RELACIONES ENTRE CLASES

En la P.O.O. en Python, las relaciones entre clases permiten modelar como os objetos interactúan entre si en un sistema. Existen tres relaciones fundamentales que definen como una clase puede relacionarse con otra.

1. **Asociación:** es una relación general entre dos clases, donde una clase utiliza a otra. Esta relación no implica propiedad ni dependencia fuerte.

Características:

- Los objetos están relacionados, pero son independientes.
- Puede ser unidireccional o bidireccional.
- Se vasa en el uso de instancias de otra clase.

Ejemplo:

Clase: profesor

Atributos: nombre

Acción:

Objetos:

```
class Profesor:
    def __init__(self,nombre):
        self.nombre = nombre


class Curso:
    def __init__(self,nombre,profesor):
        self.nombre = nombre
        self.profesor = profesor


prof = Profesor("Dr.Morillos")
curso = Curso("Muestreo",prof)
print(curso.profesor.nombre)
```

2. **<u>Agregación:</u>** Es un tipo especial de asociación. Se refiere a una relación de “todo-parte”, donde una clase (el todo) contiene a otras (las partes), pero las partes pueden existir independientemente del todo.

Características:

- Relación “tiene un” (has a).
- El objeto contenido no es destruido si el objeto contenedor desaparece.
- Los objetos se pueden reutilizar en otras partes.

Ejemplo:

Clase: departamento

Atributo: nombre

Acción: agregar_departamento

Objeto:

```
class Departamento:
    def __init__(self, nombre):
        self.nombre = nombre


class Universidad:
    def __init__(self, nombre):
        self.nombre = nombre
        self.departamentos = []
    def agregar_departamento(self, departamento):
        self.departamentos.append(departamento)
    def mostrar_departamentos(self):
        print(f"Universidad: {self.nombre}")
        print("Departamentos:")
        for dep in self.departamentos:
            print(f"- {dep.nombre}")
dep1 = Departamento("Ingeniería Estadística")
dep2 = Departamento("Informática")
uni = Universidad("Universidad Nacional del Altiplano")
uni.agregar_departamento(dep1)
uni.agregar_departamento(dep2)
uni.mostrar_departamentos()
```

3. **<u>Composición</u>**: Es un caso más fuerte que la agregación. También es una relación “todo-parte”, pero con propiedad total y dependencia de ciclo de vida.
   Si el objeto contenedor se destruye, sus partes también.

   Características:
   - Relación fuerte de pertenencia.
   - Se crean y destruyen como el objeto contenedor.

Ejemplo:

Clase: motor ,auto Objeto:miauto=auto()

Atributo: tipo,marca

Acción:arrancar

```
class Motor:
    def __init__(self,tipo):
        self.tipo = tipo

    def encender(self):
        print(f"Motor : {self.tipo} encendido")

class Auto:
    def __init__(self,marca):
        self.marca = marca
        self.motor = Motor("Electrico")

    def arrancar(self):
        print(f"Auto : {self.marca} arrancando")
        self.motor.encender()

miAuto = Auto("Tesla")
miAuto.arrancar()
miAuto = Auto("Toyota")
miAuto.arrancar()
```

Ejemplo N°4:

clase: estudiante

atributo: nombre, dni, código_estudiante

acción:

objeto:

```
class Estudiante:
    def __init__(self, nombre, dni, codigo_estudiante):
        self.nombre = nombre
        self.dni = dni
        self.codigo_estudiante = codigo_estudiante
        self.cursos = []
    def inscribirse(self, curso):
        self.cursos.append(curso)
        curso.agregar_estudiante(self)
    def mostrar_informacion(self):
        print(f"\nEstudiante: {self.nombre} , DNI: {self.dni} ,Código: {self.codigo_estudiante}")
        print("Cursos inscritos:")
        for curso in self.cursos:
            print(f" - {curso.nombre_curso}")
class Profesor:
    def __init__(self, nombre, dni, especialidad):
        self.nombre = nombre
        self.dni = dni
        self.especialidad = especialidad
    def mostrar_informacion(self):
        print(f"Profesor: {self.nombre} ,DNI: {self.dni} , Especialidad: {self.especialidad}")
class Curso:
    def __init__(self, nombre_curso, profesor):
        self.nombre_curso = nombre_curso
        self.profesor = profesor
        self.estudiantes = []
```

```
    def agregar_estudiante(self, estudiante):
        if estudiante not in self.estudiantes:
            self.estudiantes.append(estudiante)

    def mostrar_detalles(self):
        print(f"\nCurso: {self.nombre_curso}")
        print("Profesor:")
        self.profesor.mostrar_informacion()
        print("Estudiantes inscritos:")
        for est in self.estudiantes:
            print(f" - {est.nombre} ({est.codigo_estudiante})")



class Universidad:
    def __init__(self, nombre):
        self.nombre = nombre
        self.cursos = []

    def agregar_curso(self, curso):
        self.cursos.append(curso)

    def mostrar_cursos(self):
        print(f"\nUniversidad: {self.nombre}")
        for curso in self.cursos:
            curso.mostrar_detalles()



prof1 = Profesor("Ing. Juan Carlos", "01323043", "Programación")
prof2 = Profesor("Dr. Luis alberth","12038945", "programador")
prof3 = Profesor("Ing. Jose Tito", "01348893", "Programación")
prof4 = Profesor("Ing. confesor", "01290738", "estadistico")
prof5 = Profesor("Ing. Fred Torres", "01344579", "Programación")
prof5 = Profesor("Ing. Elvis", "01323451", "estadistico")
```

```
  curso1 = Curso("Lenguaje de Programación II", prof1)
curso2 = Curso("Analisis y Diseño de sistemas de informacion", prof2)
curso3 = Curso("Sistema de Gestion de Base de Datos", prof3)
curso4 = Curso("Modelos Discretos", prof4)
curso5 = Curso("Programacion Numerica", prof5)
curso6 = Curso("Inferencia Estadistica", prof6)

est1 = Estudiante("Milena Kely", "12345678", 2025007)
est2 = Estudiante("Henrry Quispe Ramos", "98765432", 2025078)
est3 = Estudiante("leydy Griselda", "74057981",2401198)

uni = Universidad("Universidad Nacional del Altiplano")
uni.agregar_curso(curso1)
uni.agregar_curso(curso2)
uni.agregar_curso(curso3)
uni.agregar_curso(curso4)
uni.agregar_curso(curso5)
uni.agregar_curso(curso6)

est1.inscribirse(curso1)
est1.inscribirse(curso2)
est2.inscribirse(curso5)
est3.inscribirse(curso2)
est1.inscribirse(curso4)
est2.inscribirse(curso6)

uni.mostrar_cursos()
est1.mostrar_informacion()
est2.mostrar_informacion()
est3.mostrar_informacion()
```