

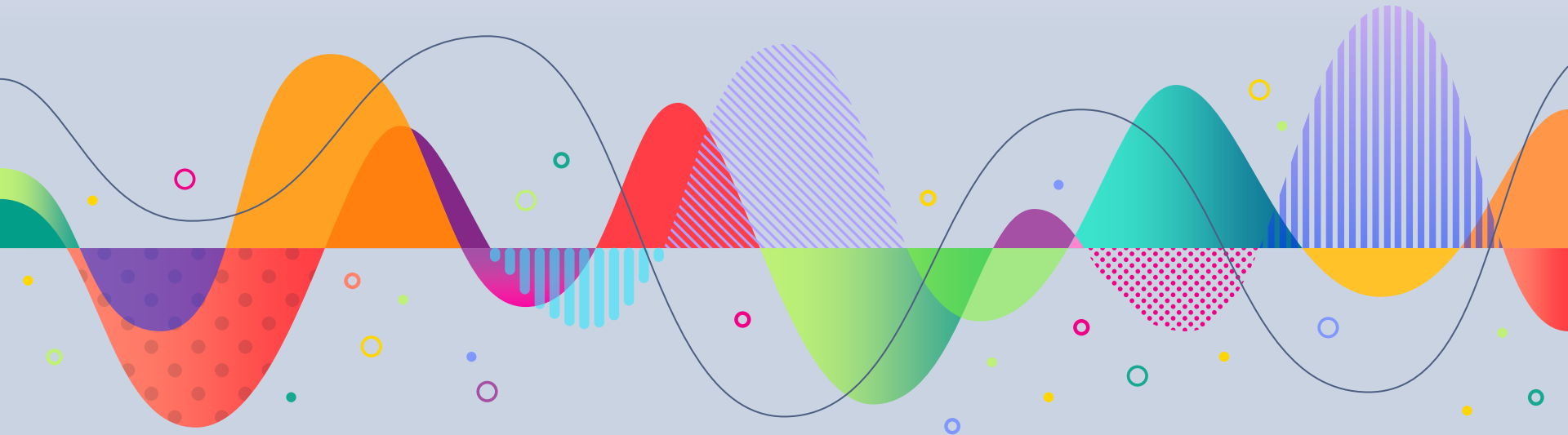
# EVALUACION PROCESUAL

## HITO 2

ESTRUCTURA DE DATOS

NILBER MAYTA CUNO

# PARTE TEORICA





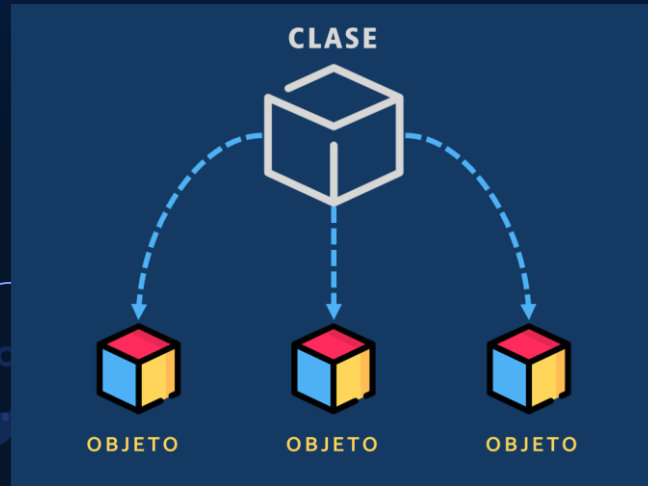
## *1. ¿A que se refiere cuando se habla de POO?*

La POO o programacion orientada a objetos es un estilo de programacion que se centra en utilizar los elementos del Sistema como objtos dividiendolos en clases.

# OBJETOS



Esta es muy util ya que nos permite la reutilizacion de codigo. Ademas de que interpretaremos todo como un objeto con identidad, estado, comportamiento.



## *2. ¿Cuáles son los 4 componentes que componen POO?*



Tenemos como los cuatro pilares  
de la POO son:

Clases, Propiedades, Metodos

Objetos



## Componentes



### Clases

Son un molde donde que tiene todas las características donde podremos crear N objetos.

### Propiedades

Son las propiedades de la clase que lo describen

# Componentes



## Metodos

Son las acciones que una clase puede realizar que va de la mano con las propiedades

## Objetos

Son los que tiene n propiedades y comportamientos, técnicamente son instancias de las clases.

### 3. ¿Cuáles son los pilares de POO?



Tenemos como pilares a la abstraccion,  
encapsulamiento, herencia y polimorfismo.  
Sin los cuales la programacion orientada a objetos  
no seria posible.



## 4. ¿Qué es Encapsulamiento y muestre un ejemplo?

Es la función de poder almacenar datos dentro de los objetos y de darles una funcionalidad.

```
1  public class Carro
2  {
3
4      public string Marca;
5
6      public int AñoSalidaAlMercado { get; set; }
7
8      private int Velocidad { get; set; }
9
10     public void Acelerar()
11     {
12         Velocidad += 10;
13     }
14
15 }
```

## 5. ¿Qué es Abstracción y muestre un ejemplo?

Es una manera de abstraer mediante una clase una entidad del mundo real.

```
1 public class Carro
2 {
3
4     public string Marca;
5
6     public int AñoSalidaAlMercado { get; set; }
7
8     public void Acelerar()
9     {
10    }
11 }
```

## 6. ¿Que es Herencia y muestre un ejemplo?

Es la capacidad de compartir código a otras instancias ahorrando trabajo.

```
public class Carro: Vehículo
{
    public void EncenderRadio()
    {
        Console.WriteLine("Encendiendo la radio");
    }
}

public class Camión: Vehículo
{
    public override void Reversa()
    {
        base.Reversa();
        Console.WriteLine("BEEP BEEP BEEP!");
    }
}
```

# 7. ¿Qué es Polimorfismo y muestre un ejemplo?



Tenemos como polimorfismo a un derivado de la herencia que al igual que la herencia nos ayudara a poder reutilizar código pero cambiando en ciertos aspectos.

```
static void Reparar(Vehículo vehículo)
{
    Console.WriteLine("Iniciando reparación");
    Console.WriteLine("Probando acelerador");
    Console.WriteLine($"Velocidad inicial {vehículo.Velocidad}");
    vehículo.Acelerar();
    Console.WriteLine($"Velocidad final {vehículo.Velocidad}");
    Console.WriteLine("Probando reversa");
    vehículo.Reversa();
    Console.WriteLine("Listo!");
}
```

## 8. Que es un ARRAY?



La definicion de un array o arreglo es una variable que pobra almacenar varios datos en el mediante un sistema de casillas

## 9. ¿Qué son los paquetes en JAVA?



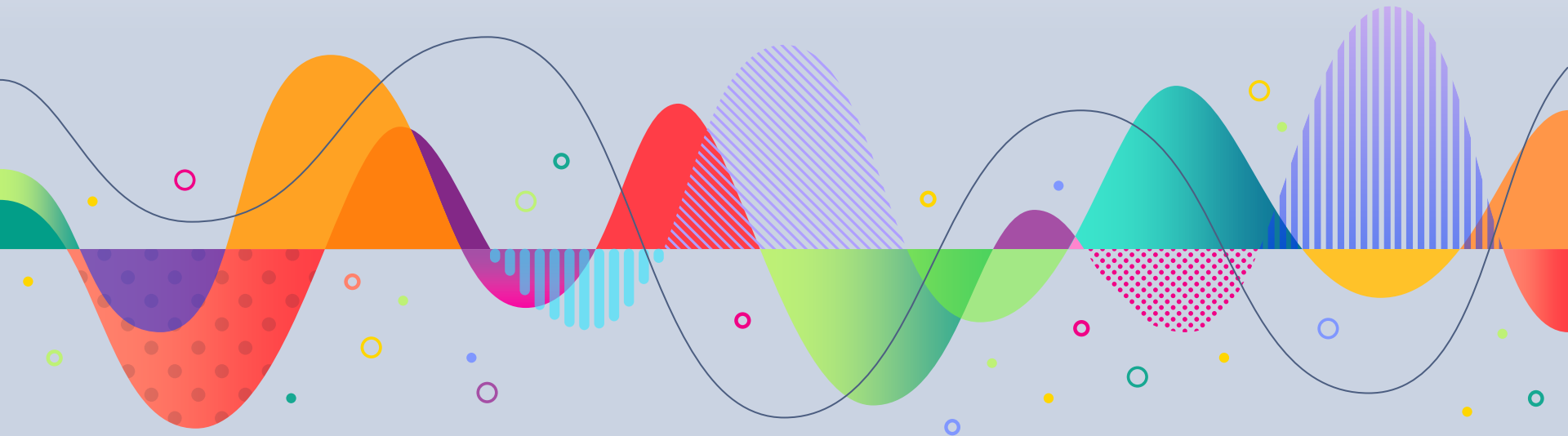
Un Paquete en Java es un contenedor de clases que permite agrupar las distintas partes de un programa y que por lo general tiene una funcionalidad y elementos comunes, definiendo la ubicación que tendrá

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

Para poder realizar un archive main o una clase ejecutable se pone el siguiente código.

```
public class Main {  
    no usages  
    public static void main(String[] args) {  
  
    }  
}
```

# PARTE PRACTICA





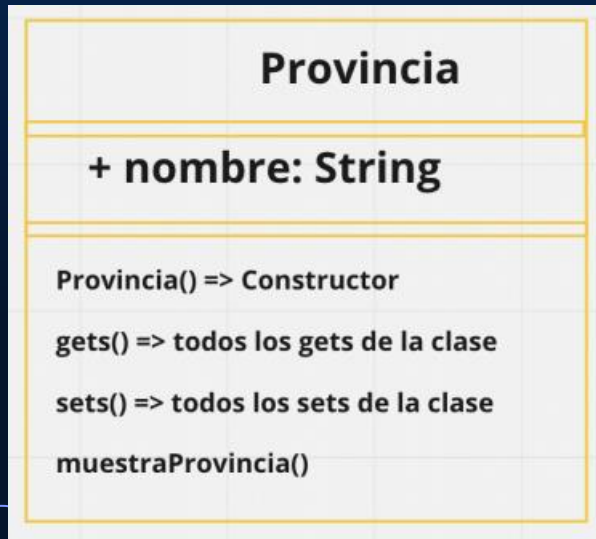
# 11. Generar la clase Provincia.



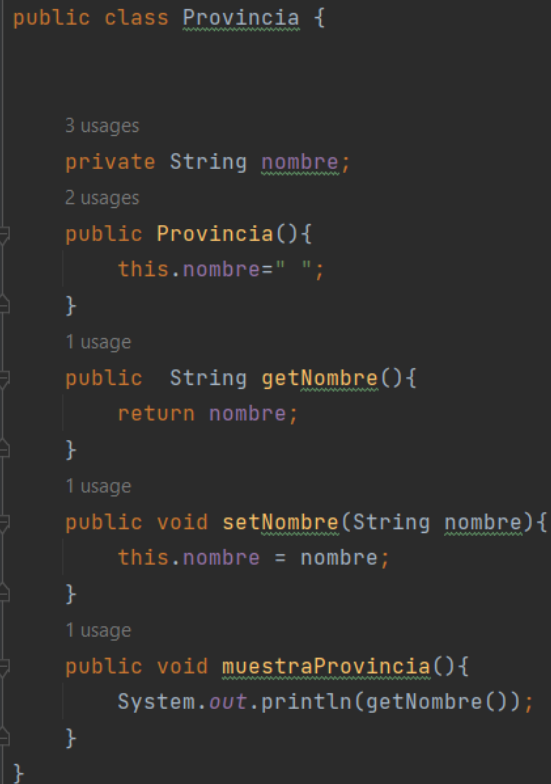
- Diseño
  - Crear una clase MAIN
    - Crear todos los gets y sets de la clase.
    - El constructor no recibe parámetros.
    - Crear una instancia de la clase Provincia
    - Mostrar los datos de una provincia

# 11. Generar la clase Provincia.

## ○ Diseño



# 11. Generar la clase Provincia.



```
public class Provincia {  
  
    3 usages  
    private String nombre;  
    2 usages  
    public Provincia(){  
        this.nombre=" ";  
    }  
    1 usage  
    public String getNombre(){  
        return nombre;  
    }  
    1 usage  
    public void setNombre(String nombre){  
        this.nombre = nombre;  
    }  
    1 usage  
    public void muestraProvincia(){  
        System.out.println(getNombre());  
    }  
}
```

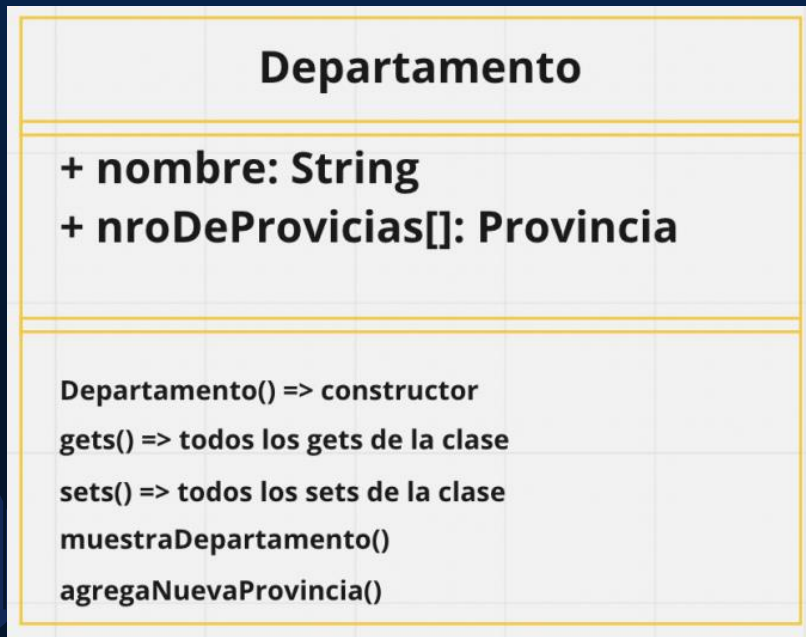
# 12. Generar la clase Departamento.



- Diseño
  - 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.

# 12. Generar la clase Departamento.

## ○ Diseño



# 12. Generar la clase Departamento.



```
public class Departamento {  
  
    3 usages  
    private String nombre;  
    3 usages  
    private Provincia[] nroProvincia;  
    2 usages  
    public Departamento() {  
  
        Provincia[] nroProvincia = new Provincia[0];  
        this.nombre = " ";  
        this.nroProvincia = nroProvincia;  
    }  
    1 usage  
    public String getNombre(){  
  
        return nombre;  
    }  
}
```

```
    public String getNombre(){  
  
        return nombre;  
    }  
    no usages  
    public Provincia[] getNroDeProvincias(){  
  
        return nroProvincia;  
    }  
    1 usage  
    public void setNombre(String nombre){  
  
        this.nombre = nombre;  
    }  
    no usages  
    public void setNroDeProvincias(Provincia[] nroDeProvincias)  
    {  
  
        this.nroProvincia = nroDeProvincias;  
    }  
}
```

# 13. Generar la clase País.



- Diseño
- 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.

# 13. Generar la clase País.

## ○ Diseño

Pais
<b>+ nombre: String</b> <b>+ nroDepartamentos: Int</b> <b>+ departamentos[]: Departamento</b>
 Pais() => Constructor gets() => todos los gets de la clase sets() => todos los sets de la clase muestraPais() agregaNuevoDepartamento()

Pais		
m	Pais()	
f	departamentos	Departamento []
f	nombre	String
f	nroDEpartamentos	int
m	setDepartamentos (Departamento [])	void
m	getDepartamentos ()	Departamento []
m	setNroDEpartamentos (int)	void
m	getNroDEpartamentos ()	int
m	setNombre (String)	void
m	agregarNuevoDepartamento (String)	void
m	getNombre ()	String
m	muestraPais ()	void



# 13. Generar la clase País.

```
public class Pais {  
  
    3 usages  
    private String nombre;  
    3 usages  
    private int nroDEpartamentos;  
    3 usages  
    private Departamento[] departamentos;  
  
    1 usage  
    public Pais(){  
        Departamento[] nroDepartamentos = new Departamento[0];  
        this.nombre = " ";  
        this.nroDEpartamentos = 0;  
        this.departamentos = nroDepartamentos;  
    }  
  
    1 usage  
    public String getNombre(){  
  
        return nombre;  
    }  
  
    1 usage  
    public int getNroDEpartamentos() { return nroDEpartamentos; }  
    2 usages  
    public Departamento[] getDepartamentos(){  
  
        return departamentos;  
    }  
}
```

```
public void setNombre(String nombre) { this.nombre= nombre; }  
2 usages  
public void setNroDEpartamentos(int nroDEpartamentos) { this.nroDEpartamentos=nroDEpartamentos; }  
1 usage  
public void setDepartamentos(Departamento[] departamentos){  
  
    this.departamentos=departamentos;  
}  
  
1 usage  
public void muestraPais(){  
  
    System.out.println(getNombre());  
  
    for(int i=0; i < this.getNroDEpartamentos() ; i++){  
        System.out.println(" DEPARTAMENTO:" +this.getDepartamentos()[i].getNombre());  
    }  
}
```

```
public void agregarNuevoDepartamento(String newDepartamento ){  
  
    Departamento[] oriDepartamento = this.getDepartamentos();  
    Departamento[] nuevoDepartamento = new Departamento[oriDepartamento.length+1];  
  
    nuevoDepartamento[oriDepartamento.length] = new Departamento();  
    nuevoDepartamento[oriDepartamento.length].setNombre(newDepartamento);  
  
    setNroDEpartamentos(nuevoDepartamento.length);  
  
    for(int i=0; i<oriDepartamento.length;i++){  
        nuevoDepartamento[i] = new Departamento();  
        nuevoDepartamento[i] = oriDepartamento[i];  
    }  
    setDepartamentos(nuevoDepartamento);  
}
```

# 14. Crear el diseño completo de las clases.

- Diseño
  - 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.

# 13. Generar la clase País.



Pais	Departamento	Provincia
<b>+ nombre: String</b> <b>+ nroDepartamentos: Int</b> <b>+ departamentos[]: Departamento</b>	<b>+ nombre: String</b> <b>+ nroDeProvincias[]: Provincia</b>	<b>+ nombre: String</b>
Pais() => Constructor gets() => todos los gets de la clase sets() => todos los sets de la clase muestraPais() agregaNuevoDepartamento()	Departamento() => constructor gets() => todos los gets de la clase sets() => todos los sets de la clase muestraDepartamento() agregaNuevaProvincia()	Provincia() => Constructor gets() => todos los gets de la clase sets() => todos los sets de la clase muestraProvincia()

# 13. Generar la clase País.

Pais	Departamento	Provincia
<b>Pais()</b>	<b>Departamento ()</b>	<b>Provincia ()</b>
f <b>departamentos</b> Departamento []	f <b>nombre</b> String	f <b>nombre</b> String
f <b>nombre</b> String	f <b>nroProvincia</b> Provincia []	m <b>getNombre ()</b> String
f <b>nroDepartamentos</b> int	m <b>muestraDepartamento (int)</b> void	m <b>setNombre (String)</b> void
m <b>setDepartamentos (Departamento [])</b> void	m <b>setNombre (String)</b> void	m <b>muestraProvincia ()</b> void
m <b>getDepartamentos ()</b> Departamento []	m <b>agregaNuevaProvincia (String)</b> void	
m <b>setNroDepartamentos (int)</b> void	m <b>setNroDeProvincias (Provincia [])</b> void	
m <b>getNroDepartamentos ()</b> int	m <b>getNombre ()</b> String	
m <b>setNombre (String)</b> void	m <b>getNroDeProvincias ()</b> Provincia []	
m <b>agregarNuevoDepartamento (String)</b> void		
m <b>getNombre ()</b> String		
m <b>muestraPais ()</b> void		

# 13. Generar la clase País.



```
public class Main {  
    no usages  
    public static void main(String[] args) {  
  
        Pais ps = new Pais();  
        ps.setNombre("BOLIVIA");  
        ps.setNroDEpartamentos(2);  
  
        ps.agregarNuevoDepartamento(newDepartamento: "La Paz");  
        ps.agregarNuevoDepartamento(newDepartamento: "Cochabamba");  
        ps.agregarNuevoDepartamento(newDepartamento: "Santa Cruz");  
  
        ps.muestraPais();  
    }  
}
```