Clases y Objetos

Curso: Programación Orientada a Objetos

LOGRO DE LA UNIDAD 1

Al finalizar la Unidad 1 el alumno podrá implementar, en un programa, los conceptos de clases y objetos.

AGENDA

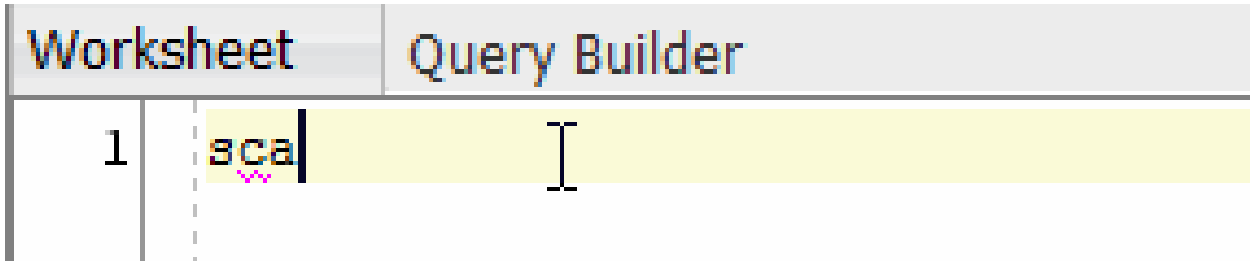
1. Introducción
2. Definición de Clases y Objetos
3. Terminología
4. Encapsulamiento
5. Ventajas / Conclusiones

1. INTRODUCCION

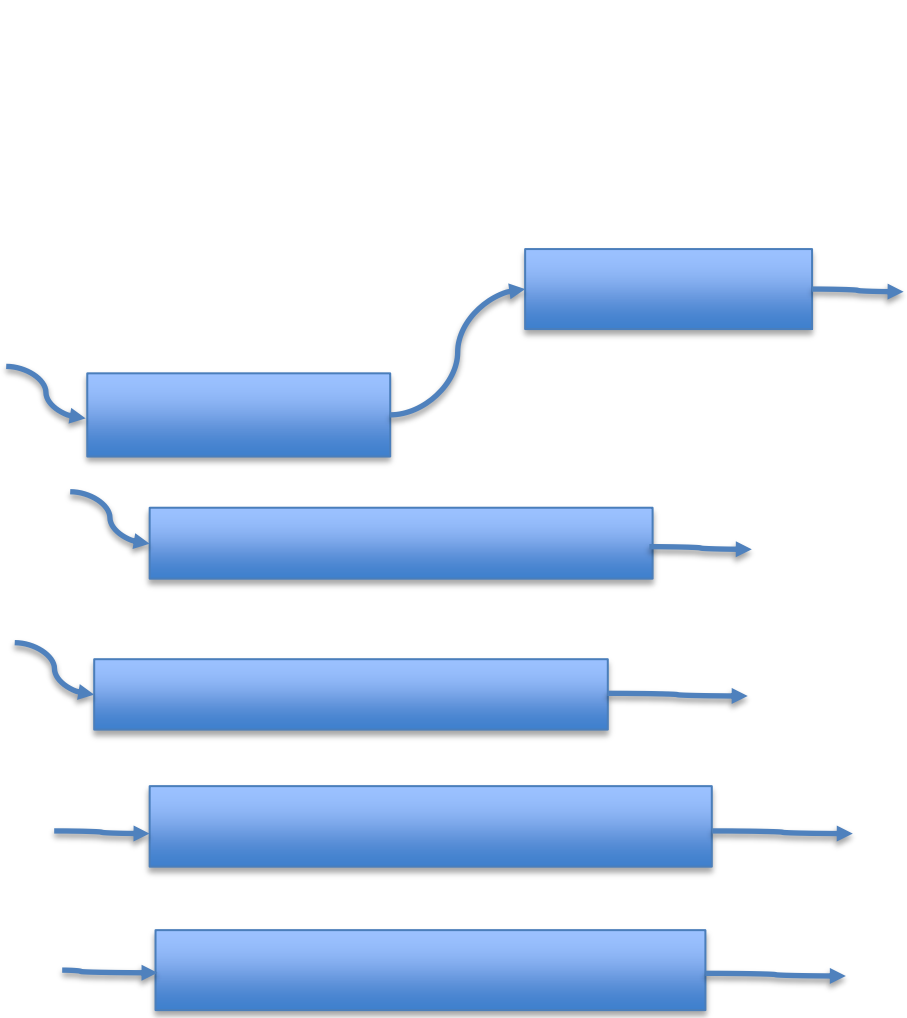
- La **programación orientada a objetos** o **POO** (**OOP**) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.
- Está basado en varios conceptos: abstracción, encapsulamiento, herencia, y polimorfismo.
- Su uso se popularizó 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.
- Toma prácticas de paradigmas de programación previos como la programación estructurada y la modular .

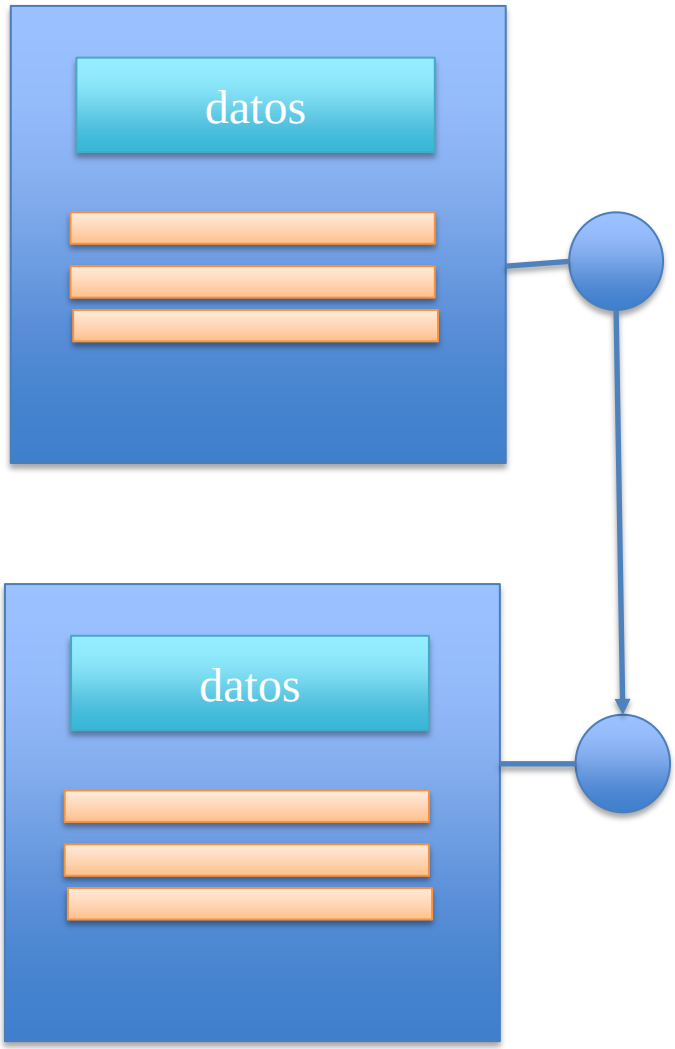
Lenguajes funcionales

Lenguajes como Prolog, Haskell, Miranda o SQL **no** son lenguajes orientados a objetos pues tienen otros objetivos como la programación funcional/estructurada (mediante funciones lógico matemáticas) o el manejo de datos en una base de datos relacional como en el caso de SQL



PROGRAMACION ESTRUCTURADA VS OBJETOS





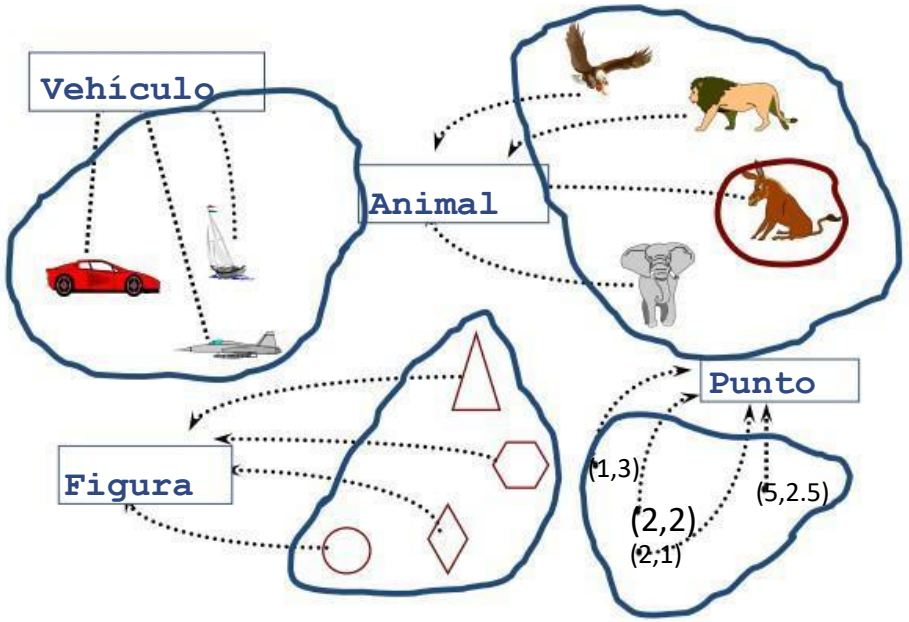
Lenguajes Orientado de Objetos

Java, Ruby, Python , C++, PHP entre otros permiten el uso del paradigma de Programación Orientada a Objetos. Cabe resaltar que el uso de uno de estos lenguajes no asegura el uso correcto de la POO pues depende de la correcta aplicación de sus principios.

```
object Persona
  attribute + nombre = "Juan";
  attribute + apellidos = "Nadie";
  attribute + telefono = "906414141";
  attribute + edad = 18;
  attribute + direccion;

  method + toString()
  {
    return nombre.concat( apellidos );
  }
endObject
```

Los objetos casi siempre los clasificamos (abstraemos)

Los objetos se encuentran en todas partes en el mundo real.



- Triángulo es un objeto de la clase Figura
- (1,3) es un objeto de la clase Punto

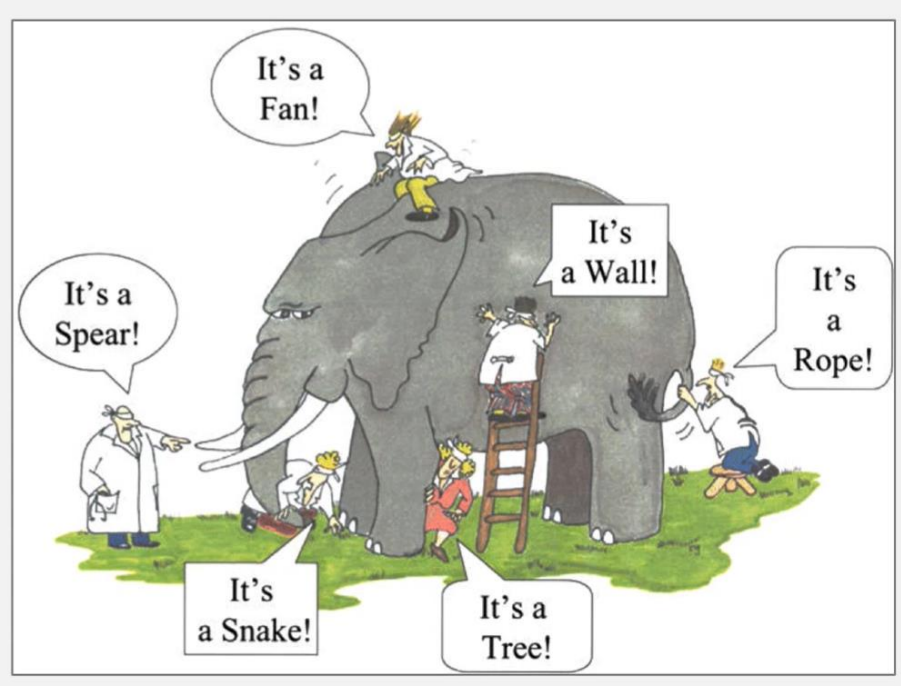
Parte del éxito del paradigma de la Programación Orientada a Objetos es el hecho de **clasificar** los objetos de manera muy similar a como lo hacemos en el mundo real. Esto es, a partir de las **características** de los objetos que percibimos.

De esa manera podemos saber que un **auto** es un **vehículo** o que el **perro** es un **animal**.

1. Abstraction

Denota características esenciales de un objeto.

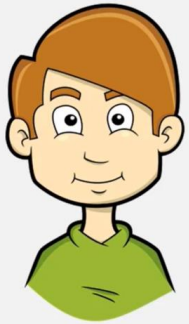
- Distinguirlo de otros objetos
- En relación con la perspectiva del espectador
- Depende del dominio del problema



Persona

SCHOOL

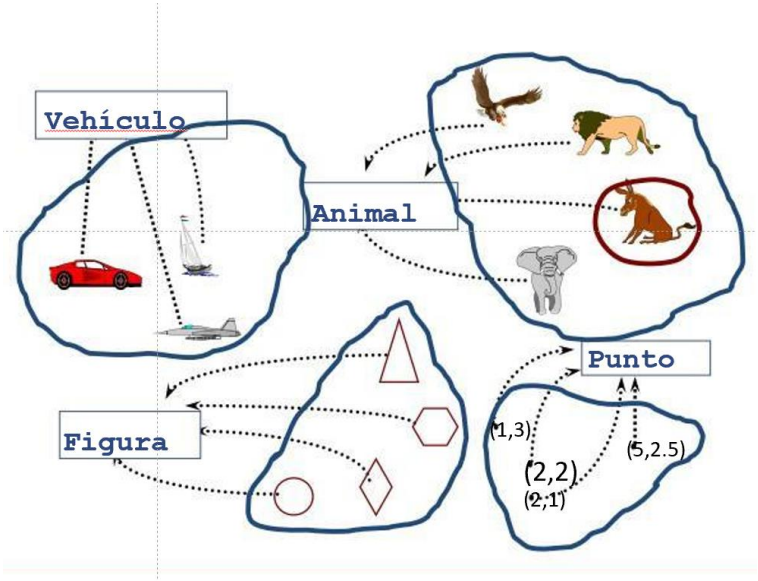
Name
RollNo
Class
Registration No.
Marks
GetGrade
CalculateTotalScore



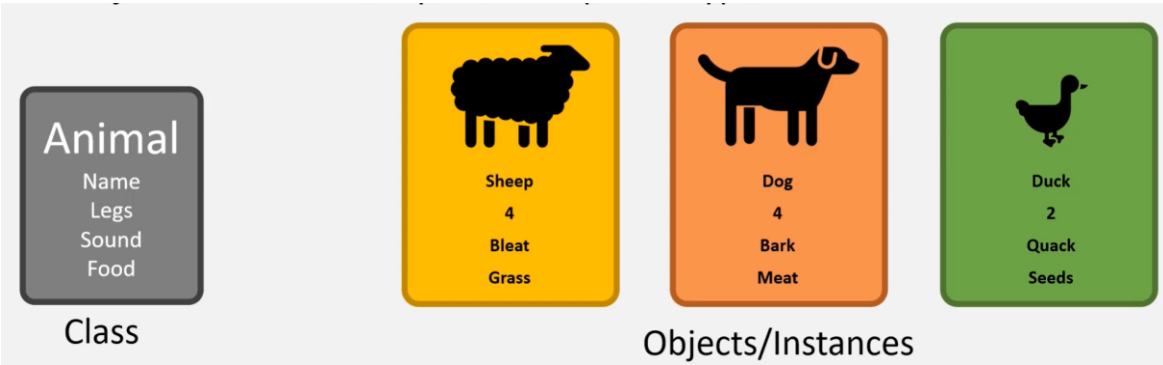
INNOCULATION

Name
Age
Weight
Gender
Allergies
School
History
AddRecord
GetRecord

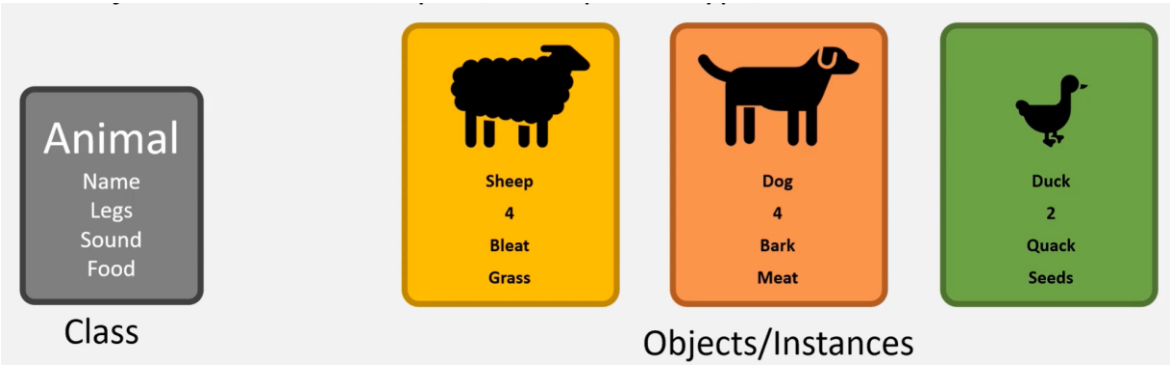
2. Clase



Caballo
color raza tipo: tamaño Sexo velocidad Max
correr() saltar() parar() modificarColor(nuevoColor)



- ✓ Clase es la representación abstracta, general de uno o más objetos.



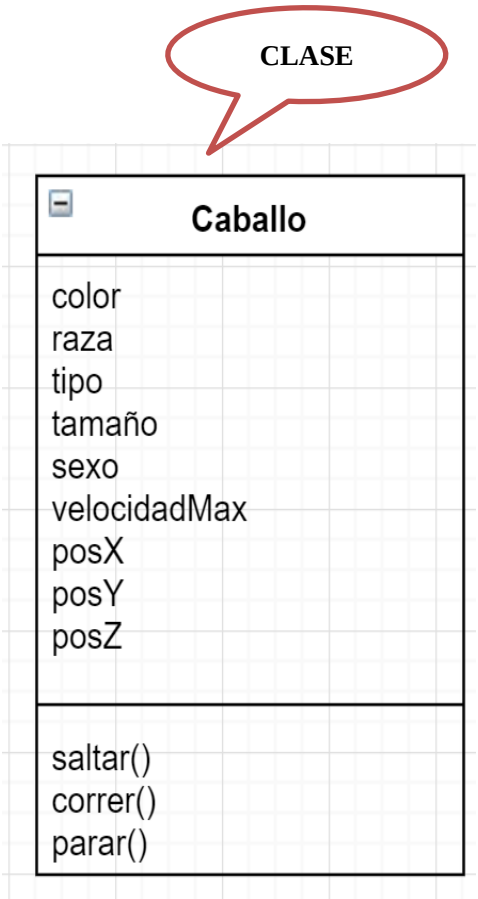
- Una Clase describe las propiedades y comportamientos de los objetos que representa.

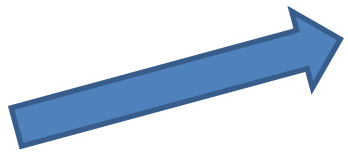
El nombre de la clase debe estar en Singular

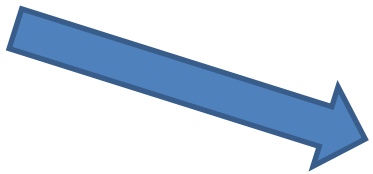
Caballo
color raza tipo tamaño Sexo velocidad Max
correr() saltar() parar() modificarColor(nuevoColor)

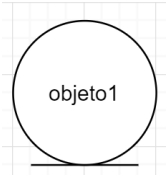
UML

UML









objeto1 : instancia de la clase
Caballo:
color = blanco
raza = andaluz
tipo = de guerra
tamaño = grande
sexo = macho
velocidadMax = 65 km/hora

objeto2: otra instancia de la clase
Caballo:
color = negro
raza = griega
tipo = de carrera
tamaño = mediano
sexo = hembra
velocidadMax = 95 km/hora

OBJETOS

2.1 Clase y Objeto

- ✓ Clase es la representación abstracta general de uno o más objetos que se construirán a partir de ella.
- ✓ Objeto: Unidad atómica que encapsula propiedades y comportamientos y posee un estado específico con valores.

“Objeto es una instancia de una Clase”

Un objeto puede caracterizar una entidad física (caballo, coche, tarjeta) o no física(ecuación matemática, punto)

Ejercicio:

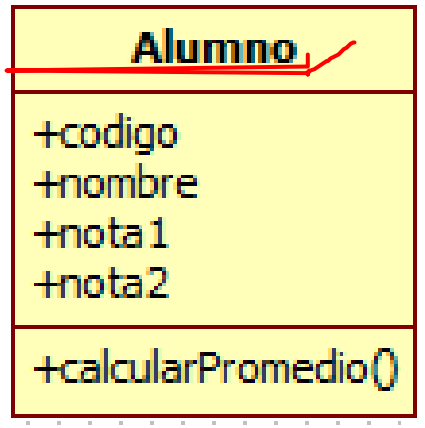
Plumon

Grupo de alumnos

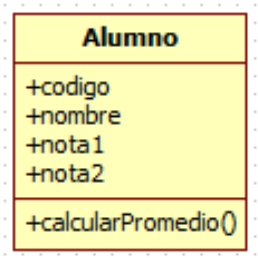
Necesitamos calcular el promedio de notas de cada alumno, sabiendo que poseen dos notas cada uno.

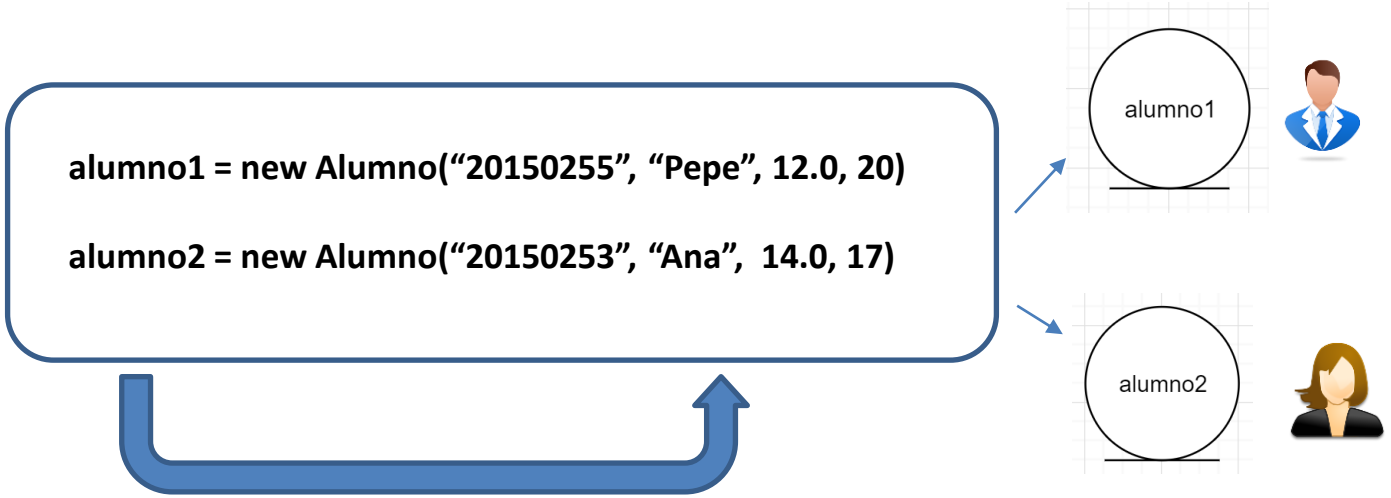
```java
public class Alumno {
    private int codigo;
    private String nombre;
    private int nota1;
    private int nota2;

    public Alumno(int codigo, String nombre, int nota1, int nota2) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }

    public double calcularPromedio(){
        double promedio = 0;
        promedio = (nota1+nota2)/2;
        return promedio;
    }
}
```

Constructor



Y como se crean los objetos en Java?






```
System.out.println(alumno1.calcularPromedio);
System.out.println(alumno2.calcularPromedio);
```

paquete

```java
package pe.upc.taller.entidades;

public class Alumno {
    private int codigo;
    private String nombre;
    private int nota1;
    private int nota2;

    public Alumno(int codigo, String nombre, int nota1, int nota2) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }

    public double calcularPromedio(){
        double promedio = 0;
        promedio = (nota1+nota2)/2;
        return promedio;
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getNota1() {
        return nota1;
    }

    public void setNota1(int nota1) {
        this.nota1 = nota1;
    }

    public int getNota2() {
        return nota2;
    }

    public void setNota2(int nota2) {
        this.nota2 = nota2;
    }
}
```

Estructura de un método

```
ámbito tipoDelMétodo nombreDelMétodo(tipo1 param1,
tipo2 param2, …) {



        return resultado;
}
```

ámbito: public, private o protected, en clases simples se usa public
tipoDelMétodo: El tipo de dato del valor que retorna, sino retorna será void.
tipo1, tipo2: El tipo de dato del parámetro respectivamente
return: palabra reservada que devuelve el valor calculado, podría obviarse si se trata de un solo proceso sin retorno.

Tipos de Datos y Jerarquía de Operadores

Tipo de variable	Bytes que ocupa	Rango de valores
boolean	2	true, false
byte	1	-128 a 127
short	2	-32.768 a 32.767
int	4	-2.147.483.648 a 2.147.483.649
long	8	$-9 \cdot 10^{18}$ a $9 \cdot 10^{18}$
double	8	$-1{,}79 \cdot 10^{308}$ a $1{,}79 \cdot 10^{308}$
float	4	$-3{,}4 \cdot 10^{38}$ a $3{,}4 \cdot 10^{38}$
char	2	Caracteres (en Unicode)

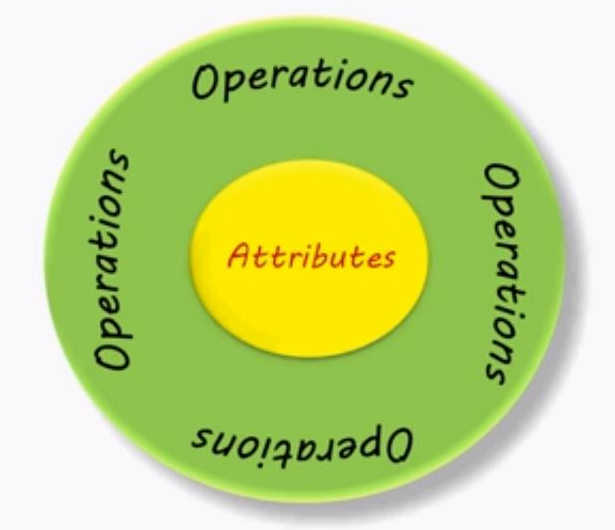
Asociatividad	Operadores por Orden de Precedencia
D-I	++expr --expr +expr -expr ~ ! (casting)
I-D	* / %
I-D	+ -
I-D	<< >> >>>
I-D	< > <= >= instanceof
I-D	== !=
I-D	&
I-D	^
I-D	|
I-D	&&
I-D	||
D-I	? :
D-I	= += -= *= /= %= &= ^= |= <<= >>= >

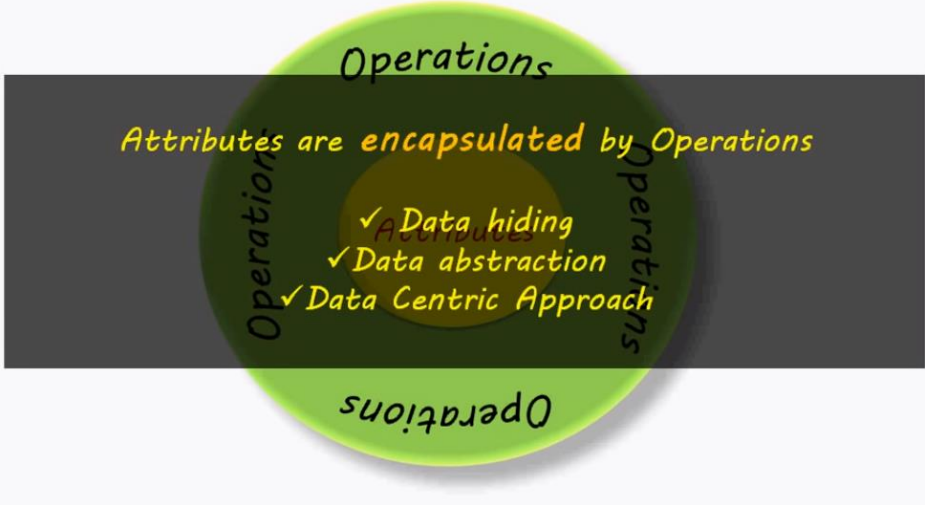
3. Terminología Básica

- **Atributo**: Propiedad, característica de un objeto, estado.
- **Comportamiento**: Acción que realiza un objeto y que será definido como método de su clase, también se le llama operación.
- **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.

4. Encapsulamiento

En las clases no es necesario que las demás clases sepan cuáles son los atributos y mucho menos manipularlos. Es por eso que se considera que los atributos deben ser privados (nadie accede directamente a ellos). Aún así necesitamos hacer que la clase haga lo que queremos que haga. Para ello lo que hacemos es acceder a su comportamiento, esto lo hacemos mediante los métodos (que suelen ser públicos).





5. Ventajas de la programación orientada a objetos

- Representa mejor a un sistema real.
- Permite crear e interpretar sistemas complejos de manera sencilla.
- Permite la construcción de prototipos
- Permite una mejor comunicación para el equipo
- Agiliza el desarrollo de software
- Facilita el mantenimiento del software
- Reusabilidad
- Extensibilidad

Gracias!