Objetivo:

Aplicar el concepto de abstracción para el diseño de clases que integran una solución, utilizando el encapsulamiento para proteger la información y ocultar la implementación.

Introducción:

Composición

La Composición es un tipo de relación de alto grado de dependencia entre la clase contenedora (*Persona*) y las clases que la componen (*Cabeza*, *Tórax*, *Pierna*, etc.), es decir, cuando se crea una instancia de la clase contenedora, deben crearse, como parte de su conformación, instancias de los objetos que la componen, ya que no tendría sentido, desde el punto de vista de la instancia de *Persona*, conformar una persona sin cabeza o sin un tórax; por otro lado, durante toda la vida del objeto de la clase *Persona* debe existir el objeto de la clase *Cabeza* por la misma razón que antes. En este mismo sentido, tampoco tendría congruencia el tener una instancia de la clase *Cabeza* que nunca haya estado relacionada con una *Persona*.

Algún lector inconforme podría argumentar que sí es posible que un objeto de la clase *Persona* deje de contener a uno de la clase *Pierna,* por ejemplo, sin afectar la existencia de primero, y en efecto: es posible. Bastaría entonces con analizar si conviene más que la clase *Pierna* se modele mejor como Agregación, pero finalmente esto es sólo un ejemplo y la idea principal considero que ha sido plasmada, lo último sería más cuestión de análisis y de la conveniencia dada para un sistema determinado tal y como sucede en la vida real.

Agregación.

La Agregación, por otro lado, es un tipo de relación con un bajo grado de dependencia. Así, por ejemplo, una instancia de la clase Persona, puede tener o no, durante su tiempo de vida (pero no es preciso que lo tenga desde su creación), un atributo de la clase Ropa sin que ello afecte su propia existencia; al mismo tiempo que un objeto de la clase Ropa podría existir independientemente de si es agregado a una Persona o a un Maniquí (clase que no aparece en el diagrama), por ejemplo.

En este tipo de relación, la existencia de los objetos involucrados es independiente, lo mismo que su tiempo de vida: un objeto de la clase Ropa podría seguir existiendo más allá del tiempo de vida del de una

Persona y viceversa, sin que ninguno de los dos se vea afectado.

Asociación.

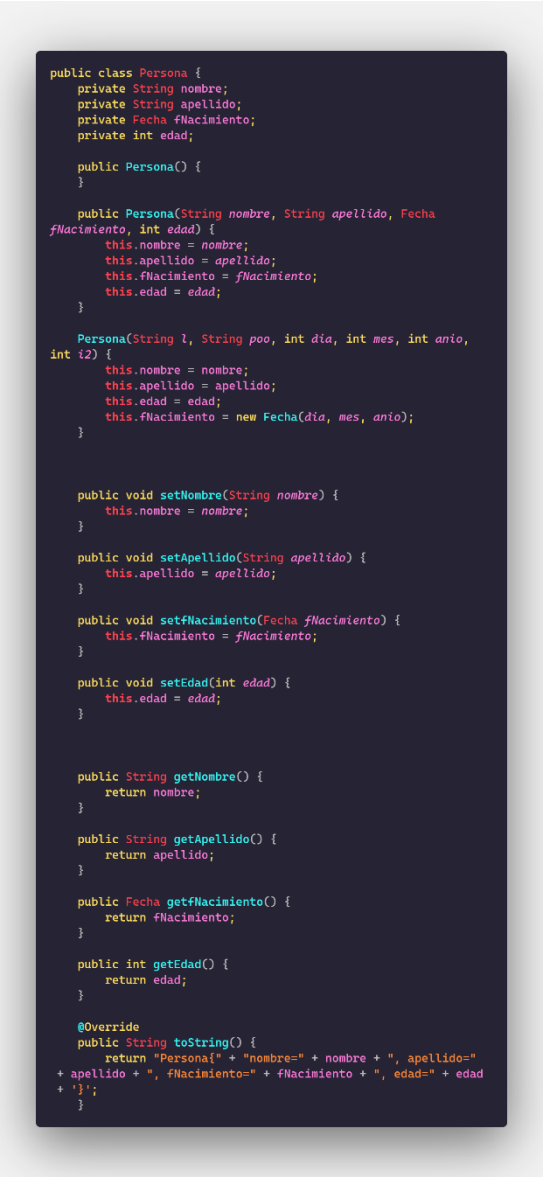
Una Asociación es aún menos dependiente en relación y tiempo. Espero que el lector coincida conmigo en que, si bien la ropa no es imprescindible para la existencia de una persona, sí es necesaria; mientras que una tarjeta de crédito podría ser útil, en el mejor de los casos necesaria, pero en definitiva prescindible, es decir, una Persona podría pasar toda su vida sin tener la necesidad de ninguna Tarjeta de Crédito, mientras que otras podrían tener muchas de ellas.

Actividades:

Obtener las características y funcionalidades principales de un objeto dado.

Se creo la clase persona

Para ejemplificar contiene sus propios atributos



```
public class Persona {
    private String nombre;
    private String apellido;
    private Fecha fNacimiento;
    private int edad;

    public Persona() {
    }

    public Persona(String nombre, String apellido, Fecha
fNacimiento, int edad) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.fNacimiento = fNacimiento;
        this.edad = edad;
    }

    Persona(String l, String poo, int dia, int mes, int anio,
int i2) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.fNacimiento = new Fecha(dia, mes, anio);
    }



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

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public void setfNacimiento(Fecha fNacimiento) {
        this.fNacimiento = fNacimiento;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }



    public String getNombre() {
        return nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public Fecha getfNacimiento() {
        return fNacimiento;
    }

    public int getEdad() {
        return edad;
    }

    @Override
    public String toString() {
        return "Persona{" + "nombre=" + nombre + ", apellido="
+ apellido + ", fNacimiento=" + fNacimiento + ", edad=" + edad
+ '}';
    }
```

Y se le añade una fecha de nacimiento con otra clase que la compone con

sus atributos

```java
public class Fecha {
    private int dia;
    private int mes;
    private int anio;

    public Fecha() {
    }

    public Fecha(int dia, int mes, int anio) {
        this.dia = dia;
        this.mes = mes;
        this.anio = anio;
    }

    public int getDia() {
        return dia;
    }

    public int getMes() {
        return mes;
    }

    public int getAnio() {
        return anio;
    }

    public void setDia(int dia) {
        this.dia = dia;
    }

    public void setMes(int mes) {
        this.mes = mes;
    }

    public void setAnio(int anio) {
        this.anio = anio;
    }

    @Override
    public String toString() {
        return dia + "/" + mes + "/" + anio;
    }
}
```

Al declarar el objeto persona con la composición queda de esta forma

```java
Persona p1 = new Persona();
                p1.setNombre("lucio");
                p1.setApellido("ramos");
                Fecha f1 = new Fecha(2, 3, 1998);
                p1.setfNacimiento(f1);
                System.out.println("preson: " + p1);
                System.out.println(f1);
                System.out.println("##############");
                Fecha f2 = new Fecha(2, 3, 1997);
                Persona p2 = new Persona("Luci", "POO", f2, 25);
                System.out.println("##############");
                Persona p3 = new Persona("lu", "POO", new Fecha(3, 3, 2000),
22);
                System.out.println("##############");
                Persona p4 = new Persona("l", "POO", 4, 8, 2000, 60);
                System.out.println(p4);
```