POO

Programación Orientada a Objetos

Pilares

1. Herencia: Clase que hereda, recibe tos los atribitos de la clase padre.

```
public class Persona{
    private String nombre;
    private String correo;

    //Set
    //Get
    //Methods Implementations
    public void Mostrar(){
        this.nombre = "Diana";
        this.correo = "diana@corhuila.edu.co";
        System.out.println(
                            "Nombre: "+this.nombre+"\n" +
                            "Correo: "+this.correo+"\n"
                            );
    }
}

public class Estudiante extends Persona{
    //here atribute
    //here methods
}

public class Profesor extends Persona{
    //here atribute
    //here methods
}
```

Dónde Estudiante tiene acceso a los atributos de Persona

2. Encapsulamiento : Corresponde a dos capas, una set que permite manipular los datos desde otra clase y el get, que permite obtener los datos de los atributos también desde otra clase.

Nota: Recodar la privacidad de los atributos, ellos deben estar en private.

```
public class Persona{
    private String nombre;
    private String correo;

    //Set
```

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

    public void setCorreo(String correo){
        this.correo = correo;
    }

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

    public String getCorreo(){
        return this.correo;
    }


    //Methods Implementations
    public void Mostrar(){
        System.out.println(
                            "Nombre: "+this.getNombre()+"\n" +
                            "Correo: "+this.getCorreo()+"\n"
                            );
    }
}

public class Estudiante extends Persona{
    //here atribute
    //here methods
}

public class Profesor extends Persona{
    //here atribute
    //here methods
}
```

3. Polimorfismo: Es la posibilidad de un métodos comportarse de deferente manera, cuando se hereda la clase padre, la clase hija, mostrará su propio comportamiento.

```
public class Persona{
    private String nombre;
    private String correo;

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

    public void setCorreo(String correo){
        this.correo = correo;
```

```
        }

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

        public String getCorreo(){
            return this.correo;
        }


        //Methods Implementations
        public void Mostrar(){
            System.out.println(
                                "Nombre: "+this.getNombre()+"\n" +
                                "Correo: "+this.getCorreo()+"\n"
                                );
        }
    }

    public class Estudiante extends Persona{

        @Override
        public void Mostrar(){
            System.out.println("Estudiante: "+this.getNombre()+" -
email"+this.getCorreo()+"\n");
        }
    }

    public class Profesor extends Persona{

        @Override
        public void Mostrar(){
            System.out.println(
                                "Data ******: "+this.getNombre()+" :
"+this.getCorreo()+"\n");
        }
    }
```

4. Abstración : Este tipo de clase se caracteriza por ser heredada por muchas clases, es decir, es una clase padre, por lo tanto, no denería permitirse la creación de un objeto sobre ella.

Ejemplo: - Clase padre : Persona (nombre, edad, tipo_documento, documento, genero) - Clase hija : Cliente - Clase hija : Proveedor - Clase hija : Estudiante - Clase hija : Profesor - Clase hija : Médido - Clase hija : N

El anterior ejemplo, se puede demostrar que Persona es una clase padre, esta tiene en atributos que comparte para muchas clases hijas.

```
    public abstract class Persona{
        private String nombre;
```

```
        private String correo;

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

        public void setCorreo(String correo){
            this.correo = correo;
        }

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

        public String getCorreo(){
            return this.correo;
        }


        //Methods Implementations
        public void Mostrar(){
            System.out.println(
                                "Nombre: "+this.getNombre()+"\n" +
                                "Correo: "+this.getCorreo()+"\n"
                                );
        }
    }

    public class Estudiante extends Persona{
        private  String codigo;

        public String getCodigo() {
            return codigo;
        }

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

        @Override
        public void Mostrar(){
            System.out.println("Estudiante: "+this.getCodigo()+" -
"+this.getNombre()+" - email"+this.getCorreo()+"\n");
        }
    }

    public class Profesor extends Persona{
        private String tipoContrato;

        public String getTipoContrato() {
            return tipoContrato;
        }
```

```
        public void setTipoContrato(String tipoContrato) {
            this.tipoContrato = tipoContrato;
        }

        @Override
        public void Mostrar(){
            System.out.println(
                    "Data ******: "+this.getNombre()+" : "+this.getCorreo()+"-
profesor de "+this.getTipoContrato()+". \n");
        }
    }
```

Probar escenarios.

1. Probando herencia

```
    public class Ejecutar{
        public static void main (String[] Arg){
            //Mostrar data
            System.out.println("Probando escenario 1.");

            Persona persona = new Persona();
            persona.Mostrar();

            Estudiante estudiante = new Estudiante();
            estudiante.Mostrar();
        }
    }
```

2. Probando encapsulamiento

```
    public class Ejecutar{
        public static void main (String[] Arg){
            persona.setNombre("Wendy");
            persona.setCorreo("wendy@Corhuila.edu.co");
            persona.Mostrar();

            System.out.println("Clase Hija Estudiante");
            Estudiante estudiante = new Estudiante();
            estudiante.setNombre("Alvaro");
            estudiante.setCorreo("alvaro@corhuila.edu.co");
            estudiante.Mostrar();

            System.out.println("Clase Hija Profesor");
            Profesor profesor = new Profesor();
            profesor.setNombre("Diana");
            profesor.setCorreo("diana@corhuila.edu.co");
```

```
            profesor.Mostrar();
        }
    }
```

3. Probando polimorfismo

```
public class Ejecutar{
    public static void main (String[] Arg){
       System.out.println("Probando escenario 1.");

        System.out.println("Clase Padre");
        Persona persona = new Persona();
        persona.setNombre("Wendy");
        persona.setCorreo("wendy@Corhuila.edu.co");
        persona.Mostrar();

        System.out.println("Clase Hija Estudiante");
        Estudiante estudiante = new Estudiante();
        estudiante.setNombre("Alvaro");
        estudiante.setCorreo("alvaro@corhuila.edu.co");
        estudiante.Mostrar();

        System.out.println("Clase Hija Profesor");
        Profesor profesor = new Profesor();
        profesor.setNombre("Diana");
        profesor.setCorreo("diana@corhuila.edu.co");
        profesor.Mostrar();
    }
}
```

4. Probando polimorfismo

```
public class Ejecutar{
    public static void main (String[] Arg){
        //Mostrar data
        System.out.println("Probando escenario 1.");

        System.out.println("Clase Hija Estudiante");
        Estudiante estudiante = new Estudiante();
        estudiante.setNombre("Alvaro");
        estudiante.setCorreo("alvaro@corhuila.edu.co");
        estudiante.setCodigo("2025A-12");
        estudiante.Mostrar();

        System.out.println("Clase Hija Profesor");
        Profesor profesor = new Profesor();
        profesor.setNombre("Diana");
        profesor.setCorreo("diana@corhuila.edu.co");
        profesor.setTipoContrato("tiempo completo");
```

```
            profesor.Mostrar();
        }
    }
```