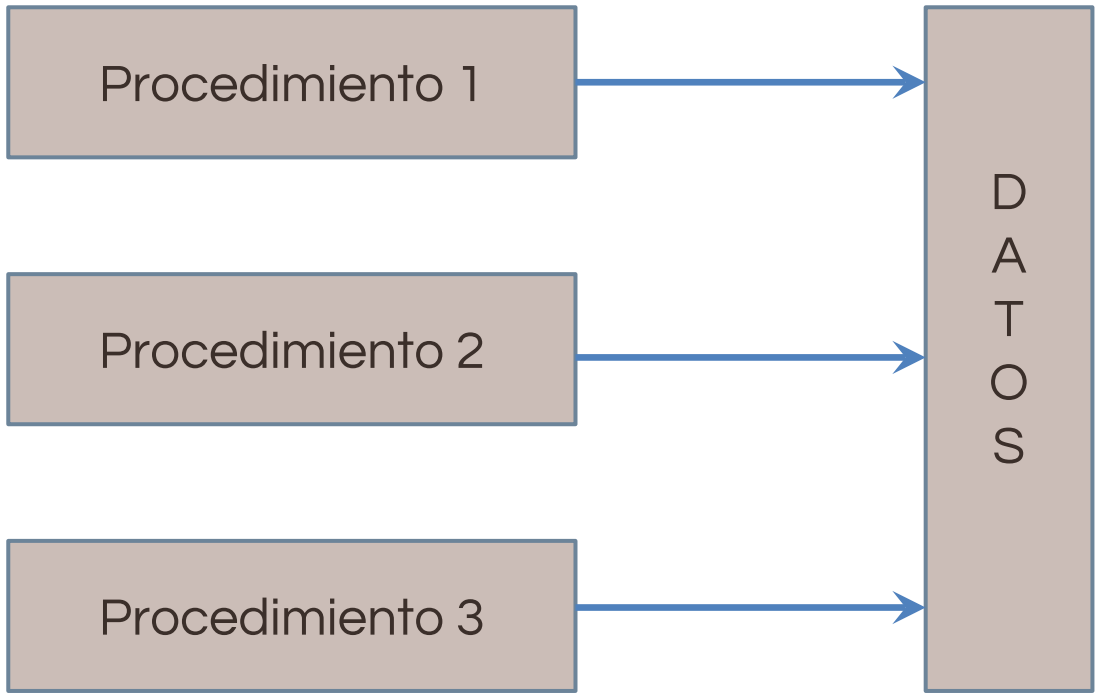
- El polimorfismo permite que distintos objetos pertenecientes a una misma clase “respondan” de diferentes formas a un mismo mensaje
- El polimorfismo permite modificar el comportamiento de un método en cada subclase. En este ejemplo, la superclase mascota tiene las subclases gato, pato y perro. El método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



En este ejemplo cada tipo de mascota "saluda" de forma distinta

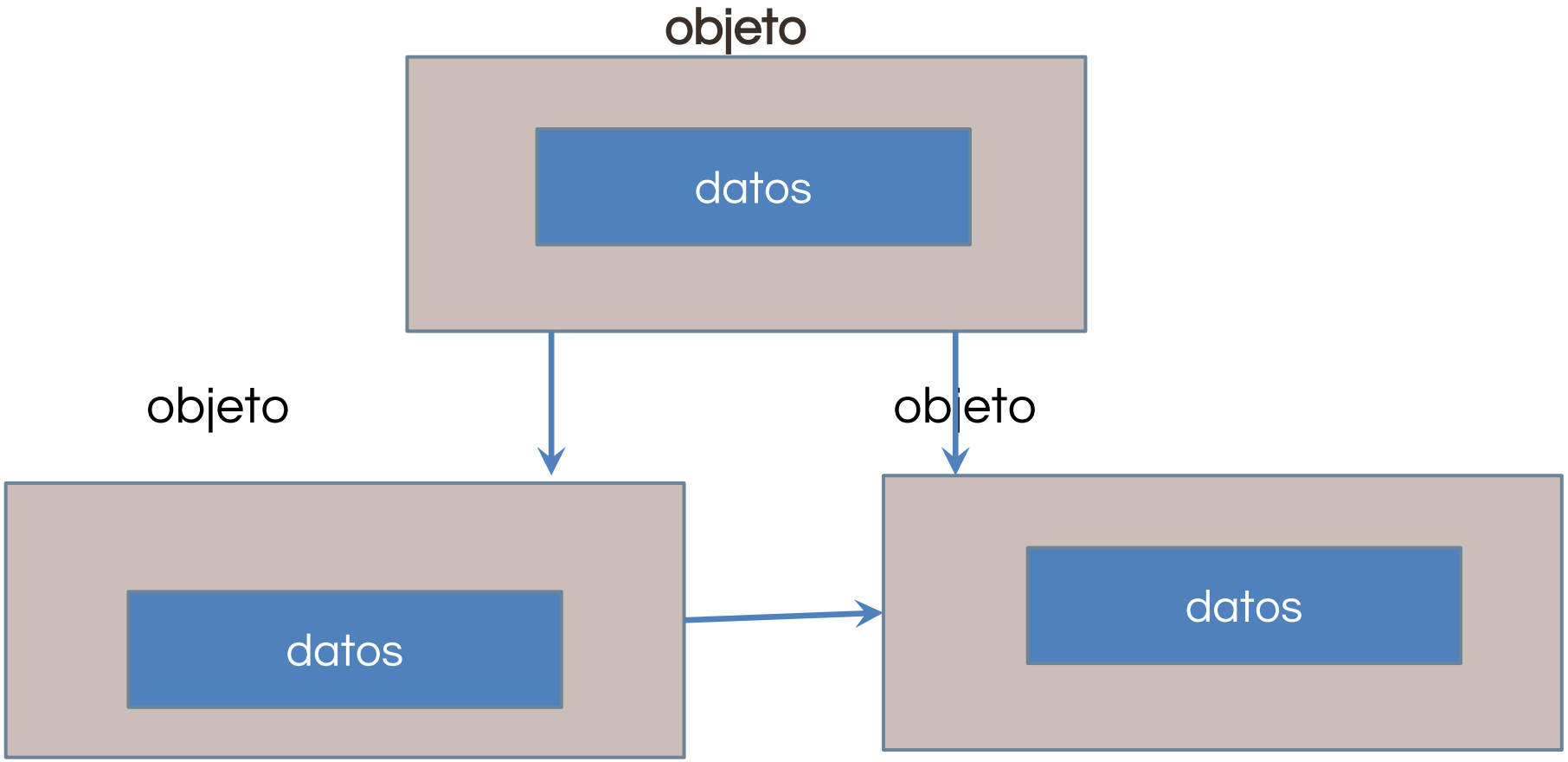
Programación Orientada a Objetos frente a la programación tradicional

Programación Tradicional



Programación Orientada a Objetos frente a la programación tradicional

Programación Orientada a Objetos



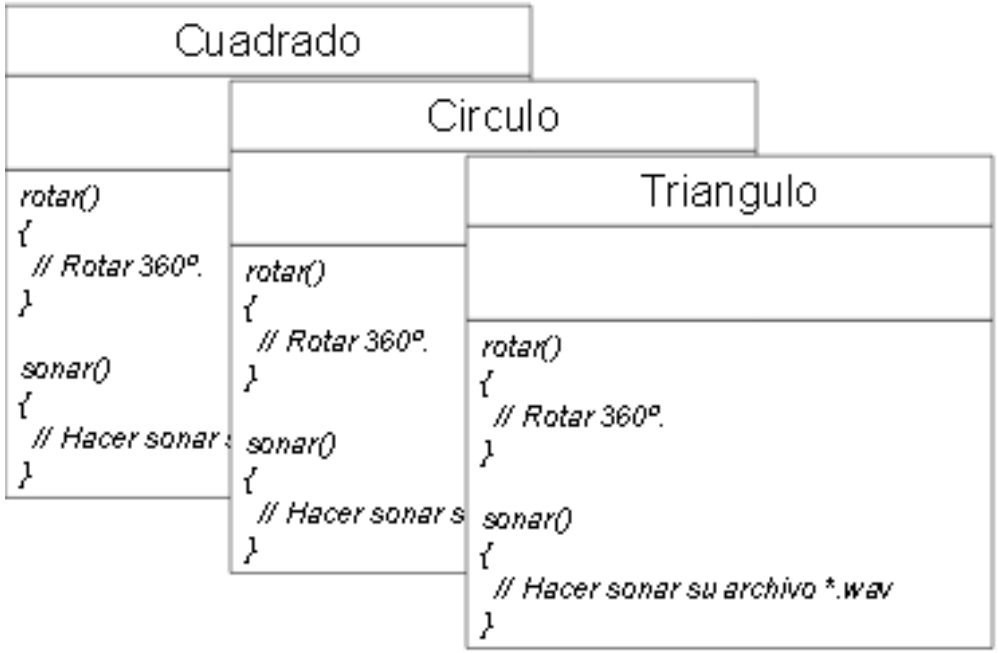
EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

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



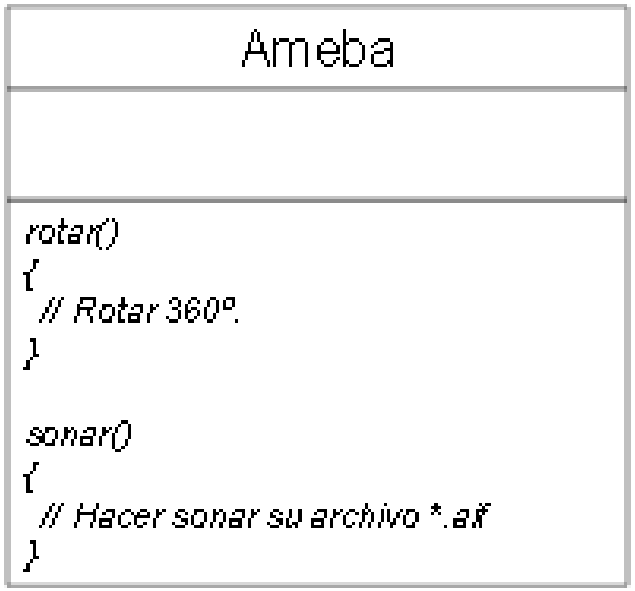
EL MUNDO NECESITA GENTE QUE AME LO QUE HACE

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

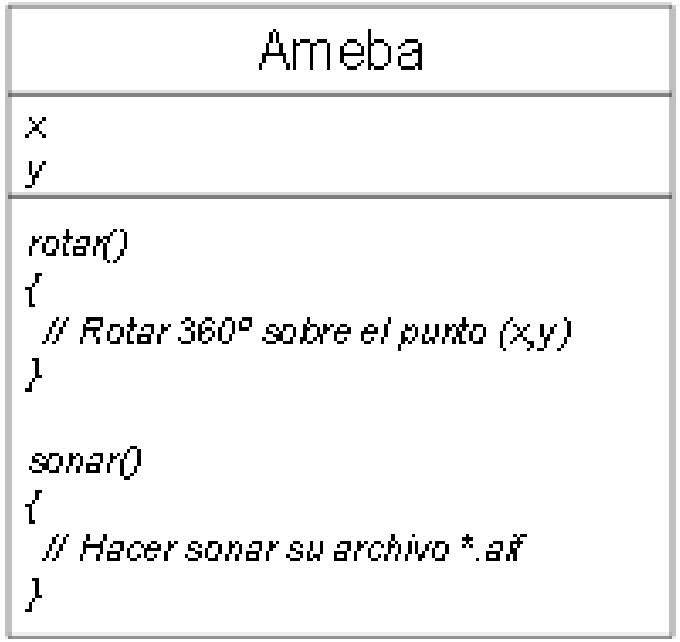


Programación Orientada a Objetos frente a la programación tradicional

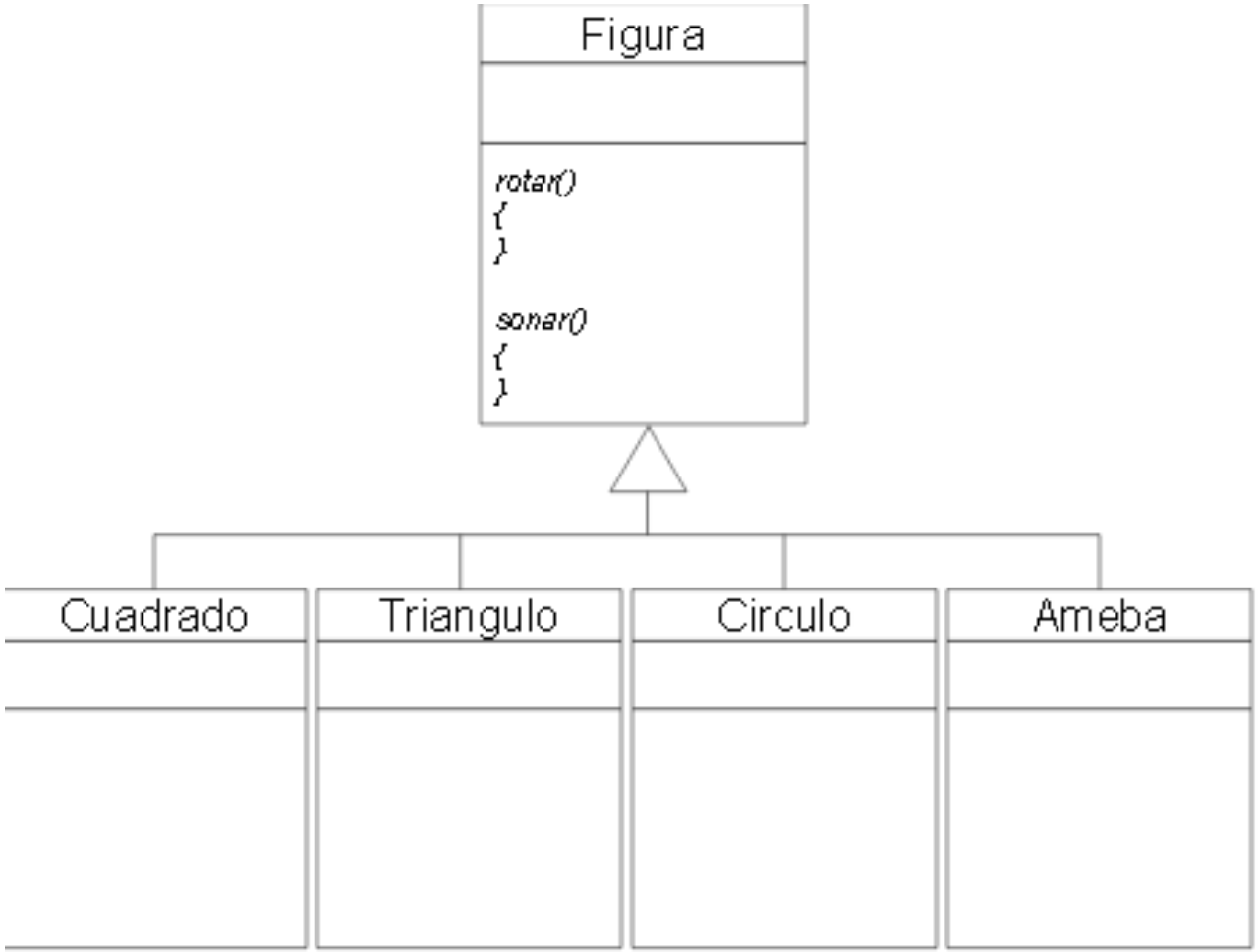
Estructurada

```
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°.
}
```

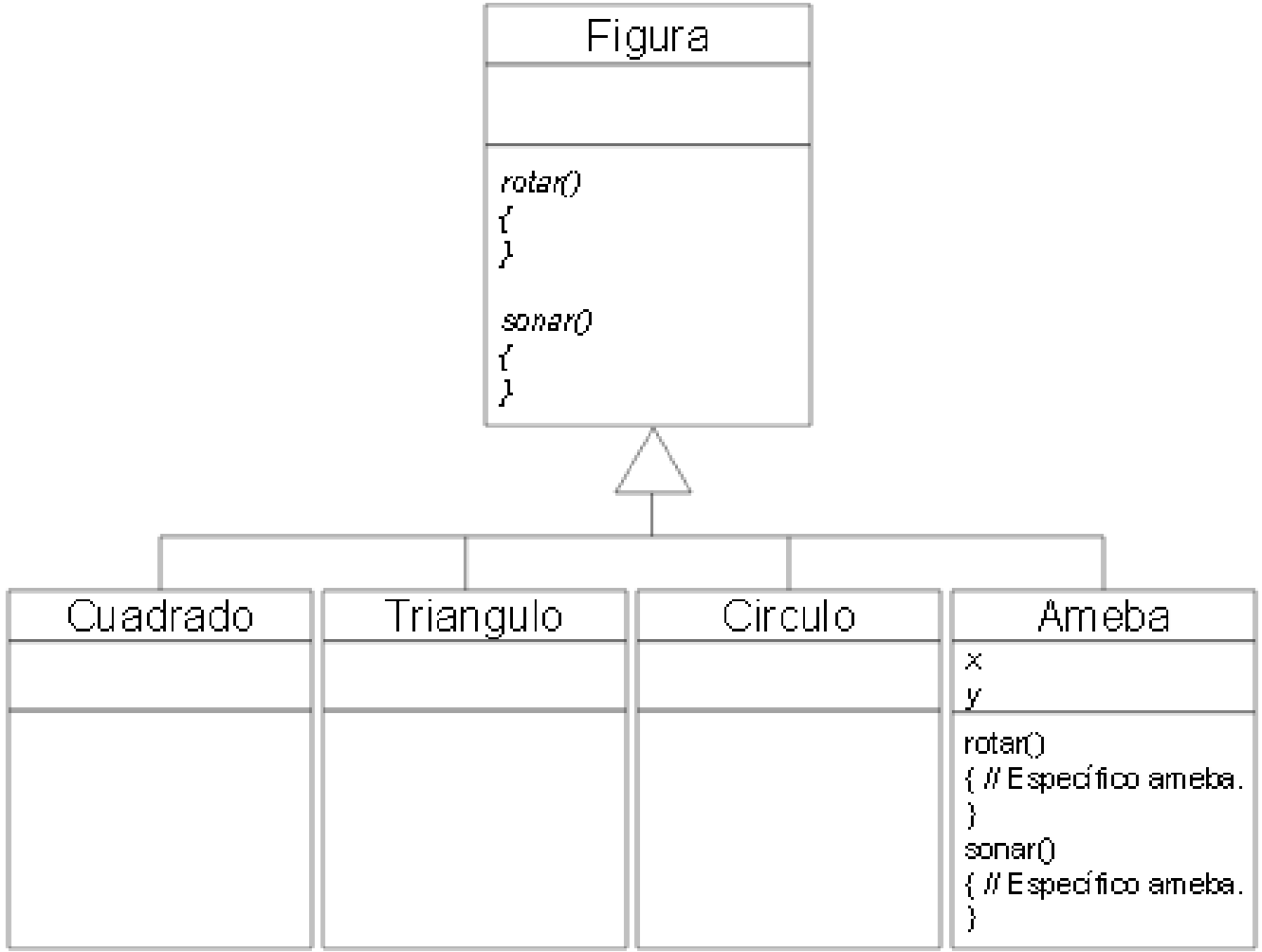
POO



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.

EL MUNDO
NECESITA GENTE
QUE AME
LO QUE HACE