UD2.- Introducción a la Programación Orientada a Objetos (POO)

Módulo: Programación
1.º DAM

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Programación estructurada versus POO

- Programación estructurada
  - Datos
  - Programas y subprogramas (funciones y procedimientos)
- POO
  - Objetos
    - Agrupación de **datos** (atributos o propiedades) y **métodos** (funciones y procedimientos).

¿Qué es una clase?

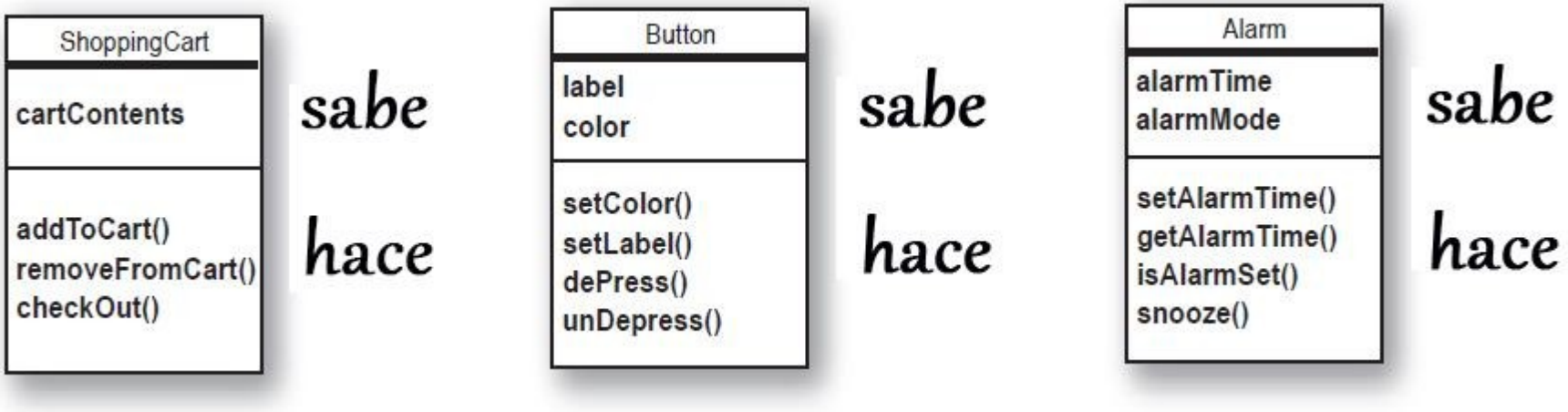
- Plantilla o estructura preliminar que describe un objeto y define sus características (atributos) y operaciones (métodos).

  - **Atributos**: son los datos que tendrán los objetos de la clase. Una clase puede contener un conjunto o ninguno. Se declaran con un nombre y un tipo de datos.

  - **Métodos**: son las operaciones que podrán realizar los objetos de la clase

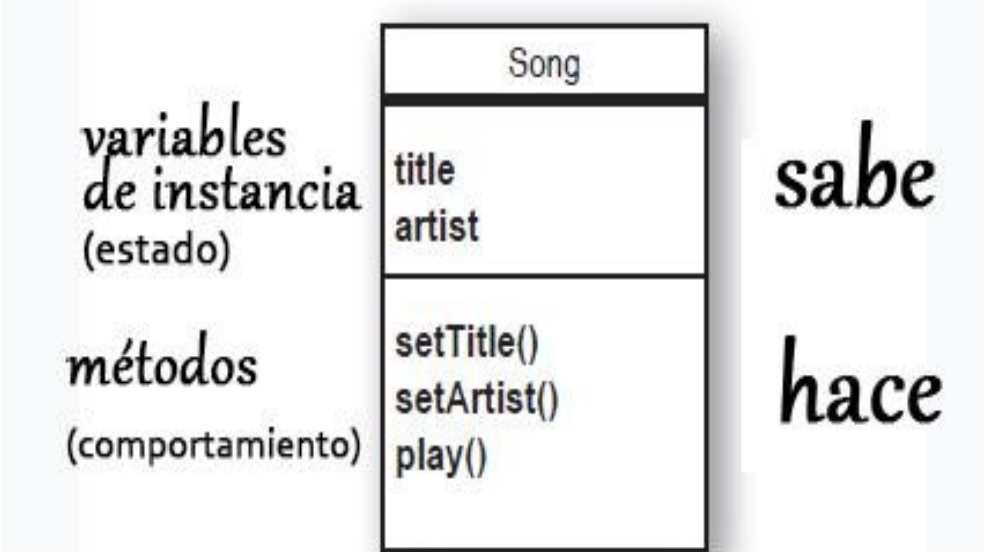
Programación Orientada a Objetos

- Cuando se diseña una clase, se piensa en los objetos de la clase que se crearán:
  - Aquello que el objeto sabe
  - Aquello que el objeto hace



Programación Orientada a Objetos

- Aquello que el objeto sabe → atributos o propiedades o variables de instancia
- Aquello que el objeto hace → métodos



Ejemplo de clase

```
class Alumno{
    String nombre;
    int edad;

    void setNombre(String n);
    void setEdad(int e);
    String getNombre();
    int getEdad();
    void visualizarDatos();
}
```

ATRIBUTOS

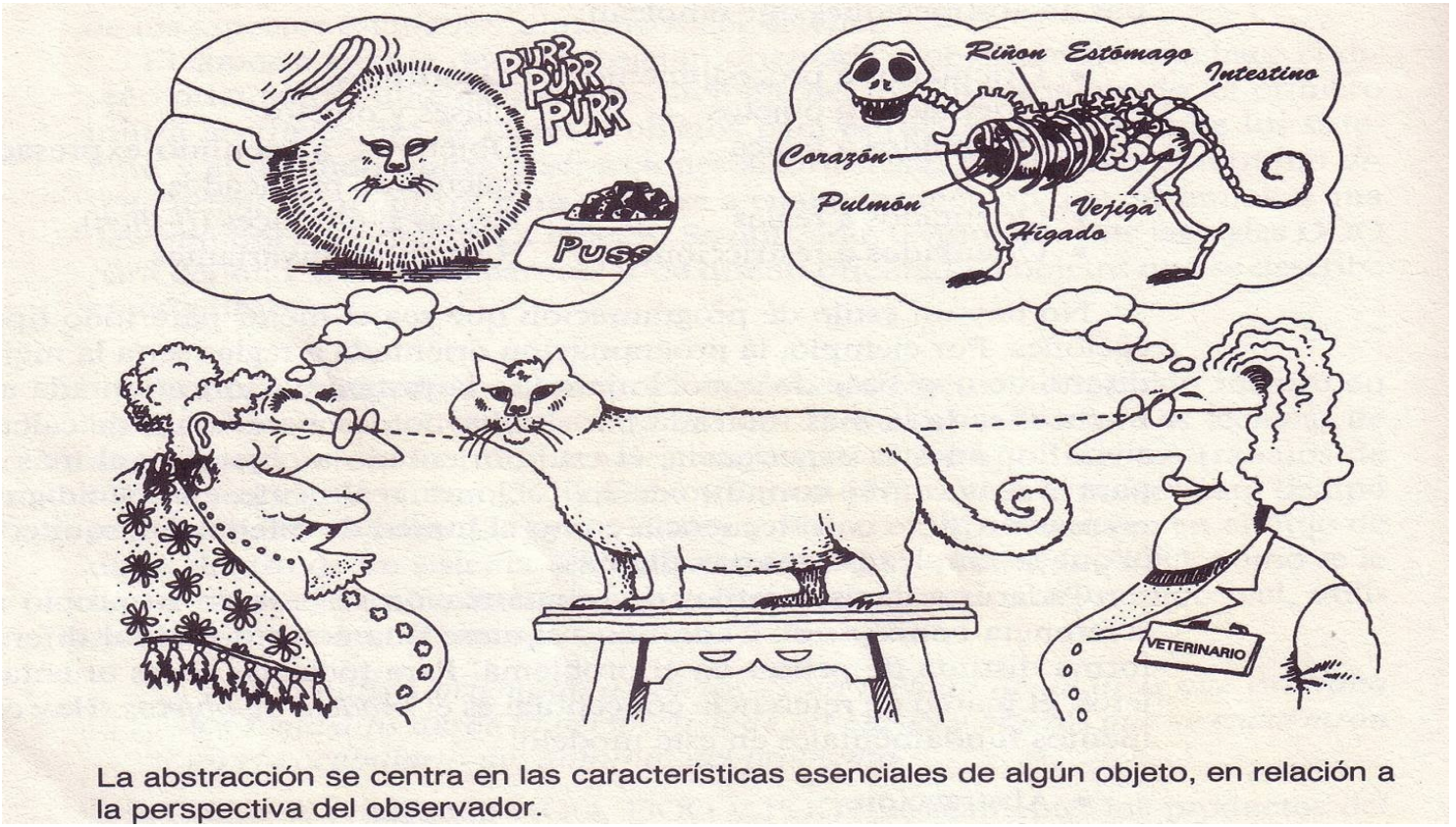
MÉTODOS

Principios básicos de la OO

- Abstracción
- Encapsulamiento
- Herencia
- Polimorfismo

Abstracción

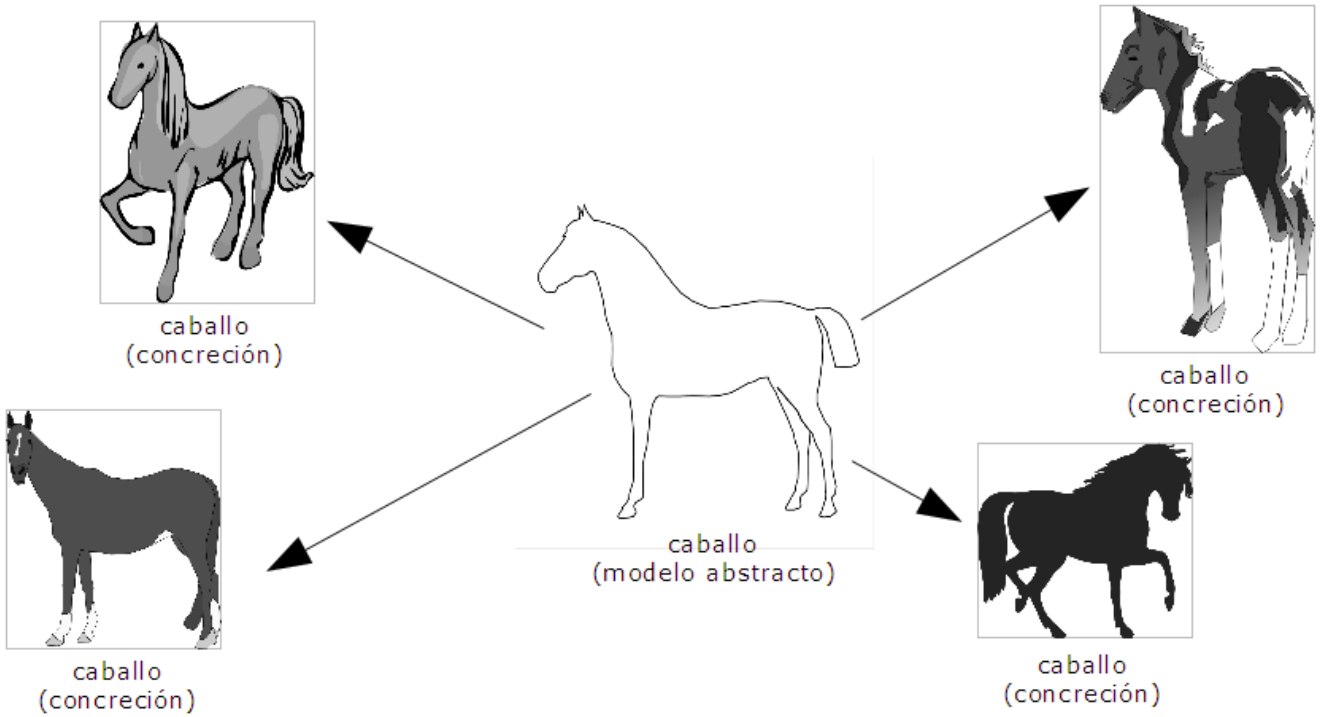
Es el mecanismo para determinar los tipos de clases, agrupando datos y funciones. No se entran en detalles de la representación interna.



La abstracción se centra en las características esenciales de algún objeto, en relación a la perspectiva del observador.

Abstracción

Ejemplo: tenemos caballos de diferente raza. Su aspecto exterior es muy diferente, pero sabemos que todos pertenecen a la clase Caballo porque realizamos una abstracción o identificación mental de los elementos comunes que tienen (cola, cuatro pates, son rápidos...).



Encapsulamiento

Los objetos serán “cajas negras”: sabemos **qué hace** a través de su interfaz **pero no** sabemos **como** lo hace.

Encapsulamiento

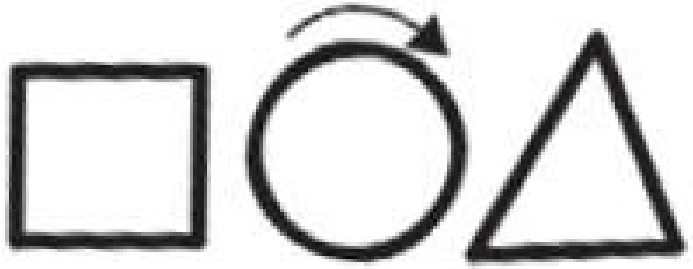
Ejemplo: un terminal de autoservicio es sencillo de utilizar para el usuario. Se ocultan los detalles de implementación interna (que es compleja).

¿Qué es una clase?

Imaginamos un problema...

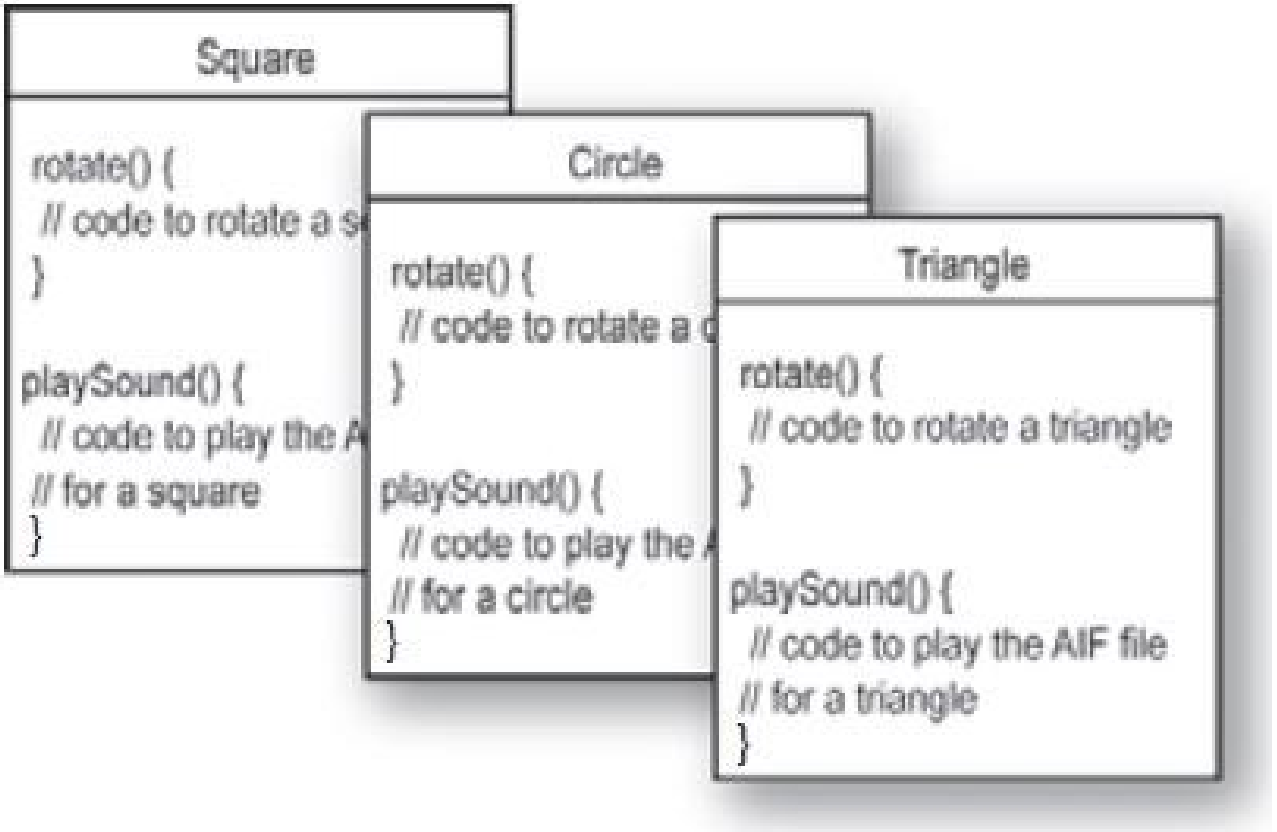
La especificación del problema

Tendremos formas: un cuadrado, una esfera y un triángulo. Cuando el usuario pulse en una forma, la forma debe rotar 360º en sentido horario y debe sonar un sonido característico de esa forma en concreto.



¿Qué es una clase?

Creamos una clase para cada forma...



¿Qué es una clase?

Necesitamos añadir una nueva forma...

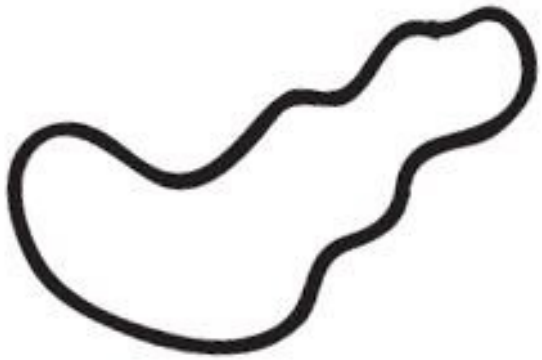
Amoeba

```
rotate() {
  // code to rotate an amoeba
}

playSound() {
  // code to play the new
  // .hif file for an amoeba
}
```

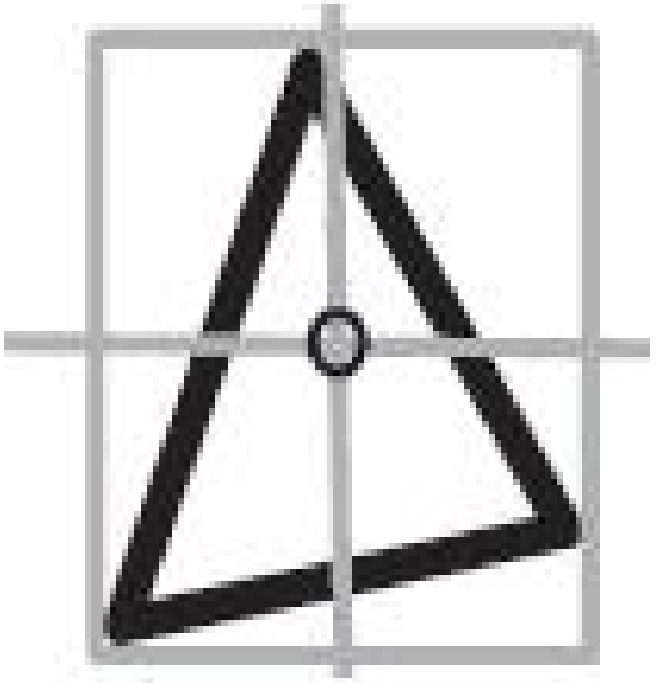
← Añadimos a la especificación

Tendremos una forma nueva: una ameba. Cuando el usuario pulse en la ameba, la forma rotará como las otras formas y hará sonar un sonido específico para la ameba.



Método rotate()

- Pasos para rotar una forma:
  - Determinar el rectángulo que rodea la forma
  - Calcular el centro de la figura y rotar la figura sobre este punto.



Pero la ameba no rota igual...

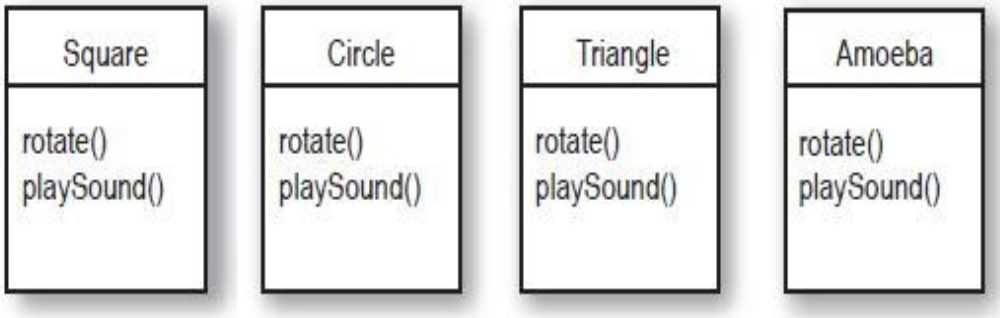
Lo que la especificación olvidó mencionar

Amoeba

```
int xPoint
int yPoint
rotate() {
  // code to rotate an amoeba
  // using amoeba's x and y
}
playSound() {
  // code to play the new
 // .hif file for an amoeba
}
```

No sirve el mismo método para rotar la ameba, se tendrán que definir las coordenadas.

Clases y características comunes

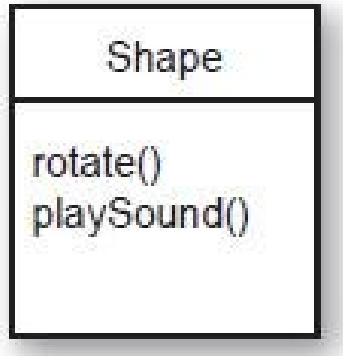


1

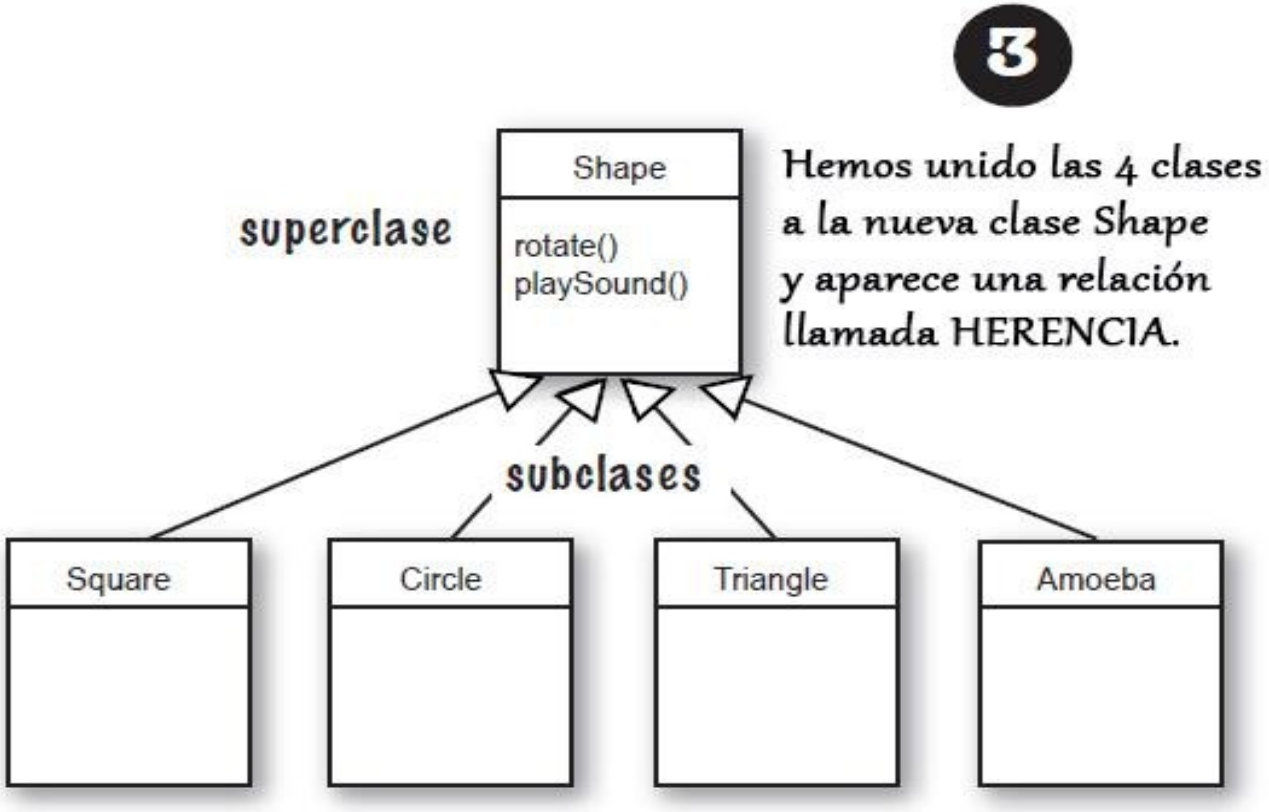
Tenemos 4 clases.
¿Qué tienen en común entre ellas?

2

Todas ellas son formas, y tienen que rotar y hacer sonar una melodía. Por eso, resumimos las características comunes y las ponemos en una nueva clase llamada 'shape' (forma).

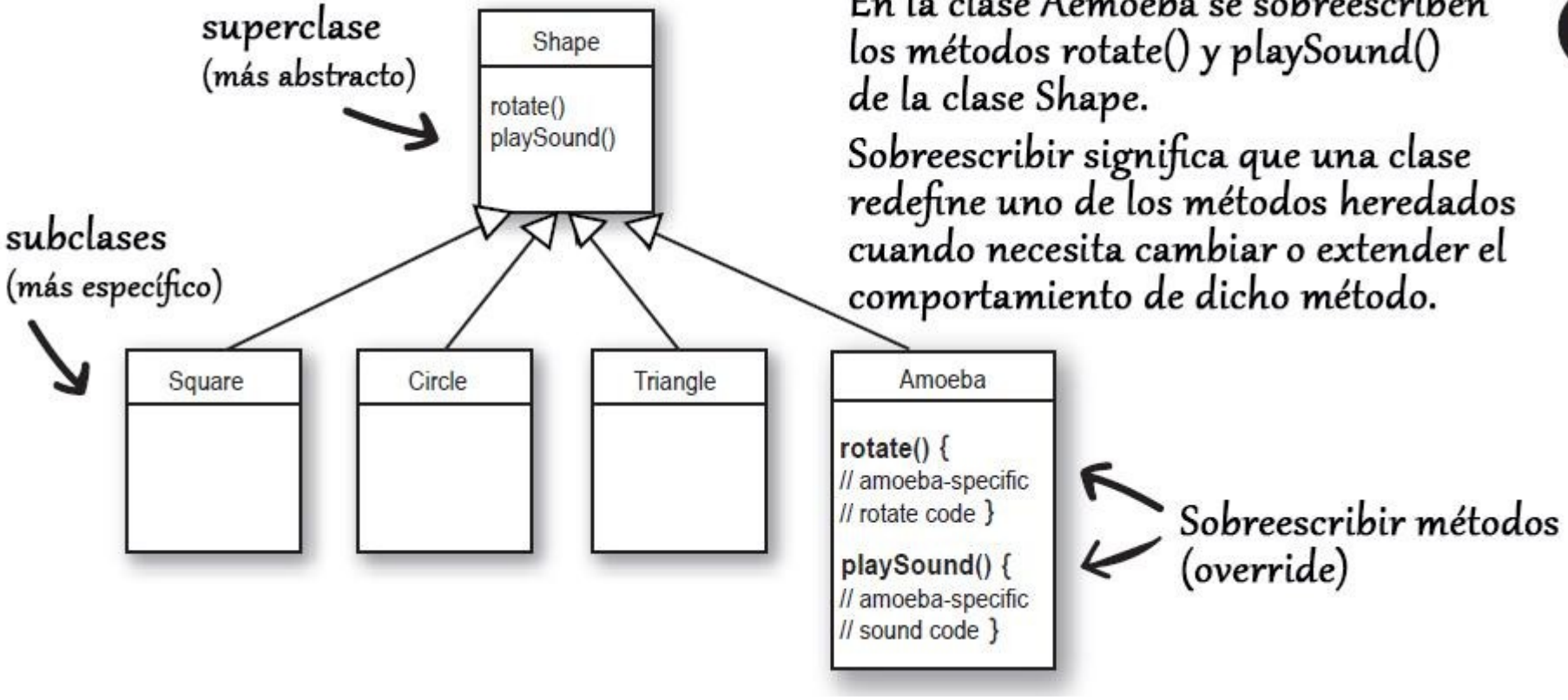


Superclasses y subclases.
Herencia



Las cuatro clases heredan de la clase *Shape*, por lo tanto heredan sus métodos.

Sobrescribir métodos. Polimorfismo



En la clase Aemoeba se sobreescriben los métodos rotate() y playSound() de la clase Shape.

Sobreescribir significa que una clase redefine uno de los métodos heredados cuando necesita cambiar o extender el comportamiento de dicho método.

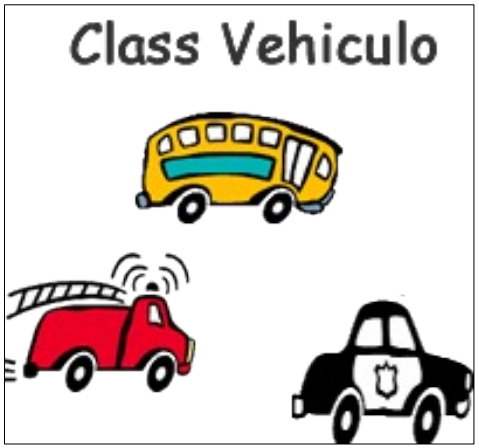
4

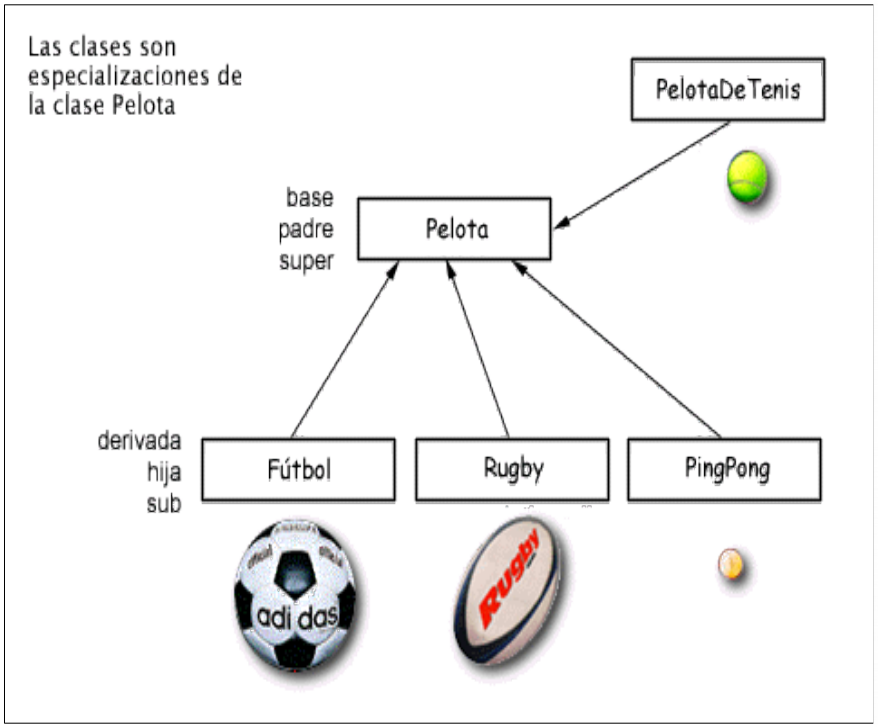
Sobrecargar métodos. Polimorfismo

```
 1 public class Calculos{
 2
 3       public int Suma(int a, int b){
 4             return a + b;
 5       }
 6
 7       public double Suma(double a, double b){
 8             return a + b;
 9       }
10
11       public long Suma(long a, long b){
12             return a + b;
13       }
14
15 }
```

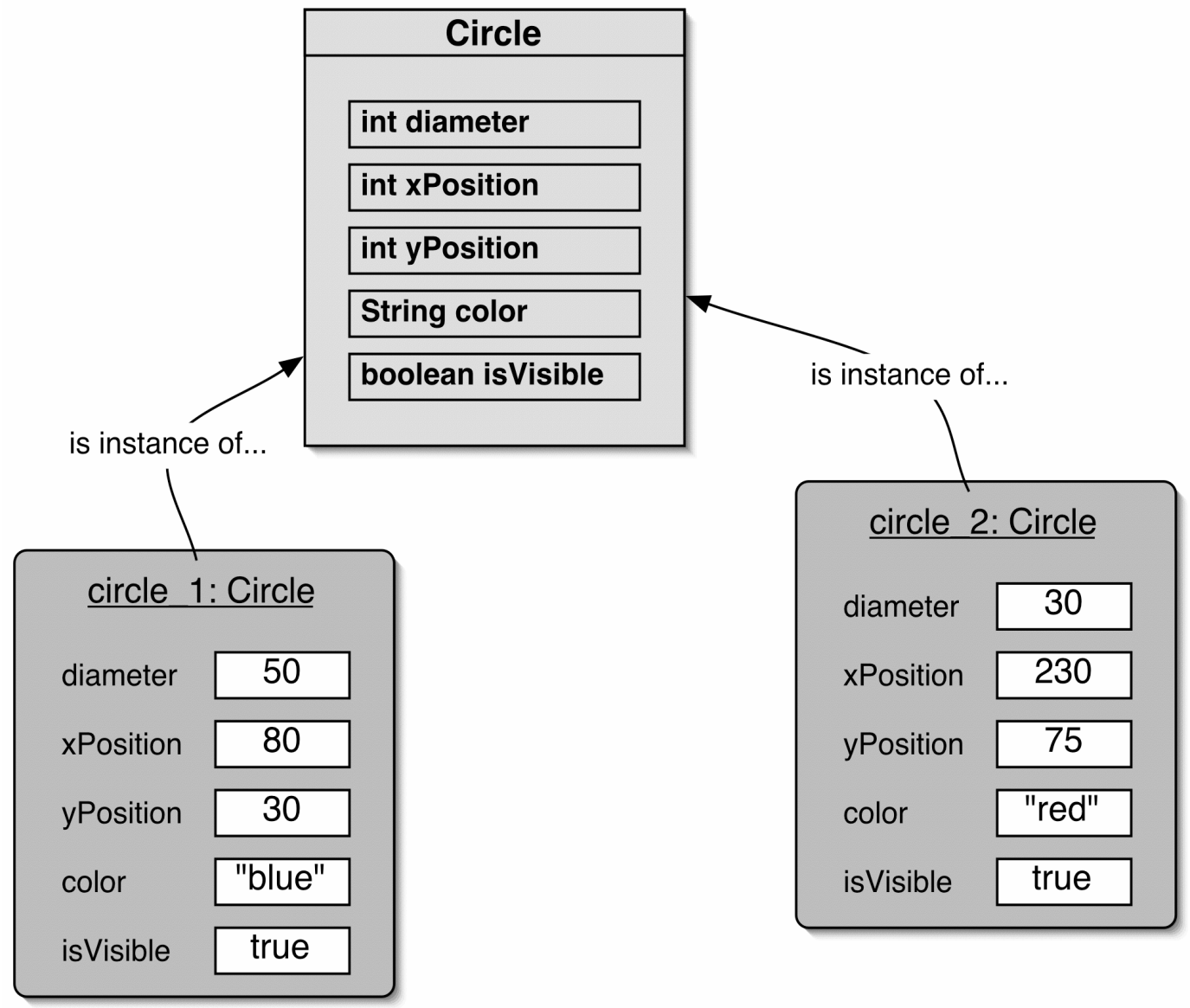
¿Qué es un objeto?

- Es una instancia de una clase.
- Características de los objetos:
  - Identidad: se distinguen unos de los otros.
  - Comportamiento: pueden realizar tareas.
  - Estado: guardan información que puede cambiar con el tiempo.





¿Qué es un objeto?



Ejemplo

CLASE

Cuenta
titular: String; saldo: double;
reintegro(importe:double); ingreso(importe:double);

ATRIBUTOS

MÉTODOS

OBJETO

“Jose Martínez”	titular
1200.0	saldo

```
class Cuenta{
    String titular;
    double saldo;

    void ingreso(double importe){
        saldo=saldo + importe;
    }
    void reintegro(double importe){
        if (importe <= saldo)
            saldo=saldo – importe;
    }
}
```

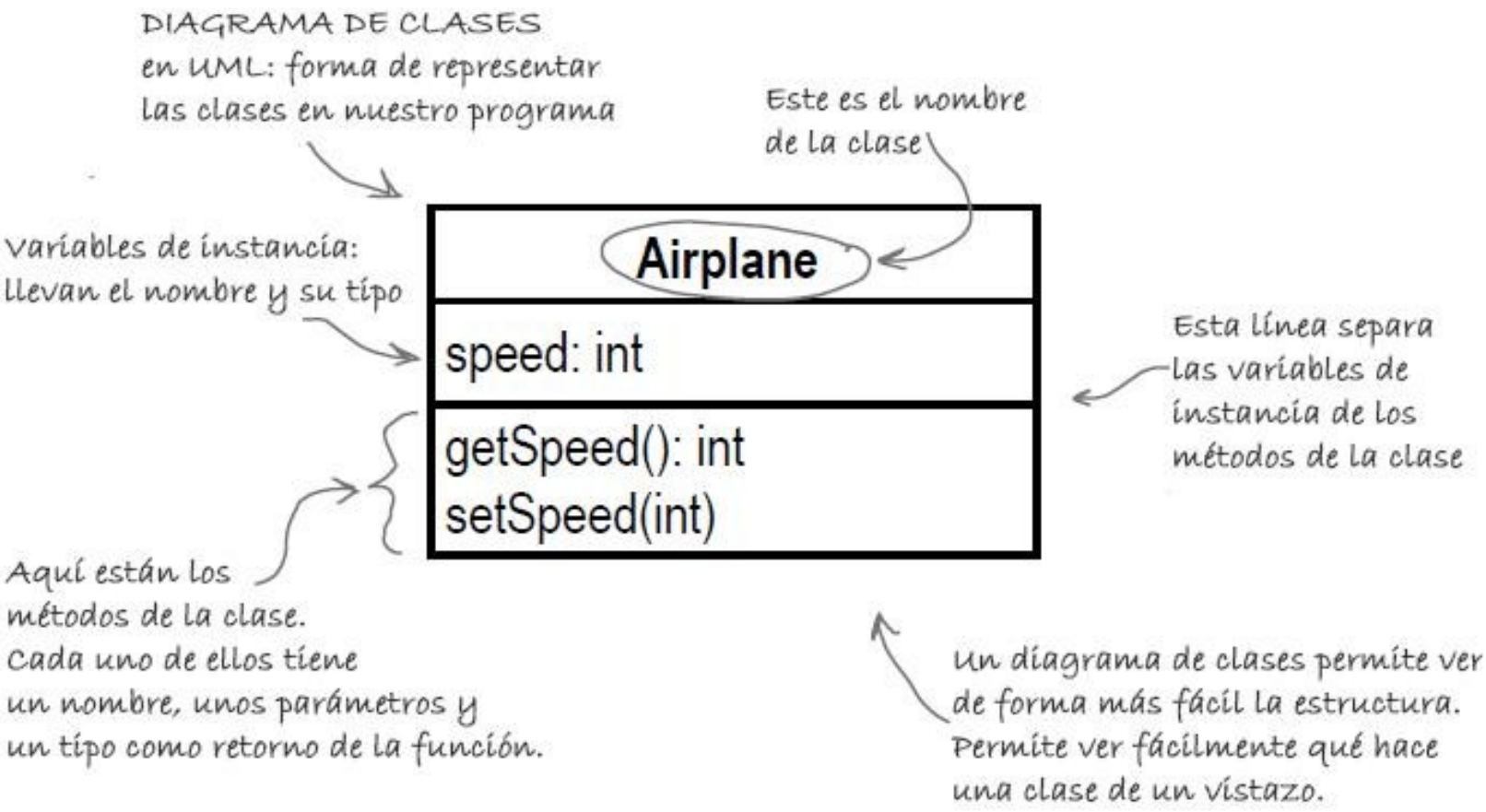
ATRIBUTOS

MÉTODOS

UML y diagramas de clases

- UML (Unified Modelling Language): lenguaje de modelado de datos.
- Diagrama de clases: es un esquema de UML que utilizaremos para definir las clases (atributos y métodos) y las relaciones entre ellas.

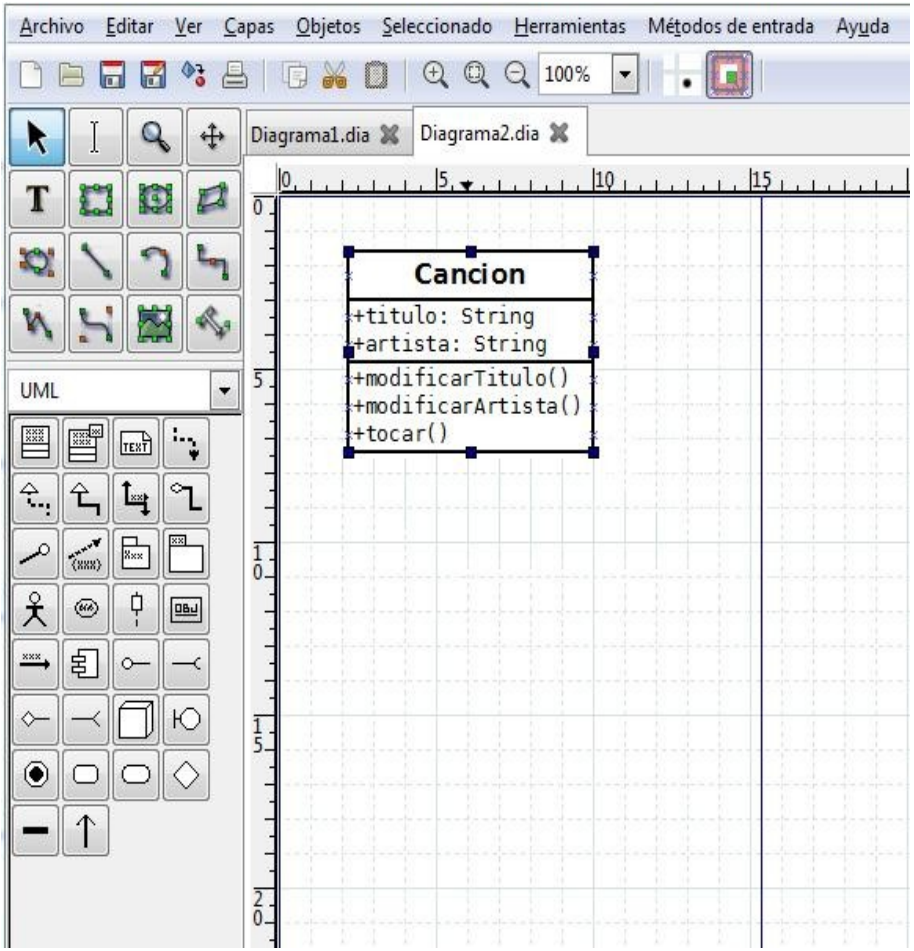
Diagrama de clases



Diagramas de clases con Día

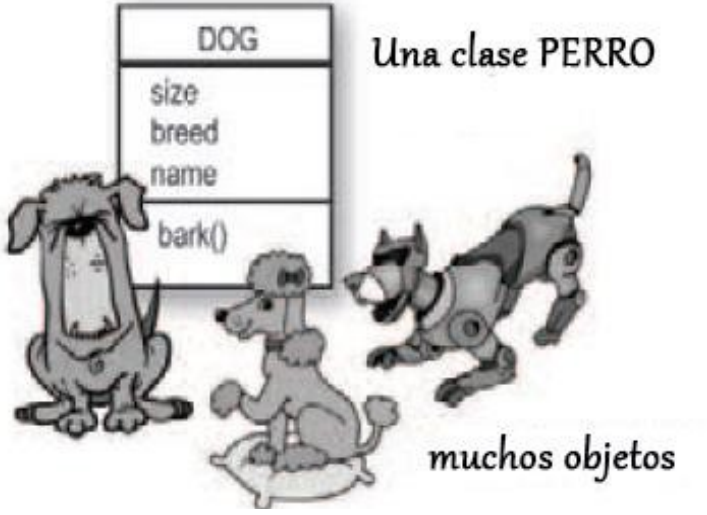
Dia Diagram Editor

- El editor UML Día nos permite modelar los diagramas de clases.
- Para cada clase indicaremos:
  - Nombre
  - Atributos: nombres y tipos de datos
  - Métodos: nombre, parámetros (si han) y tipos de retorno (si han).



Clases y objetos

- Una clase **no es** un objeto...
- … pero la clase puede ser utilizada para construir objetos.



Clases y objetos

- Una clase es el molde para el objeto.
- Cada objeto se crea desde una clase y tiene sus propios valores para las variables de instancia de esa clase.
- Ejemplo: de la clase *Button* se pueden crear muchos objetos Button diferentes, con su color, tamaño, forma y hashtag.

Clases y objetos

- Un objeto es como una entrada en la agenda:
  - Cada ficha tiene los mismos espacios en blanco (variables de instancia o atributos).
  - Al llenar una nueva ficha, creamos el objeto.
  - Los métodos su acciones que podemos hacer con los objetos (obtenerNombre(), escribirNombre()).
  - Cada ficha puede hacer las r
    pero con datos diferentes.

El operador punto (.)

Da acceso al estado y al comportamiento del objeto (atributos y métodos).

```
//creamos un nuevo objeto Cuenta
Cuenta c1 = new Cuenta();
//cambiamos el titular de la cuenta
c1.titular=”María Martínez”;
//invocamos al método ingreso
c1.ingreso(200.20);
```

Trabajando con objetos...

- ¿Qué necesitamos (**de momento**)?
  - Una clase para el objeto
  - Una clase *test* para probar el objeto, que contendrá el método main()
  - Pondremos en las clases test el mismo nombre que la clase que contiene el objeto con el sufijo TestDrive.
  - Ejemplo: Alumno y AlumnoTestDrive
  - IMPORTANTE: recordad que se tiene que ejecutar

Trabajando con clases...

1 Escribe tu clase

```
class Dog {

  int size;
  String breed;
  String name;

  void bark() {
    System.out.println("Ruff! Ruff!");
  }
}
```

variables de instancia

un método

Trabajando con clases...

2 Escribe una clase test (TestDrive)

un método público (más adelante se escribirá el código)

```
class DogTestDrive {
  public static void main (String[] args) {
    // el código de la clase test irá aquí
  }
}
```

3 En la clase test, crear el objeto y acceder a sus variables y métodos

```
class DogTestDrive {
  public static void main (String[] args) {
    Dog d = new Dog();
    d.size = 40;
    d.bark();
  }
}
```

crear el objeto Dog

usar el operador punto (.) para modificar la variable size

operador punto (.)

y llamar al método Bark()

Definición de clase y clase TestDrive

- Tendremos en los archivos .java las dos clases, la definición de la clase y su clase TestDrive asociada.
- Las clases definidas en cada fichero...
  - Ejemplo.java
  - ExempleTestDrive.java
- … crearán dos ficheros .class:
  - Ejemplo.class
  - ExempleTestDrive.class (se ejecutará este).