



# ESTRUCTURA DE DATOS


---



# ¿A Que Se Refiere Cuando Se Habla De Poo?

---


La Programación Orientada a Objetos (POO) es un paradigma de programación es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el **concepto de clases y objetos**. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.





# ¿Cuáles son los 4 componentes que componen POO?

## 4 componentes que componen POO:

1. **Clases:** Las clases pueden ser definidas como un molde que contendrá todas las características y acciones con las cuales podemos construir N cantidad de objetos.
  2. **Propiedades:** Las propiedades son las características de una clase, tomando como ejemplo la clase humanos, las propiedades podrían ser: nombre, el género, la altura, color de cabello, color de piel, etc.
  3. **Métodos:** Los métodos son las acciones que una clase puede realizar, siguiendo el mismo ejemplo anterior, estas podrían ser: caminar, comer, dormir, soñar, respirar, nadar, etc.
  4. **Objetos:** Son aquellos que tienen propiedades y comportamientos, estos pueden ser físicos o conceptuales.
- 



# ¿Cuáles son los pilares de POO?


---


**Abstracción:** Es cuando separamos los datos de un objeto para luego generar un molde (una clase).

**Encapsulamiento:** Lo puedes utilizar cuando deseas que ciertos métodos o propiedades sean inviolables o inalterables.

**Herencia:** Nos permite crear nuevas clases a partir de otras. Si tuviéramos una clase “Autos” y quisiéramos crear unas clases “Auto deportivo” o “Auto clásico”, podríamos tomar varias propiedades y métodos de la clase “Autos”. Esto nos da una jerarquía de padre e hijo.


**Polimorfismo:** Proviene de Poli = muchas, morfismo = formas. Se utiliza para crear métodos con el mismo nombre pero con diferente comportamiento.

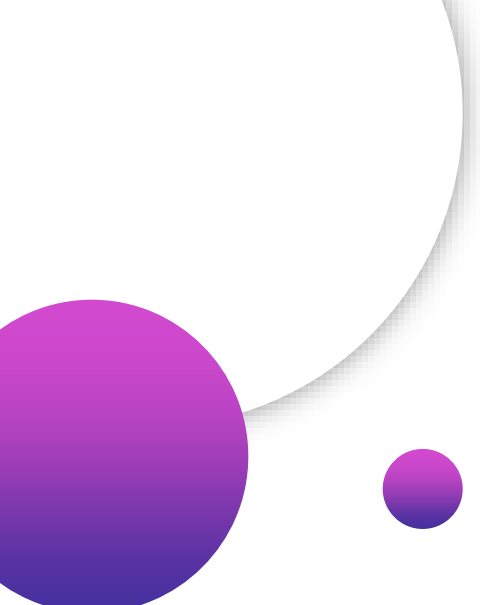




# ¿Qué es Encapsulamiento y muestre un ejemplo?

Un ejemplo del encapsulamiento podría ser una cuenta de banco, donde el usuario no puede simplemente aumentar su balance de dinero, si no que debe depender de unos métodos previamente validados para aumentar dicho balance






# . ¿Qué es Abstracción y muestre un ejemplo?

Abstraction es el concepto de programación orientada a objetos que shows sólo los atributos esenciales y hides información innecesaria. El objetivo principal de la abstracción es ocultar los detalles innecesarios a los usuarios

## Ejemplo.-

podemos saber que una mesa es una mesa más allá de si es cuadrada o redonda, de madera o de plástico, con 4, 3 o 6 patas.



# ¿Que es Herencia y muestre un ejemplo?

El cual es un mecanismo que permite derivar una clase a otra clase. En otras palabras, tendremos unas clases que serán hijos, y otras clases que serán padres. Las clases hijas pueden utilizar tanto sus métodos y propiedades como de la clase padre, siempre que su modificador de acceso lo permita.

```
class DosDimensiones{  
    double base;  
    double altura;  
    void mostrarDimension(){  
        System.out.println("La base y altura es:  
        "+base+" y "+altura);  
    }  
}
```



# ¿Qué es Polimorfismo y muestre un ejemplo?

En programación orientada a objetos, polimorfismo es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros (diferentes implementaciones) utilizados durante su invocación. Dicho de otro modo el objeto como entidad puede contener valores de diferentes tipos durante la ejecución del programa.

```
class Animal {  
    public void makeSound() {  
        System.out.println("Grr...");  
    }  
};  
class Cat extends Animal {  
    public void makeSound() {  
        System.out.println("Meow");  
    }  
}  
class Dog extends Animal {  
    public void makeSound() {  
        System.out.println("Woof");  
    }  
}
```





# Que es un ARRAY?

---

Los arrays se utilizan para agrupar objetos del mismo tipo.

De esta manera, podemos referirnos a este grupo con el mismo nombre. pero te lo puedes encontrar con muchos nombres:


- *Arreglos*
  - *Vectores*
  - *matrices*
- 



## ¿Qué son los paquetes en JAVA?

---

Los paquetes son el mecanismo que usa Java para facilitar la modularidad del código. Un paquete puede contener una o más definiciones de interfaces y clases, distribuyéndose habitualmente como un archivo. Para utilizar los elementos de un paquete es necesario importar este en el módulo de código en curso, usando para ello la sentencia `IMPORT`



# ¿Cómo se define una clase main en JAVA y muestra un ejemplo?

El método main() acepta un parámetro (y solo uno): una matriz de tipo String. Esta matriz recoge los valores que introduzcas a la hora de ejecutar tu aplicación desde la línea de comandos. Da igual el valor que introduzcas; el JRE lo transformará a String.

```
public class Main {  
    public static void main(String[] args) {  
  
        Scanner lectura=new Scanner(System.in);  
        int nPais =1;  
        Pais[] pais = new Pais[nPais];  
    }  
}
```

Generar la clase Provincia  
Crear una clase MAIN

- Crear todos los gets y sets de la clase.
- El constructor no recibe parámetros.
- Crear una instancia de la clase Provincia.
- Mostrar los datos de una provincia.

```
package Pais;

public class Provincia {

    private String nombre;

    //constructor sin parametros
    public Provincia(){
        this.nombre="";
    }

    public Provincia(String nombre){
        this.nombre=nombre;
    }

    //get obtner
    public String getNombre(){
        return this.nombre;
    }

    //set = establecer
    public void setNombre(String nuevoNombre){
        this.nombre=nuevoNombre;
    }

    //mostrar provincia
    public void mostrarProvincia(){
        System.out.println("----- Datos de Provincia -----");
        System.out.println("Nombre de provincia: "+getNombre());
    }

}
```

Generar la clase Departamento.  
Crear una clase MAIN (Utilizar el MAIN del anterior ejercicio)

- Crear todos los gets y sets de la clase.
- El constructor no recibe parámetros.
- Crear una instancia de la clase Departamento.
- Omitir el método agregaNuevaProvincia()
- Mostrar los datos de los departamentos.

```
public class Departamento {

    private String nombre;
    private Provincia[] nroProvincias;

    public Departamento() {
        this.nombre = "";
        this.nroProvincias = new Provincia[0];
    }
    public Departamento(String nombre, Provincia[] nroProvincias) {
        this.nombre = nombre;
        this.nroProvincias = nroProvincias;
    }
    //crear un metodo que ingrese una provincia

    public void agregaNuevaProvincia(Provincia[] nuevoNroProvincias) {
        this.nroProvincias=nuevoNroProvincias;
    }

    //get
    public String getNombre() {
        return this.nombre;
    }

    public Provincia[] getNroProvincias() {
        return this.nroProvincias;
    }
    //set
    public void setNombre(String nuevoNombre) {
        this.nombre = nuevoNombre;
    }
    //set
    public void setNroProvincias(Provincia[] nuevoNroProvincias){
        this.nroProvincias = nuevoNroProvincias;
    }

    //mostrar
    public void mostrarDepartamento(){
        System.out.println("----- Datos de departamento -----");
        System.out.println("Nombre del departamento: "+getNombre());

        for(int i=0;i <this.getNroProvincias().length;i++) {
            this.getNroProvincias()[i].mostrarProvincia();
        }
    }
}
```

**Generar la clase País.**  
**Crear una clase MAIN (Utilizar el MAIN del anterior ejercicio)**

- **Crear una instancia de la clase País**
- **El constructor no recibe parámetros.**
- **Crear una instancia de la clase Departamento.**
- **Omitir el método agregaNuevoDepartamento()**
  - **Mostrar los datos del País.**

```

Public class Pais {
    private String nombre;
    private int nroDepartamentos;

    private Departamento[] departamentos1;

    public Pais() {
        this.nombre = "";
        this.nroDepartamentos = 0;
        this.departamentos1 = new Departamento[0];
    }
    public Pais(String nombre, int nroDepartamentos, Departamento[]
departamentos1) {
        this.nombre = nombre;
        this.nroDepartamentos = nroDepartamentos;
        this.departamentos1 = departamentos1;
    }

    public void agregaNuevoDepartamento(Departamento[]
nuevoDepartamentos1) {
        this.departamentos1 = nuevoDepartamentos1;
    }
}

```

```

//get
public String getNombre() {
    return this.nombre;
}

public int getNroDepartamentos() {
    return this.nroDepartamentos;
}

public Departamento[] getDepartamentos1() {
    return this.departamentos1;
}

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

public void setNroDepartamentos(int nuevoNroDepartamentos) {
    this.nroDepartamentos = nuevoNroDepartamentos;
}

public void setDepartamentos(Departamento[] nuevoDepartamento) {
    this.departamentos1 = nuevoDepartamento;
}

public void mostrarPais() {
    System.out.println("-----Datos del pais-----");
    System.out.println("Nombre de departamentos: " + getNombre());
    System.out.println("Numero de departamentos: " + getNroDepartamentos());

    for (int i = 0; i < this.getDepartamentos1().length; i++){
        this.getDepartamentos1()[i].mostrarDepartamento();
    }
}
}

```

**Crear el diseño completo de las clases.**

**Crear todos gets y sets de cada clase.**

- **Implementar los métodos agregarNuevoDepartamento(), agregarNuevaProvincia(), es decir todos los métodos.**
- **El método agregarNuevoDepartamento permite ingresar un nuevo departamento a un país.**
- **El método agregarNuevaProvincia permite ingresar una nueva provincia a un departamento.**
  - **La clase Main debe mostrar lo siguiente:**
    - **Crear el PAÍS Bolivia**
    - **Al país Bolivia agregarle 3 departamentos.**
    - **Cada departamento deberá tener 2 provincias.**



```

public class Main {
    public static void main(String[] args) {

        Scanner lectura=new Scanner(System.in);
        int nPais =1;
        Pais[] pais = new Pais[nPais];

        for(int i=0;i<nPais;i++){
            System.out.println("Ingresar pais "+(i+1)+" ": ");
            String nombrePais = lectura.nextLine();

            int nDepartamento =3;
            Departamento[] departamentos = new Departamento[nDepartamento];

            for(int j=0;j<nDepartamento;j++){

                System.out.println("Ingresar departamento "+(j+1)+" ": ");
                String nombreDepartamento = lectura.nextLine();

                int nProvincia=2;
                Provincia[] provincias = new Provincia[nProvincia];

                for(int k=0;k<nProvincia;k++){

                    System.out.println("Ingresar provincia "+(k+1)+" ": ");
                    String nombrePovincia = lectura.nextLine();

                    Provincia pr1 = new Provincia();
                    pr1.setNombre(nombrePovincia);
                    provincias[k] = pr1;

                }
            }
        }
    }
}

```

```

Departamento dep1 = new Departamento();
    dep1.setNombre(nombreDepartamento);
    dep1.setNroProvincias(provincias);
    departamentos[j] = dep1;

    dep1.mostrarDepartamento();
}

Pais pais1 = new Pais();
pais1.setNombre(nombrePais);
pais1.setNroDepartamentos(nDepartamento);
pais1.setDepartamentos(departamentos);
pais[i] = pais1;

    pais1.mostrarPais();
}
}
}

```

```

Ingresar pais 1:
Bolivia
Ingresar departamento 1:
Santa Cruz
Ingresar provincia 1:
Andres Ibañez
Ingresar provincia 2:
obispo santiestevan
----- Datos de departamento -----
Nombre del departamento: Santa Cruz
----- Datos de Provincia -----
Nombre de provincia: Andres Ibañez
----- Datos de Provincia -----
Nombre de provincia: obispo santiestevan
Ingresar departamento 2:
la paz
Ingresar provincia 1:
Caranavi
Ingresar provincia 2:
Manco Kapac
----- Datos de departamento -----
Nombre del departamento: la paz
----- Datos de Provincia -----
Nombre de provincia: Caranavi
----- Datos de Provincia -----
Nombre de provincia: Manco Kapac
Ingresar departamento 3:
Tarija
Ingresar provincia 1:
Gran chaco
Ingresar provincia 2:
Aviles

```

```

----- Datos de departamento -----
Nombre del departamento: Tarija
----- Datos de Provincia -----
Nombre de provincia: Gran chaco
----- Datos de Provincia -----
Nombre de provincia: Aviles
----- Datos del pais -----
Nombre de departamentos: Bolivia
Numero de departamentos: 3
----- Datos de departamento -----
Nombre del departamento: Santa Cruz
----- Datos de Provincia -----
Nombre de provincia: Andres Ibañez
----- Datos de Provincia -----
Nombre de provincia: obispo santiestevan
----- Datos de departamento -----
Nombre del departamento: la paz
----- Datos de Provincia -----
Nombre de provincia: Caranavi
----- Datos de Provincia -----
Nombre de provincia: Manco Kapac
----- Datos de departamento -----
Nombre del departamento: Tarija
----- Datos de Provincia -----
Nombre de provincia: Gran chaco
----- Datos de Provincia -----
Nombre de provincia: Aviles

Process finished with exit code 0

```