UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Sesión 3:

Relaciones de asociación, agregación y composición

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Indicador de Logro

Al finalizar la sesión, el estudiante demuestra la capacidad de explicar claramente las relaciones entre clases en un modelo orientado a objetos, identificando correctamente ejemplos de asociación, agregación y composición,en la solución de problemas usando Java.

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Utilidad

Un **diagrama de clases** es importante porque nos permitirá representar gráficamente y de manera estática la estructura general de un sistema, mostrando cada una de las clases y sus interacciones (como herencias, asociaciones, agregaciones, composiciones). Los **diagramas de clases** son el pilar fundamental del *modelado con UML*, siendo ampliamente utilizados tanto para *análisis como para diseño de sistemas y software en general*.

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

RECORDANDO LA CLASE ANTERIOR

- **clases y objetos**
- **constructores**

Agregación y Composición en POO

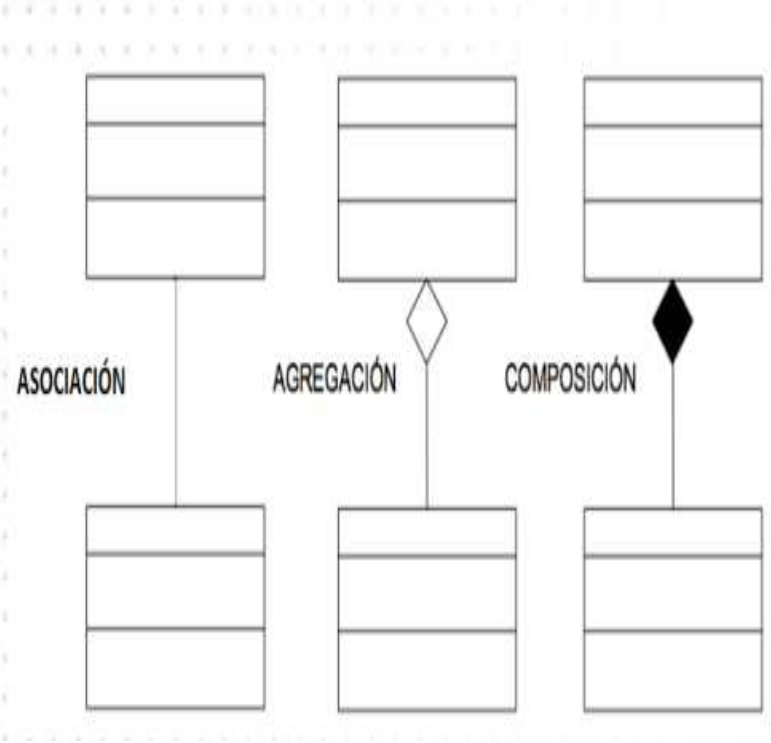
https://www.youtube.com/watch?v=U9-iM-gA7-E

Definición de relaciones

Una relación es un vinculo entre dos objetos, este vinculo, se presenta por que, ambos objetos, se necesitan mutuamente para lograr la realización de una actividad o la realización de unos servicios. Estas relaciones nacen por la misma dinámica social o por que la naturaleza así lo estableció.

Que nunca se me olvide:

LAS RELACIONES DEPENDEN DEL CONTEXTO
"Dominio del Problema"



Relación de dependencia entre clases

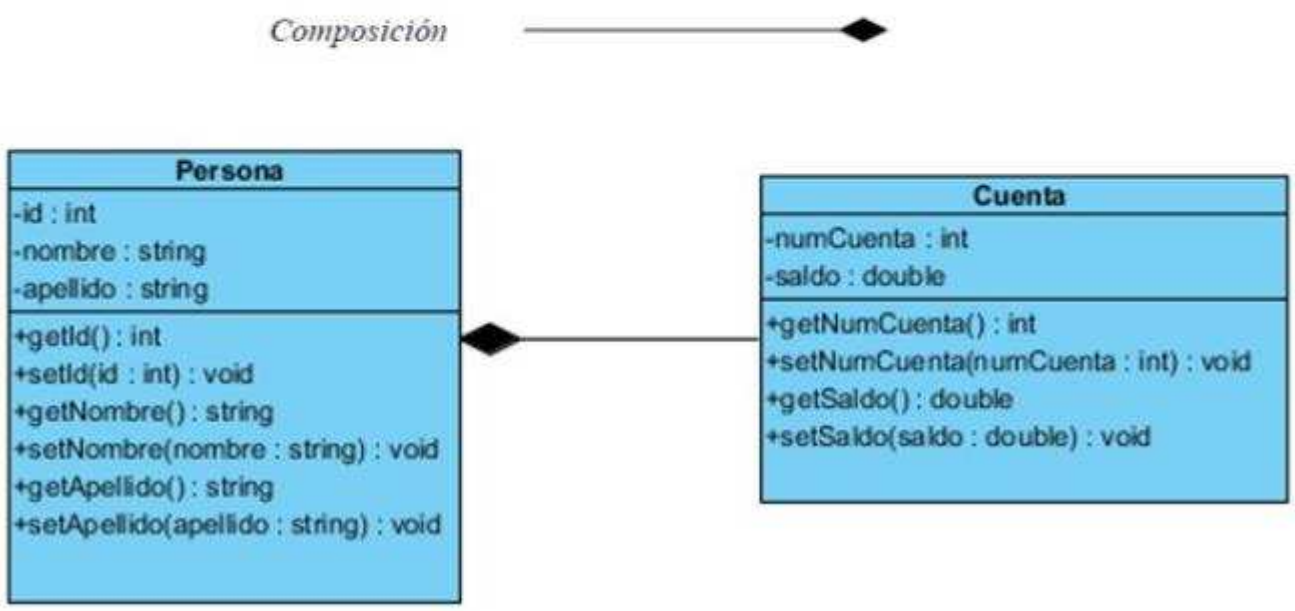
Composición (Definición)

- Se constituye entre el ***todo*** y la ***parte***.
- La clase ***todo*** controla la existencia de las clases ***parte.*** Mientras existe el ***todo=*** existe la ***parte***.
- Al crear un objeto del ***todo*** se crea un objeto de la ***parte***
- Objeto A ***tiene un/posee un/contiene un*** objeto de la clase B
- Los objetos suelen estar compuestos de conjuntos de objetos más pequeños; un coche es un conjunto de motor y carrocería, un motor es un conjunto de piezas, y así sucesivamente.
- Si se elimina el ***todo*** se eliminan las ***partes***.
- Contiene un atributo, que puede ser una colección y además de ello la clase que contiene la colección debe tener un método que agregue los elementos a la colección.

Relación de dependencia entre clases

Composición (Notación)

- La notación para representar las composiciones, es una línea con un rombo lleno en el extremo apuntando a la clase **todo**



Relación de dependencia entre clases

Composición (Implementación)

```
Public class Persona {
        private int id;
        private String nombre;
        private String apellido;
        public  Cuenta cuenta = new Cuenta();

        public int getId() {
        return id;
        }
        public void setId(int id) {
        this.id=id;
        }
        public string getNombre() {
        return nombre;
        }
        public void setNombre(String nombre) {
        this.nombre=nombre;
        }
        public string getApellido() {
        return apellido;
        }
        public void setApellido(String apellido) {
        this.apellido=apellido;
        }
        public int getCuenta() {
        return cuenta;
        }
        public void setCuenta(Cuenta cuenta) {
        this.cuenta=cuenta;
        }
```

```
public  class Cuenta {
        private int numCuenta;
        private double saldo;

        public  int getNumCuenta() {
        return NumCuenta;
        }

        public  void setNumCuenta(int numCuenta)
        this.numCuenta=numCuenta;
        }

        public  double getSaldo() {
        return saldo;
        }

        public  void setSaldo(double saldo) {
        this.saldo=saldo;
        }
}
```

Relación de dependencia entre clases

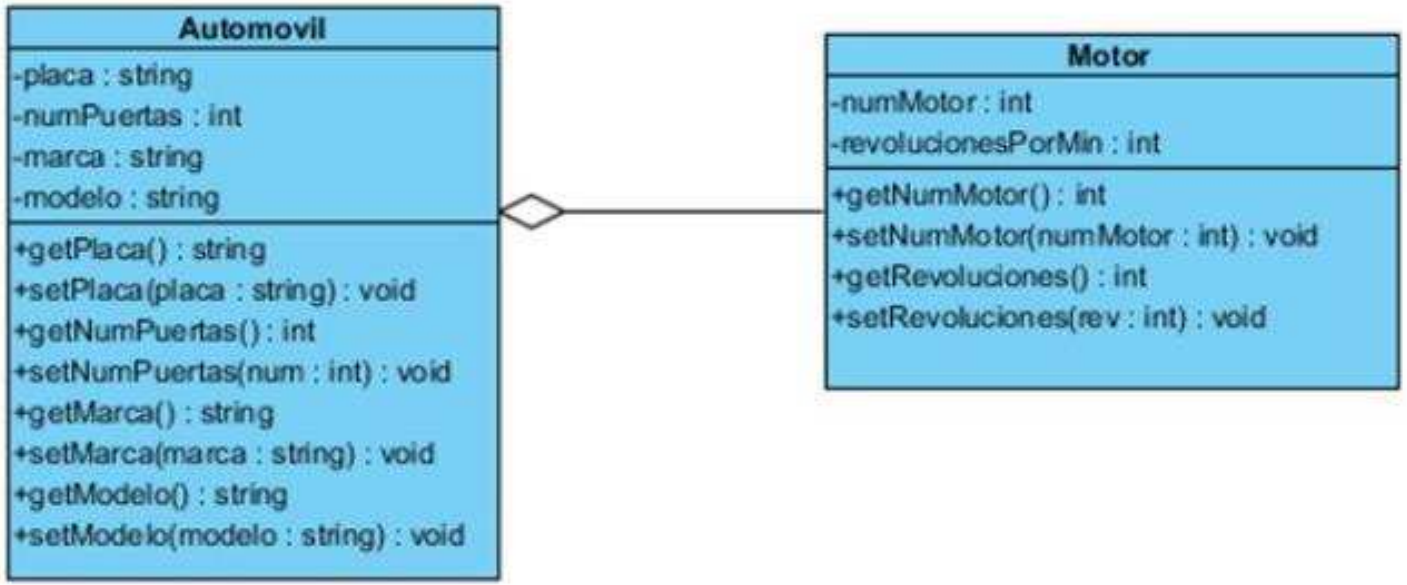
Agregación (Definición)

- Existe una relación de agregación si en el enunciado del problema hay expresiones de la forma "tiene_un", "es_parte_de"...
- Se constituye entre el ***todo*** y la ***parte***.
- La existencia del ***todo*** no condiciona la existencia de la ***parte***.
- Si se elimina el ***todo*** NO se eliminan las ***partes***.
- Contiene un atributo, que puede ser una colección, es decir un array, vector, etc, y además de ello la clase que contiene la colección debe tener un método que agregue los elementos a la colección.

Relación de dependencia entre clases

Agregación (Notación)

- La notación para representar las agregaciones, es una línea con un rombo vacío en el extremo apuntando a la clase **todo**



Relación de dependencia entre clases

Agregación (Implementación)

```
public class Automovil {
        private String placa;
        private int numPuertas;
        private String marca;
        private String modelo;
        public Motor motor;

        public String getPlaca() {
        }

        public void setPlaca(String placa) {
        }

        public int getNumPuertas() {
        }

        public void setNumPuertas(int num) {
        }

        public String getMarca() {
        }

        public void setMarca(String marca) {
        }

        public String getModelo() {
        }

        public void setModelo(String modelo) {
        }
```

```
public class Motor {
        private int numMotor;
        private int revolucionesPorMin;

        public int getNumMotor() {
        }

        public void setNumMotor(int numMotor) {
        }

        public int getRevoluciones() {
        }

        public void setRevoluciones(int rev) {
        }
}
```

Programación Orientada a Objetos

Conclusiones:

- Las relaciones son fundamentales en POO.
- Permiten modelar sistemas complejos de manera eficiente.
- Un buen entendimiento mejora la calidad del diseño de software.

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Actividad

Implementa los ejemplos 1,2.

UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Cierre

¿Qué hemos aprendido hoy?