


# ESTRUCTURA DE DATOS

FREDDY MACHACA MAMANI

The background is black with several thin white lines of varying lengths and orientations. A small cyan square is located in the upper right quadrant. The text '01' is rendered in a large, bold, cyan font, with the '0' having a segmented, digital appearance.

01

# PARTE TEORICA

# QUE ES POO

Conocido como programación orientado a objetos es un paradigma de programación que parte del concepto de objetos



# Pilares del POO

## ABSTRACCIÓN

Cuando separamos datos de un objeto para generar un molde (clase)

## ENCAPSULAMIENTO

Se utiliza para ciertos método o propiedad inalterables

## HERENCIA

Nos permite crear nuevas clases a partir de otras ya existentes

## POLIMORFISMO

Se utiliza para crear métodos con el mismo nombre pero con diferente comportamiento

# Encapsulamiento

Es un principio fundamental y consiste en ocultar el estado interno del objetivo y obliga que toda interacción se realice a través de los métodos del objeto

```
public class Jugador {  
  
    3 usages  
    private String nombre;  
  
    3 usages  
    private String apellido;  
  
    3 usages  
    private String ci;  
  
    3 usages  
    private int edad;  
}
```

```
public String getNombre() { return this.nombre; }  
  
2 usages  
public String getApellido() {  
    System.out.println("Este es el apellido");  
    return this.apellido;  
}  
  
1 usage  
public String getCi() { return this.ci; }  
  
1 usage  
public int getEdad() { return this.edad; }  
  
5 usages  
public void setNombre(String nuevoNombre) { this.nombre = nuevoNombre; }  
6 usages  
public void setApellido(String nuevoApellido) {  
    this.apellido = nuevoApellido;  
}  
  
5 usages  
public void setCi(String nuevoCi) { this.ci = nuevoCi; }  
5 usages  
public void setEdad(int nuevaEdad) {  
    this.edad = nuevaEdad;  
}
```

# Abstracción

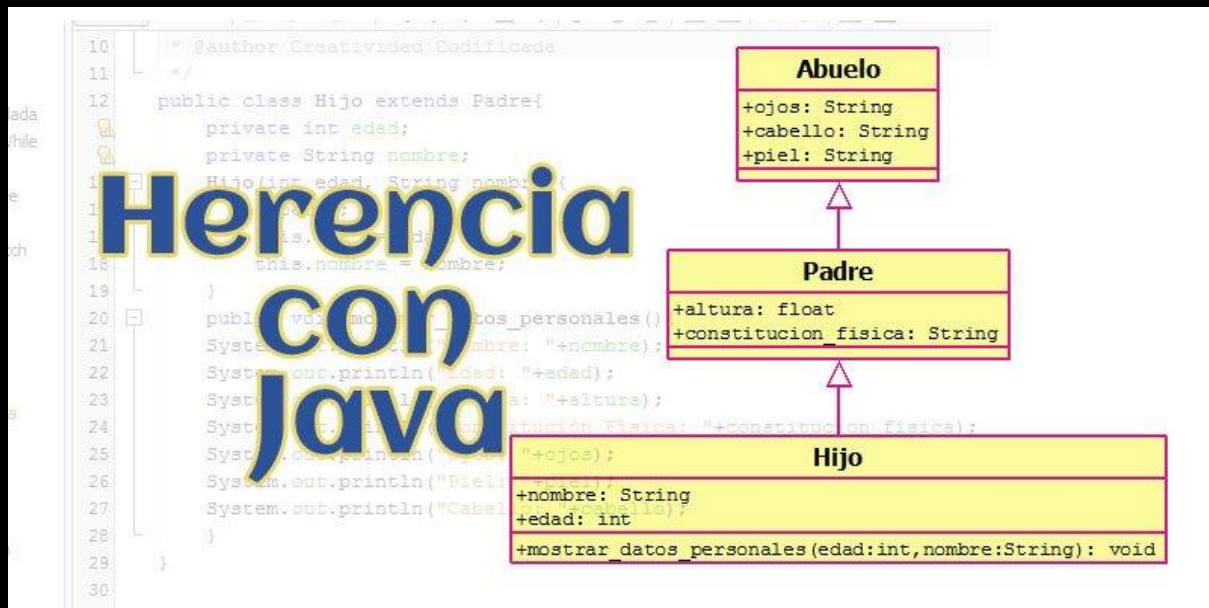
La abstracción consiste en seleccionar datos de un conjunto más grande para mostrar solo los detalles relevantes del objeto. Ayuda a reducir la complejidad y el esfuerzo de programación. En Java, la abstracción se logra usando clases e interfaces abstractas.

Si se quisiera crear una aplicación bancaria y se le pide que recopile toda la información sobre su cliente. Se debe seleccionar solo la información útil para su aplicación bancaria de ese grupo. Datos como nombre, dirección, información fiscal, etc. tienen sentido para una aplicación bancaria

- ☒ **Nombre completo**
- ☒ **dirección**
- ☒ **Número de contacto**
- ☒ **Información sobre los impuestos**

# Herencia

La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos.



# Polimorfismo

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 Overload
{
    void demo (int a)
    {
        System.out.println ("a: " + a);
    }
    void demo (int a, int b)
    {
        System.out.println ("a and b: " + a + ", " + b);
    }
    double demo(double a) {
        System.out.println("double a: " + a);
        return a*a;
    }
}
class MethodOverloading
{
    public static void main (String args [])
    {
        Overload Obj = new Overload();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```



# ARRAY

Una array o arreglo es una colección de variables del mismo tipo, a la que se hace referencia por un nombre común.

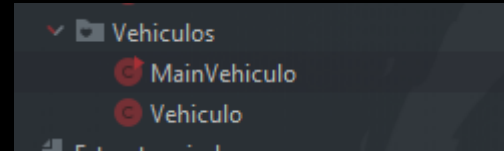
```
java.util.Arrays;  
class Main {  
    public static void main(String a[]) {  
        int a[]={1,2,3,4,5};  
        int b[]=new int[a.length];  
  
        //copying one array to another  
        b=Arrays.copyOf(a,a.length);  
  
        //printing array  
        for(int i=0;i<b.length;++i){  
            System.out.print(b[i]+" ");  
        }  
    }  
}
```

# Paquetes en Java

Un Paquete en Java es un contenedor de clases que permite agrupar las distintas partes de un programa y que por lo general tiene una funcionalidad y elementos comunes, definiendo la ubicación de dichas clases en un directorio de estructura jerárquica.



# Método Main




El método main o “clase principal” es un elemento de las clases de Java que permite que se pueda ejecutar un programa. Generalmente se expresa entre paréntesis () e incluye una matriz de tipo String.

```
public class MainVehiculo {
    public static void main(String[] args) {
        // instanciando vehiculo desde el constructor sin parametros
        Vehiculo v1= new Vehiculo();

        v1.6Derecha();
        v1.CambiarCarril();
        v1.frenar();
        //instanciando un vehiculo con todos los parametros
        Vehiculo v2= new Vehiculo( nombre: "Hyonda", color: "rojo", modelo: "fast", tamaño: "4x4", marca: "Nissan");
        v2.mover();
        //crear un metodo que permite mostrar todos los parametros
        v2.MostrarVehiculo();

        v2.setColor("AZUL");
        v2.MostrarVehiculo();
        v2.GetNombreVehiculo();
    }
}
```

The background is black with several thin white lines of varying lengths and orientations. A small cyan square is located in the upper right quadrant. The text '02' is rendered in a large, cyan, pixelated font on the left side.

02

# PARTE PRACTICA

# POO EN JAVA

01

Clase Provincia

Diseño y construcción de la clase

02

Clase Departamento

Diseño y construcción de la clase

03

Clase Pais

Diseño y construcción de la clase

04

Diseño Completo de las partes

Diseño completo de la clase

01

# 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



//INSTANCIAR PROVINCIA

```
Provincia p1 = new Provincia( nombre: "Achocalla");
Provincia p2 = new Provincia( nombre: "Dentista");
Provincia p3 = new Provincia( nombre: "Tiradentes");
Provincia p4 = new Provincia( nombre: "Guarulhos");
```

```
public class Provincia {

    4 usages
    private String nombre;

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

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

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

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

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

}
```

## 02

# 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.

Departamento	
(m) Departamento ()	
(m) Departamento (String, Provincia [])	
(f) nroProvincias	Provincia []
(f) nombre	String
(m) setNroProvincias (Provincia [])	void
(m) setNombre (String)	void
(m) getNombre ()	String
(m) mostrarDepartamento ()	void
(m) agregaNuevaProvincia (Provincia [])	void
(m) getNroProvincias ()	Provincia []

//INSTANCIAR DEPARTAMENTOS

```
Departamento dep1 = new Departamento( nombre: "La Paz", pro1);
Departamento dep2 = new Departamento( nombre: "Oruro", pro2);
```

```
public Departamento() {
    this.nombre = "";
    this.nroProvincias = new Provincia[0];
}

2 usages
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
1 usage
public String getNombre() {
    return this.nombre;
}

2 usages
public Provincia[] getNroProvincias() {
    return this.nroProvincias;
}

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

//set
1 usage
public void setNroProvincias(Provincia[] nuevoNroProvincias){
    this.nroProvincias = nuevoNroProvincias;
}

//mostrar
2 usages
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();
    }
}
```

# 03

## 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.

Pais	
m	Pais(String, int, Departamento [])
m	Pais()
f	nroDepartamentos int
f	nombre String
f	departamentos1 Departamento []
m	getDepartamentos1() Departamento []
m	setNroDepartamentos(int) void
m	agregaNuevoDepartamento (Departamento []) void
m	getNombre() String
m	getNroDepartamentos() int
m	setDepartamentos (Departamento []) void
m	setNombre (String) void
m	mostrarPais() void

//INSTANCIAR PAIS

```
Pais pais1 = new Pais();
pais1.agregaNuevoDepartamento(dep);
pais1.mostrarPais();
```

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

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

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

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

//set
1 usage
public void setNombre(String nuevaNombre) { this.nombre = nuevaNombre; }

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

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

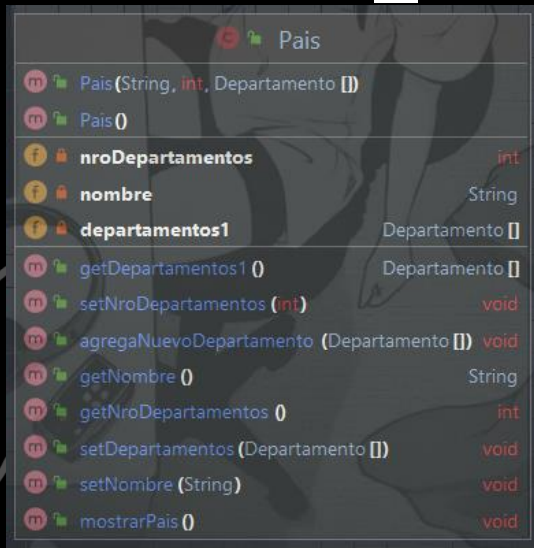
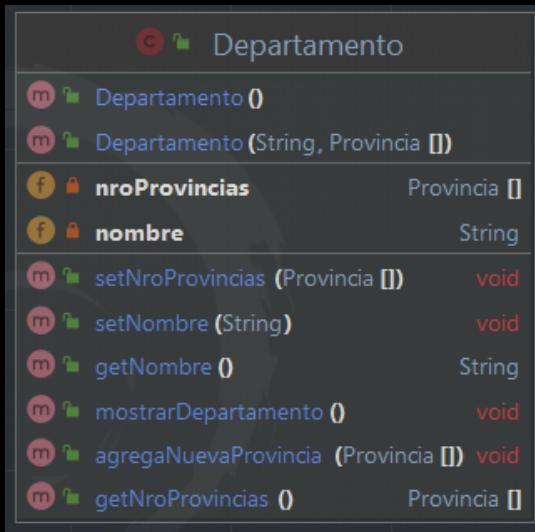
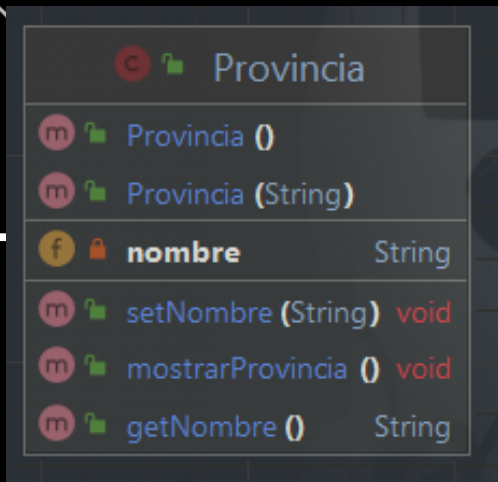
2 usages
public void mostrarPais() {
    System.out.println("-----Datos del pais-----");
    System.out.println("Nombre de departamentos: " + getNombre());
    System.out.println("Nro de departamentos: " + getNroDepartamentos());

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



# 04

## Diseño completo



- 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.

```

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

```

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

04

```
public class Main {  
    public static void main(String[] args) {  
        //INSTANCIAR PROVINCIA  
        Provincia p1 = new Provincia( nombre: "Achocalla");  
        Provincia p2 = new Provincia( nombre: "Dentista");  
        Provincia p3 = new Provincia( nombre: "Tiradentes");  
        Provincia p4 = new Provincia( nombre: "Guarulhos");  
  
        Provincia[] pro1 = new Provincia[2];  
        pro1[0] = p1;  
        pro1[1] = p2;  
  
        Provincia[] pro2 = new Provincia[2];  
        pro2[0] = p3;  
        pro2[1] = p4;  
  
        //INSTANCIAR DEPARTAMENTOS  
  
        Departamento dep1 = new Departamento( nombre: "La Paz", pro1);  
        Departamento dep2 = new Departamento( nombre: "Oruro", pro2);  
  
        Departamento[] dep = new Departamento[2];  
        dep[0] = dep1;  
        dep[1] = dep2;  
  
        //INSTANCIAR PAIS  
        Pais pais1 = new Pais();  
        pais1.agregaNuevoDepartamento(dep);  
        pais1.mostrarPais();  
    }  
}
```

# Quemado

```
-----Datos del pais-----  
Nombre de departamentos:  
Nro de departamentos: 0  
----- Datos de departamento -----  
Nombre del departamento: La Paz  
----- Datos de Provincia -----  
Nombre de provincia: Achocalla  
----- Datos de Provincia -----  
Nombre de provincia: Dentista  
----- Datos de departamento -----  
Nombre del departamento: Oruro  
----- Datos de Provincia -----  
Nombre de provincia: Tiradentes  
----- Datos de Provincia -----  
Nombre de provincia: Guarulhos
```

# 04

La clase Main debe mostrar lo siguiente:

- Crear el PAÍS Bolivia
- Al país Bolivia agregarle 3 departamentos.
- Cada departamento deberá tener 2 provincias.

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

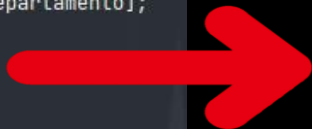
for(int i=0;i<nPais;i++){
    System.out.println("Ingresar pais "+(i+1)+": ");
    String nombrePais = lectura.nextLine();

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

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

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

    pais1.mostrarPais();
}
```



```
System.out.println("Ingresar departamento "+(i+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[i] = dep1;

dep1.mostrarDepartamento();
```

04

Ingresar pais 1:

Bolivia

Ingresar departamento 1:

Santa cruz

Ingresar provincia 1:

a

Ingresar provincia 2:

b

----- Datos de departamento -----

Nombre del departamento: Santa cruz

----- Datos de Provincia -----

Nombre de provincia: a

----- Datos de Provincia -----

Nombre de provincia: b

Ingresar departamento 2:

Oruro

Ingresar provincia 1:

c

Ingresar provincia 2:

d

----- Datos de departamento -----

Nombre del departamento: Oruro

----- Datos de Provincia -----

Nombre de provincia: c

----- Datos de Provincia -----

Nombre de provincia: d

Ingresar departamento 3:

La Paz

Ingresar provincia 1:

e

Ingresar provincia 2:

f

-----Datos del pais-----

Nombre de departamentos: Bolivia

Nro de departamentos: 3

----- Datos de departamento -----

Nombre del departamento: Santa cruz

----- Datos de Provincia -----

Nombre de provincia: a

----- Datos de Provincia -----

Nombre de provincia: b

----- Datos de departamento -----

Nombre del departamento: Oruro

----- Datos de Provincia -----

Nombre de provincia: c

----- Datos de Provincia -----

Nombre de provincia: d

----- Datos de departamento -----

Nombre del departamento: La Paz

----- Datos de Provincia -----

Nombre de provincia: e

----- Datos de Provincia -----

Nombre de provincia: f