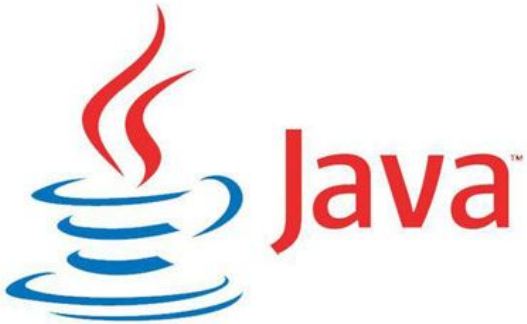
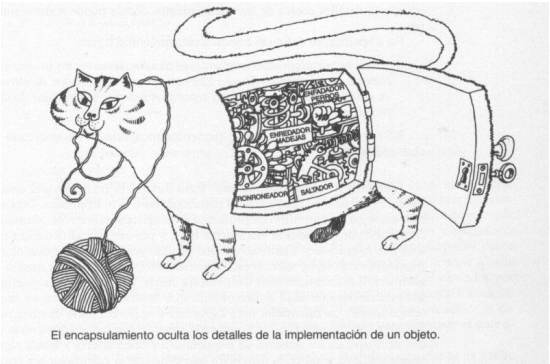
Paradigma de programación orientada a objetos



Pilares POO

Encapsulamiento

El estado interno de un objeto es privado. En la implementación JAVA, esto se traduce a atributos “private”



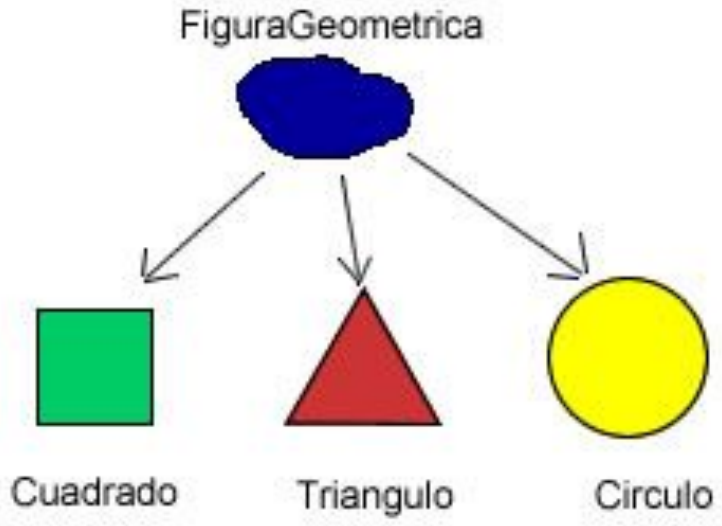
El encapsulamiento oculta los detalles de la implementación de un objeto.

Pilares POO

Herencia

Permite establecer jerarquías entre los objetos.

Los objetos de la clase hijo son del tipo de la clase padre.



Pilares POO

Polimorfismo

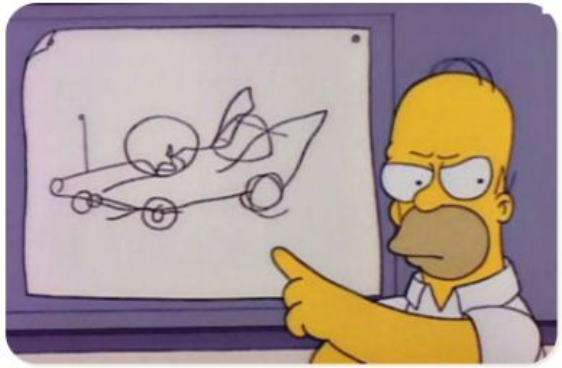
Objetos de distinta forma pueden responder al mismo mensaje.

Pilares POO

Abstracción

Nos olvidamos de la forma que tiene el objeto. Solamente nos interesa las responsabilidades que el objeto deberá saber responder.

Abstracción



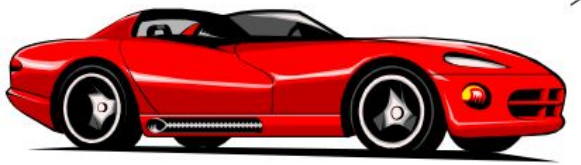
Homero Simpson construyendo el auto de sus sueños

Énfasis en el ¿qué hace? más que en el ¿cómo lo hace?

Componentes POO

Objeto

Entidad que posee un estado interno y un comportamiento. Es una instancia de una clase.



- Atributos:
  - color
  - velocidad
  - ruedas
  - motor
- Métodos:
  - arranca()
  - frena()
  - dobla()

Componentes POO

Clase

Modelo de un objeto.
Se puede pensar como una fábrica de objetos.
Además existen las clases abstractas, que no permiten crear nuevas instancias. Se utilizan para modelar jerarquías.

Componentes POO

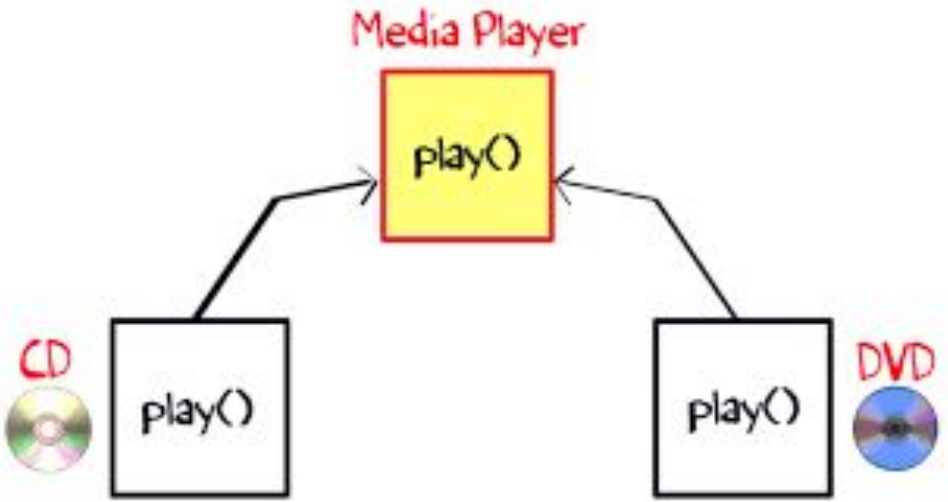
Mensaje / Metodos

Modelan la responsabilidad de un objeto.
¿Qué debe hacer?

Componentes POO

Interfaz

Contrato que define qué comportamiento tendrán todas la clases que la implementen.
Es decir, en una interfaz se definen qué métodos deberán implementar dichas clases.



Overriding (Sobreescritura)

Se utiliza para sobreescribir en una clase el comportamiento de un método del padre.

```java
public class Mamifero {

    public void comer(){
        System.out.println("Como como un Mamifero");
    }
}
```

```java
public class Perro extends Mamifero{

    @Override
    public void comer() {
        System.out.println("Estoy comiendo como un perro");
    }
}
```

Overloading (Sobrecarga)

Se utiliza para definir un mismo método, pero con distintos parámetros.

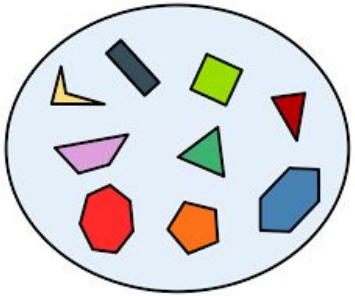
```
public Double suma(Integer unEntero, Double unDecimal)

public Double suma(Integer unEntero, Integer otroEntero)
```

Estructuras de datos

Colecciones

Listas
List

Conjuntos
Set



Equals (comparación entre objetos y para Lists)

Se define para determinar cuando dos objetos son iguales.
Lo utilizan las listas para saber si el objeto está dentro.

```
@Override
public boolean equals(Object obj) {

  //CASTEO: JAVA ASUME QUE UN OBJETO ES DE UN TIPO INDICADO POR EL USUARIO
  Cliente otroCliente = (Cliente) obj;

  if(this.getNumeroDeCliente().equals(otroCliente.getNumeroDeCliente())){
     return true;
  }
  else{
     return false;
  }
}
```

HashCode (comparación para Set)

Se define para determinar cuándo dos objetos son iguales.
Lo utilizan los conjuntos para saber si el objeto está dentro

```
@Override
public int hashCode() {
   return numeroDeCliente.hashCode();
}
```