

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<html> <head>
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
<meta name="revisit-after" content="2 days">
<meta name="netinsert" content="840.0.1.1.2.1">
<meta name="distribution" content="global">
<meta name="resource-type" content="document">
<meta name="rating" content="General">
<meta name="ROBOTS" content="INDEX, ALL">
<meta name="ROBOTS" content="INDEX, FOLLOW">
<meta http-equiv="content-language" content="es">
</head>
```

# PROGRAMACIÓN ORIENTADA A OBJETOS

## INTRODUCCION A LA PROGRAMACION ORIENTADA A OBJETOS

### Clases y Objetos

#### UNIDAD 1



# LOGRO DE LA UNIDAD 1

- Al finalizar la Unidad 1 el alumno verifica los conceptos de clases y objetos usando pruebas unitarias.



# AGENDA

1. Definición de Programación Orientada a Objetos
2. Clases y Objetos
3. Terminología
4. Encapsulamiento
5. Ventajas y Desventajas de la POO
6. Conclusiones



# 1. Programación orientada a objetos

- La **programación orientada a objetos** o **POO** (**OOP** según sus siglas en inglés) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.
- Está basado en varias técnicas, incluyendo **herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento**.
- Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.



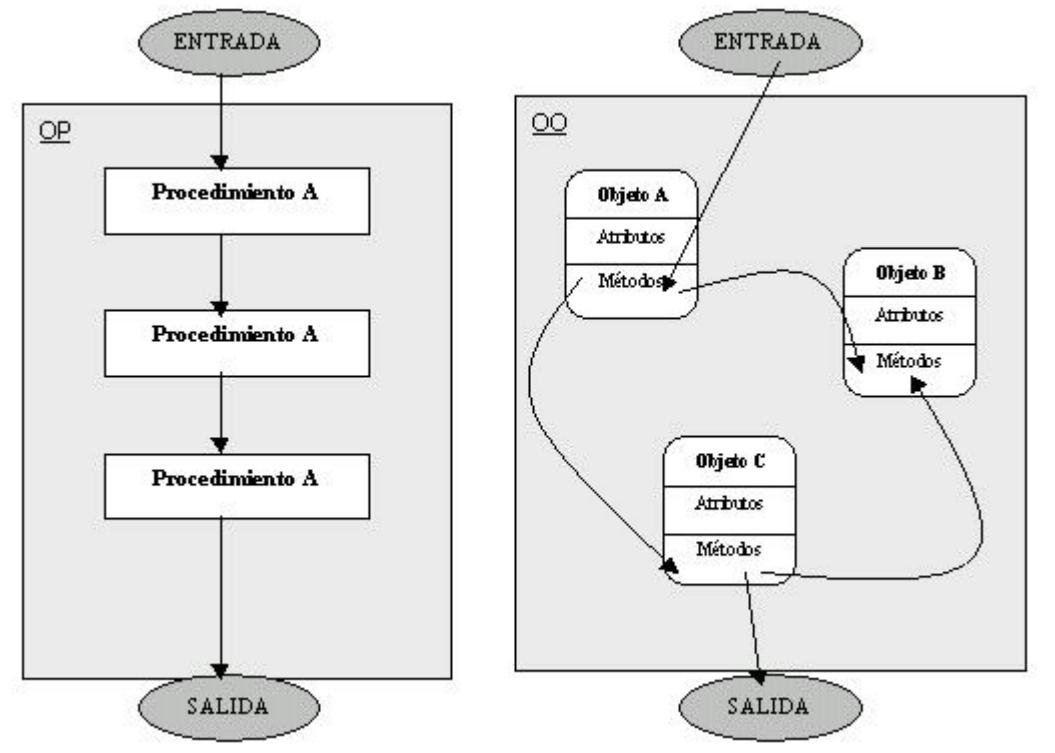
# Lenguajes de Programación Orientada a Objetos

- C++
- Objective C
- Java
- Smalltalk
- Eiffel
- Lexico (en castellano)
- Ruby
- Python
- SDK
- OCAML
- Object Pascal
- CLIPS
- Visual .net
- Actionscript
- COBOL
- Perl
- C#
- Visual Basic.NET
- PHP
- Simula
- Delphi
- PowerBuilder
- Maya
- Corel draw



# Programación Orientada a Objetos

- Toma prácticas de paradigmas de programación previos como la programación estructurada y la modular.



## 2. Clase

- ✓ **Clase** es la representación abstracta de uno o más objetos.
- ✓ **Clase** posee un tipo de dato y tiene asociado atributos y métodos.
- ✓ **Clase** es la clasificación de un conjunto de objetos.



## 2. 1 Objeto

- Un **objeto** puede caracterizar una entidad física (caballo, coche, tarjeta) o no física( ecuación matemática)
- **Objeto:** Unidad atómica que encapsula atributos y comportamiento y posee un estado específico.



## 2.2 Clase - Objeto

- Clase Caballo

- Atributos o características:

- color
- raza
- tipo:
- tamaño
- Sexo
- velocidadMax
- posX
- posY
- posZ

- Comportamientos u operaciones:

- salta()
- corre()
- retrocede()
- parte()

- Objeto , “miMascota” es la instancia de la clase Caballo:

- color = blanco
- raza = andaluz
- Tipo = de guerra
- Tamaño = grande
- Sexo = macho
- velocidadMax = 65 km/hora
- posX = 344
- posY = 242
- posZ = 55

- Objeto , “elCorcel” es otra instancia de la clase Caballo:

- color = negro
- raza = griega
- Tipo = de carrera
- Tamaño = mediano
- Sexo = hembra
- velocidadMax = 95 km/hora
- posX = 533
- posY = 222
- posZ = 22



## 2.2 Clase - Objeto

**Clase:**

**Punto**

**Atributos:**

x

y

color

**Métodos:**

crearse

mostrarse

ocultarse

moverse

**Instancias de la Clase Punto**

**Objeto 1**

**Atributos:**

21

45

verde

**Objeto 3**

**Atributos:**

200

15

rojo

**Objeto 2**

**Atributos:**

142

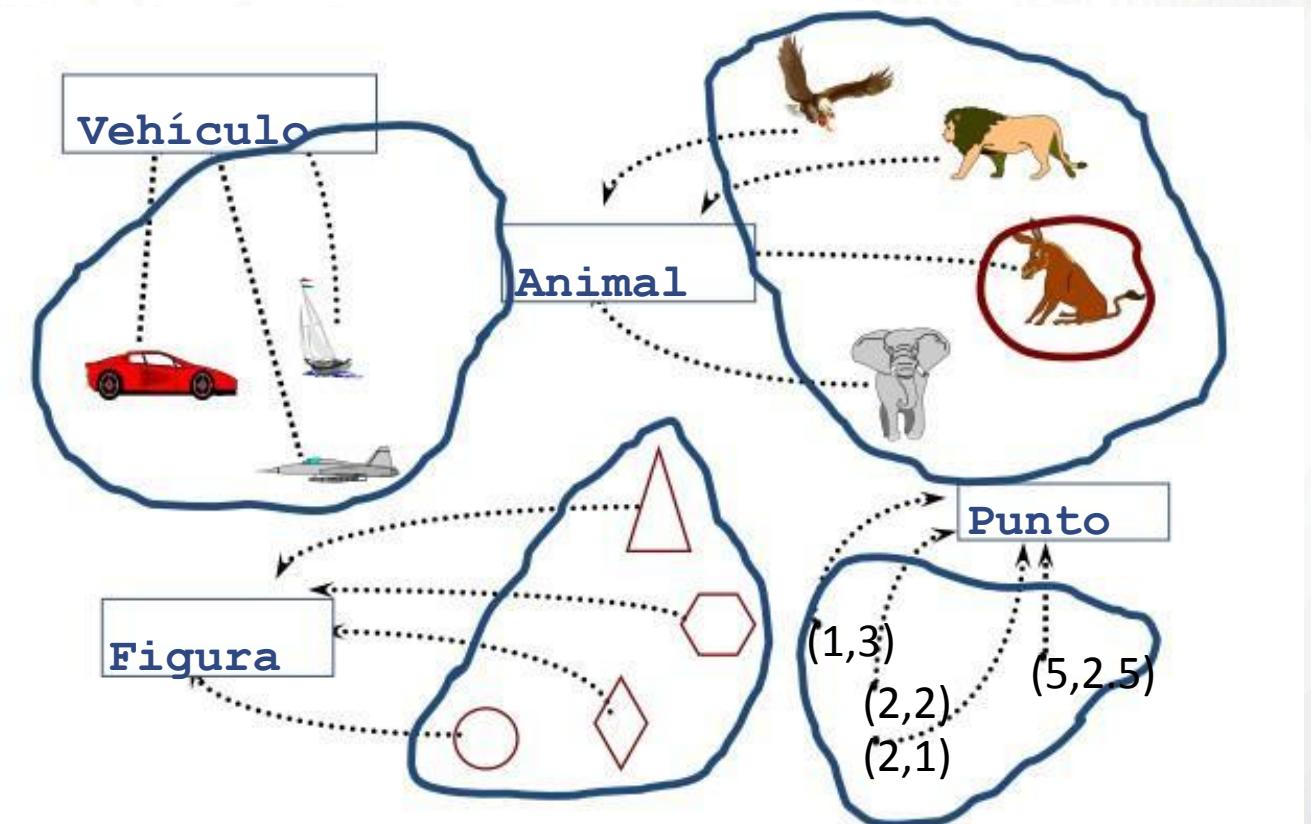
90

celeste



## 2.2 Casi siempre los clasificamos

Las clases y los objetos están en todas partes en el mundo real.



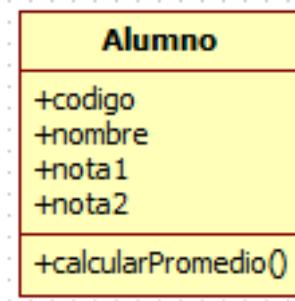
## 2.3 Una Clase representada en código java

**Clase = Atributo(s) + Comportamiento(s)**

```
public class Alumno {  
    private String codigo;  
    private String nombre;  
    private int nota1, nota2;  
    public double calcularPromedio(){  
        return (nota1 + nota2)/2.0;  
    }  
}
```

Atributos

Comportamiento



Clase representada en UML

## 2.4 Y como se crean objetos en código java?

```
public class Alumno {  
    private String codigo;  
    private String nombre;  
    private int nota1, nota2;  
  
    public Alumno(String codigo, String nombre, int nota1, int nota2){  
        this.codigo = codigo;  
        this.nombre = nombre;  
        this.nota1 = nota1;  
        this.nota2 = nota2;  
    }  
  
    public double promedio(){  
        return (nota1 + nota2)/2.0;  
    }  
}
```

```
Alumno alumno1 = new Alumno("8888", "Juan Perez", 20, 18);  
  
Alumno alumno2 = new Alumno("9999", "Ana Lopez", 17, 18);
```

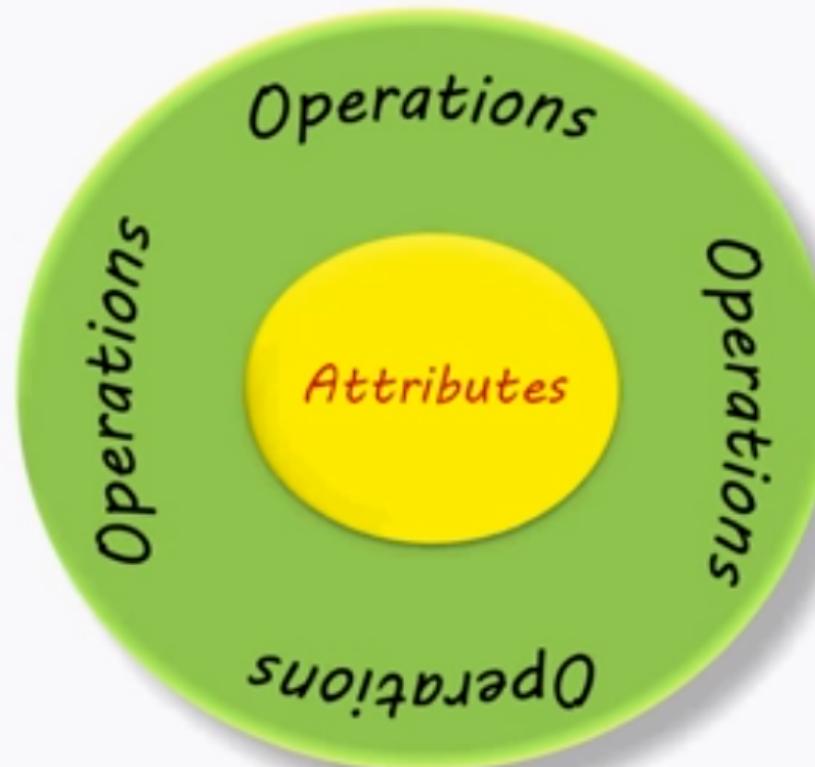


### 3. Terminología Básica

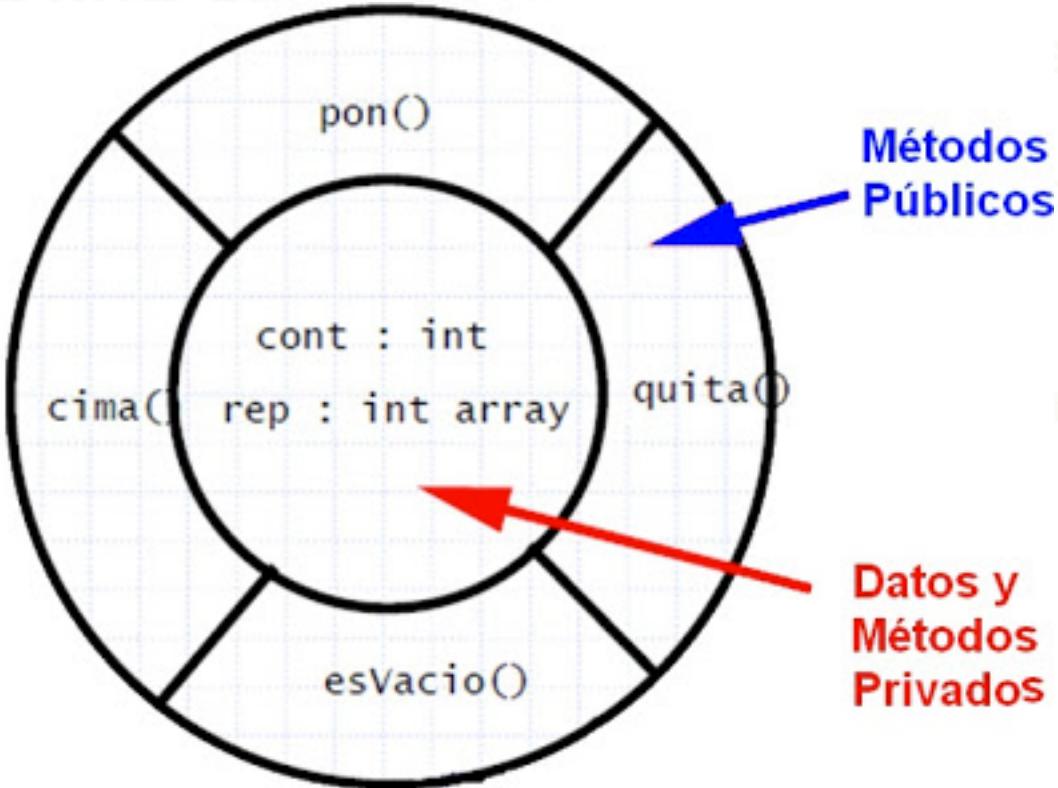
- Atributo: Estado, propiedad de un objeto.
- Comportamiento: Acción que realiza un objeto y que será definido como método de su clase.
- Mensaje o solicitud: Invocación a un objeto para que ejecute cierto método . Forma de comunicarse entre los objetos para lograr un funcionamiento determinado del sistema.
- Programa Orientado a Objetos: Conjunto de objetos colaborando o Conjunto de objetos enviando mensajes y respondiendo a otros mensajes



# 4. Encapsulamiento



# 4.1 Encapsulamiento



## 5. VENTAJAS DE POO

- Reusabilidad
- Extensibilidad
- Facilidad de Mantenimiento
- Portabilidad
- Rapidez de Desarrollo
- Más fáciles de entender porque se utilizan abstracciones más cercanas a la realidad



## 5. DESVENTAJAS DE POO

- Para procesos simples como leer y escribir algunos datos no tienes necesidad de definir clases y objetos. Sin embargo en Lenguajes POO tienes que realizar ese paso extra.
- Dificultad en la abstracción.
- El concepto que un programador tiene de lo que constituye un objeto abstracto puede no coincidir con la visión de otro programador. Los objetos a menudo requieren una extensa documentación.



## 6. Conclusiones

- Representa mejor a un sistema real.
- Permite crear sistemas más complejos.
- Permite la construcción de prototipos
- Permite una mejor comunicación para el equipo
- Agiliza el desarrollo de software
- Facilita el mantenimiento del software



## 6. Conclusiones

- ❑ El futuro de POO probablemente se encuentra en lenguajes como Ruby y Lua, en el que se construye el concepto de objeto en el lenguaje y no siempre es explícitamente controlado por el programador.
- ❑ Ruby, por ejemplo, trata todo como un objeto, incluidas cadenas, números, tu programa y los contenidos del directorio en el cual trabajas. Casi nunca tendrás que declarar algo como un objeto.



## 6. Conclusiones

- Eres libre de utilizar el lenguaje como quieras: como programación orientada a objetos, como un lenguaje funcional, como lenguaje de procedimiento o incluso en formas que mezclan dichos paradigmas.

