<codoa codo/>


# DJANGO
# Clase 4

## Python – Diseño POO

BA

# Les damos la bienvenida

Vamos a comenzar a grabar la clase

## Clase 03

**Python - Introducción**

- Fundamentos del lenguaje
- Debug en Python
- Entorno virtual
- Módulos y librerías
- Tipos de datos
- Funciones

## Clase 04

**Python – Diseño POO**

- Diseño de clase (draw.io, EA, Visual Paradigm, etc)
- Modelo de Dominio
- Diagrama de Clases
- Identidad, estado y comportamiento
- Relaciones entre clases
- Polimorfismo

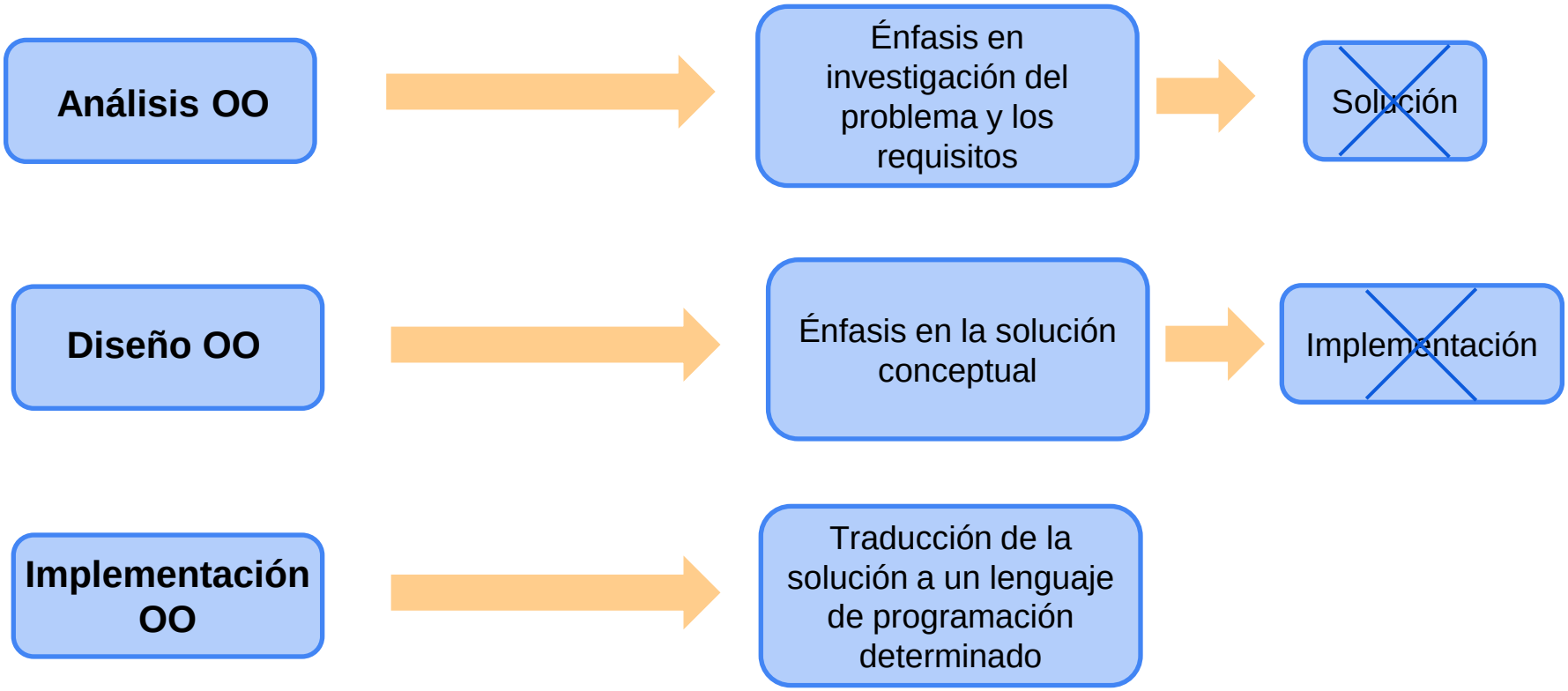
# ¿Qué es POO?

El Paradigma orientado a objetos, define los programas en término de comunidades de objetos. Los objetos con características comunes se agrupan en clases.

**ES UNA FORMA DE VER EL MUNDO**

Se impuso por:

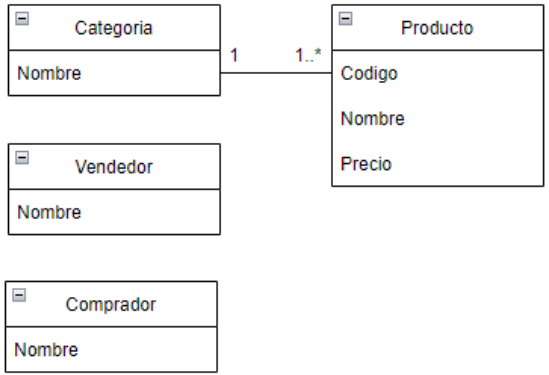
- Reduce la brecha entre el mundo de los problemas y el mundo de los modelos.
- Conceptos comunes a lo largo de todo el ciclo de vida
- Uso de patrones
- Aumento complejidad de los sistemas
- Aumento de necesidad de reutilización


Análisis OO
Énfasis en investigación del problema y los requisitos
Solución
Diseño OO
Énfasis en la solución conceptual
Implementación
Implementación OO
Traducción de la solución a un lenguaje de programación determinado

**Análisis OO**

Se presta especial atención a encontrar y describir los conceptos del dominio del problema

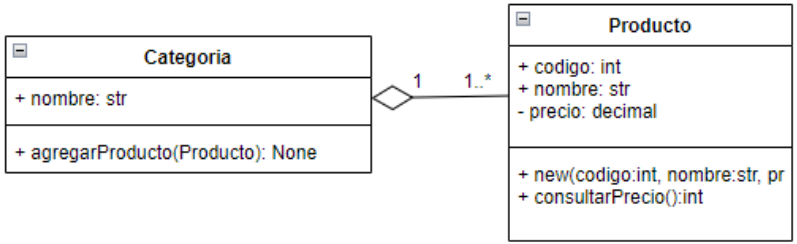
Modelo de Dominio




**Diseño OO**

Se presta atención a la definición de los objetos software y en como colaboran para satisfacer los requisitos

Diagrama de Clases



# Estado, comportamiento e identidad

“El **estado** de un objeto abarca todas las propiedades (normalmente estáticas) del mismo, más los valores actuales (normalmente dinámicos) de cada una de esas propiedades”

“El **comportamiento** nos muestra como actúa y reacciona un objeto, en términos de sus cambios de estado y paso de mensajes”

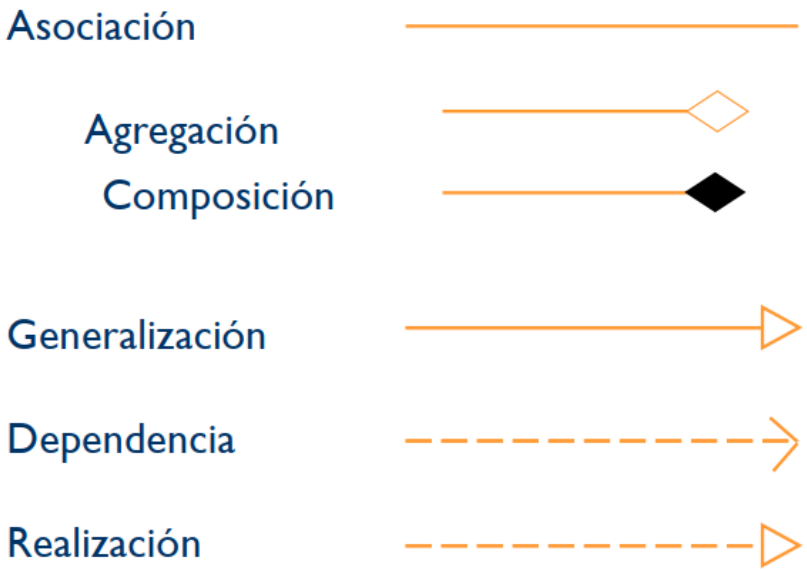
“La **identidad** es aquella propiedad de un objeto que lo distingue de todos los demás objetos”

# Relaciones entre clases

- Las clases generalmente no se encuentran aisladas, existen tres tipos principales de relaciones:

  - **Dependencias**: relaciones de uso entre clases
  - **Asociaciones**: relaciones estructurales entre clases
  - **Generalizaciones**: conectan clases generales con sus especializaciones (se implementa a través de la herencia)

# Relaciones entre clases



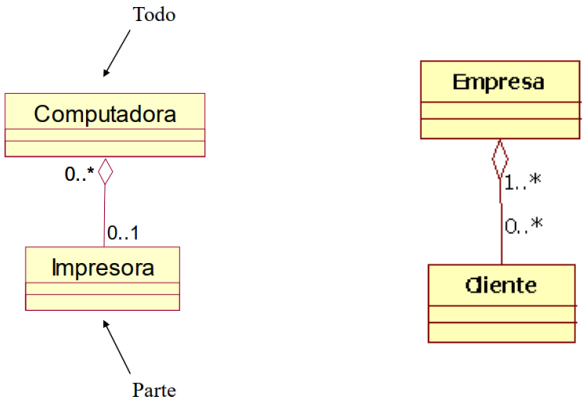
# Relaciones entre clases

**Dependencia**

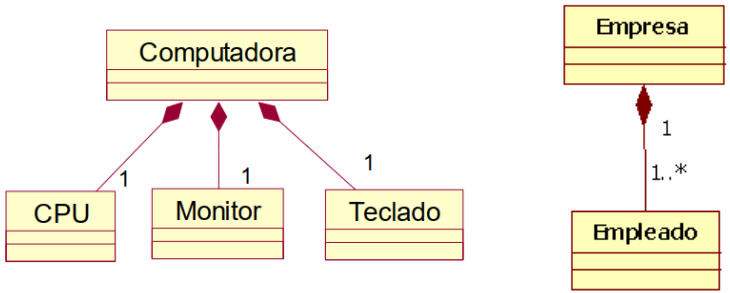

Por ejemplo para resolver una ecuación de segundo grado, tenemos que recurrir a la función sqrt de la clase Math para calcular la raíz cuadrada.

# Relaciones entre clases

## Agregación

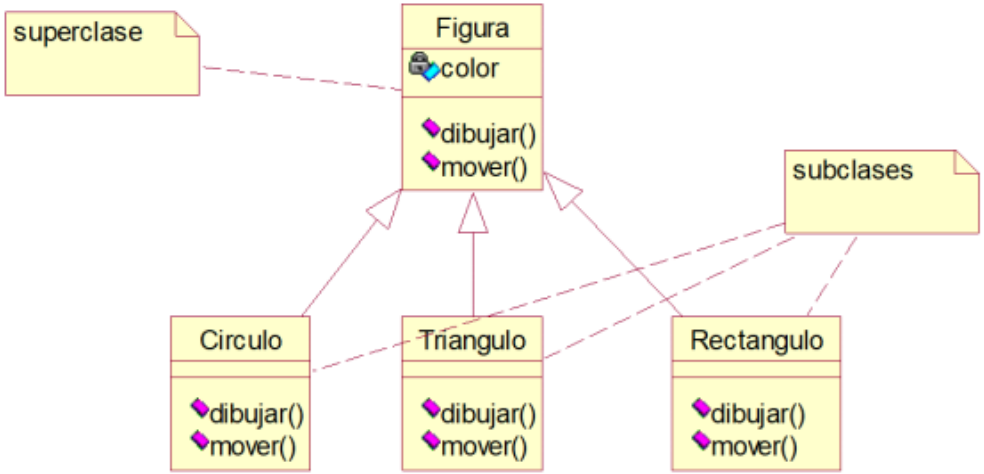



## Composición



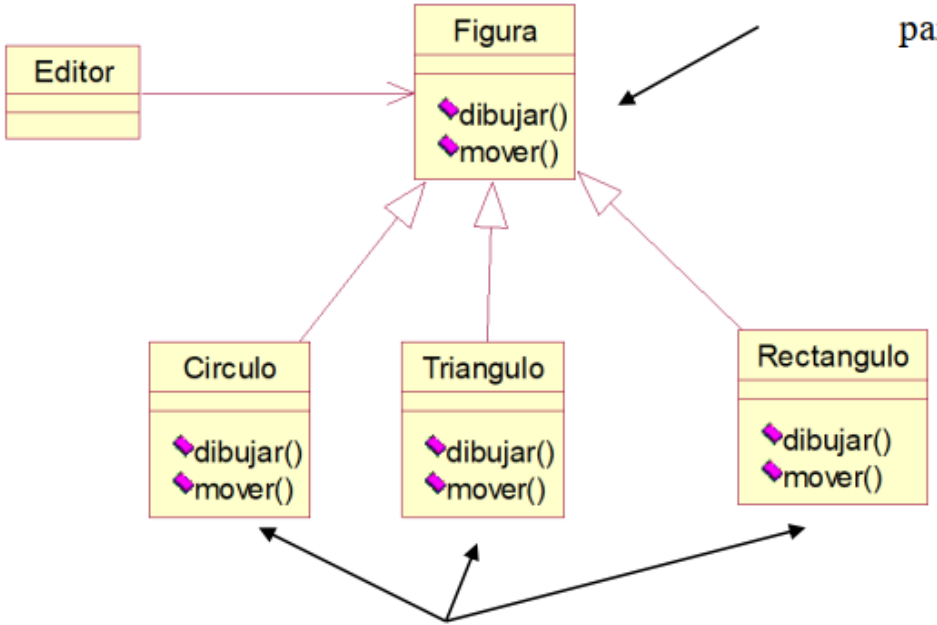
# Relaciones entre clases

**Generalización**



# Polimorfismo

El método dibujar() no es igual para cada figura




Cada vez que se invoque el método dibujar() dependerá de la clase a la pertenece el objeto

```
unCirculo.dibujar();
UnTriangulo.dibujar();
unRectangulo.dibujar();
```

Redefino en las subclases el método dibujar()

# No te olvides de completar la asistencia y consultar dudas

# Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

**TODO EN EL AULA VIRTUAL**