

# COMPOSICIÓN Y AGREGACIÓN



# Ejercicio 1.

## COMPOSICIÓN

### 1. Sean las siguientes clases:

Habitación<nombre, tamaño (en metros cuadrados)

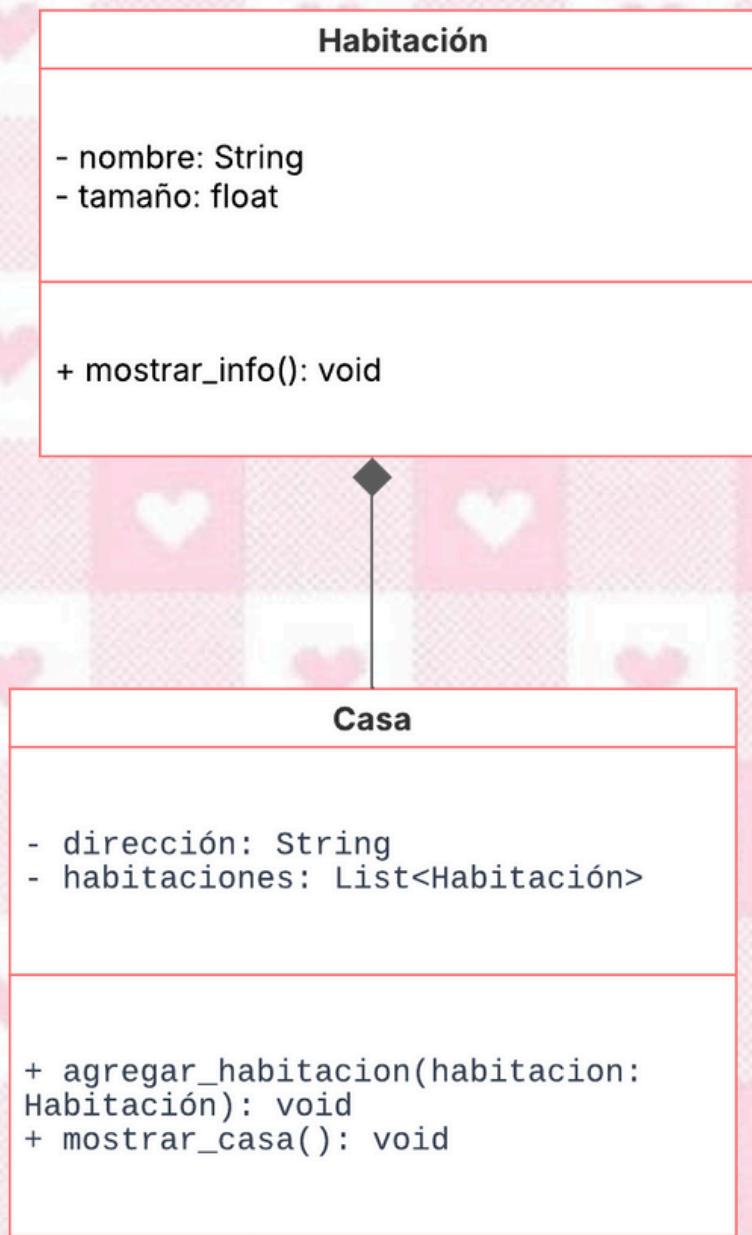
Métodos: mostrar\_info() (muestra el nombre y tamaño de la habitación)

Casa<dirección, habitaciones (lista de objetos de tipo Habitación)

Métodos: agregar\_habitacion(habitacion), mostrar\_casa() (muestra la dirección y la información de todas las habitaciones)

- Implementa las clases con sus constructores, getters y setters.
- Crea una casa y agrega varias habitaciones.
- Muestra la información de la casa y sus habitaciones.

## Diagrama de clases Uml



# Ejecucion

## Python

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux> & C:/Users/teran/OneDrive/Desktop/Progra II/Practica 2 aux/Composicion y Agregacion/ejemplo.py
Direccion: Avenida Principal 456
Habitacion: Dormitorio
Tamaño: 18 m^2
Habitacion: Baño
Tamaño: 8 m^2
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```

## Java

```
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Composicion y Agregacion\eje>
java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\teran\AppData\Roaming\redhat.java\jdt_ws\Java_80f93fce\bin' 'Main'
Casa en Calle Falsa 123:
Habitación: Sala, Tamaño: 20 m2
Habitación: Cocina, Tamaño: 10 m2
Habitación: Dormitorio, Tamaño: 15 m2
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Composicion y Agregacion\eje>
```

# Ejercicio 3.

3. Crea un POO de clases para modelar un avión y sus partes. El avión está compuesto por partes como el motor, las alas y el tren de aterrizaje. Si el avión se destruye, las partes también se destruyen.

**Parte**<nombre, peso (en kg)

Métodos: mostrar\_info() (muestra el nombre y el peso de la parte)

**Avión**<modelo, fabricante, partes (lista de objetos de tipo Parte)

Métodos: agregar\_parte(parte), mostrar\_avion() (muestra el modelo, fabricante y la información de todas las partes)

- Implementa las clases con sus constructores, getters y setters.
- Crea un avión y agrega varias partes.
- Muestra la información del avión y sus partes.

## Diagrama de clases Uml



# Ejecucion

## Python

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux> & C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Composicion y Agrupacion.py
Avion: Airbus A320 - Airbus
Parte: Turbina
Peso: 2200 kg
Parte: Tren de aterrizaje
Peso: 1300 kg
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```

## Java

```
PROBLEMS 5    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Composicion y Agrupacion> java.exe '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Composicion y Agrupacion.jar' 'Vehiculo'
Vehículo tipo: Avión, Marca: Boeing 747
Partes:
Parte: Motor, Peso: 2500.0 kg
Parte: Ala izquierda, Peso: 1500.0 kg
Parte: Ala derecha, Peso: 1500.0 kg
Parte: Cabina, Peso: 1200.0 kg
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Composicion y Agrupacion>
```

# Ejercicio 5.

5. Desarrolla un POO para un equipo de fútbol y sus jugadores. El equipo está compuesto por jugadores, y si el equipo se destruye, los jugadores también se destruyen. Además, los jugadores pueden ser de diferentes tipos (portero, defensa, mediocampista, delantero).

**Clase Padre:** Jugador<nombre, número, posición>

Métodos: mostrar\_info() (muestra el nombre, número y posición del jugador)

**Clases Derivadas:** Portero, Defensa, Mediocampista, Delantero (heredan de Jugador)

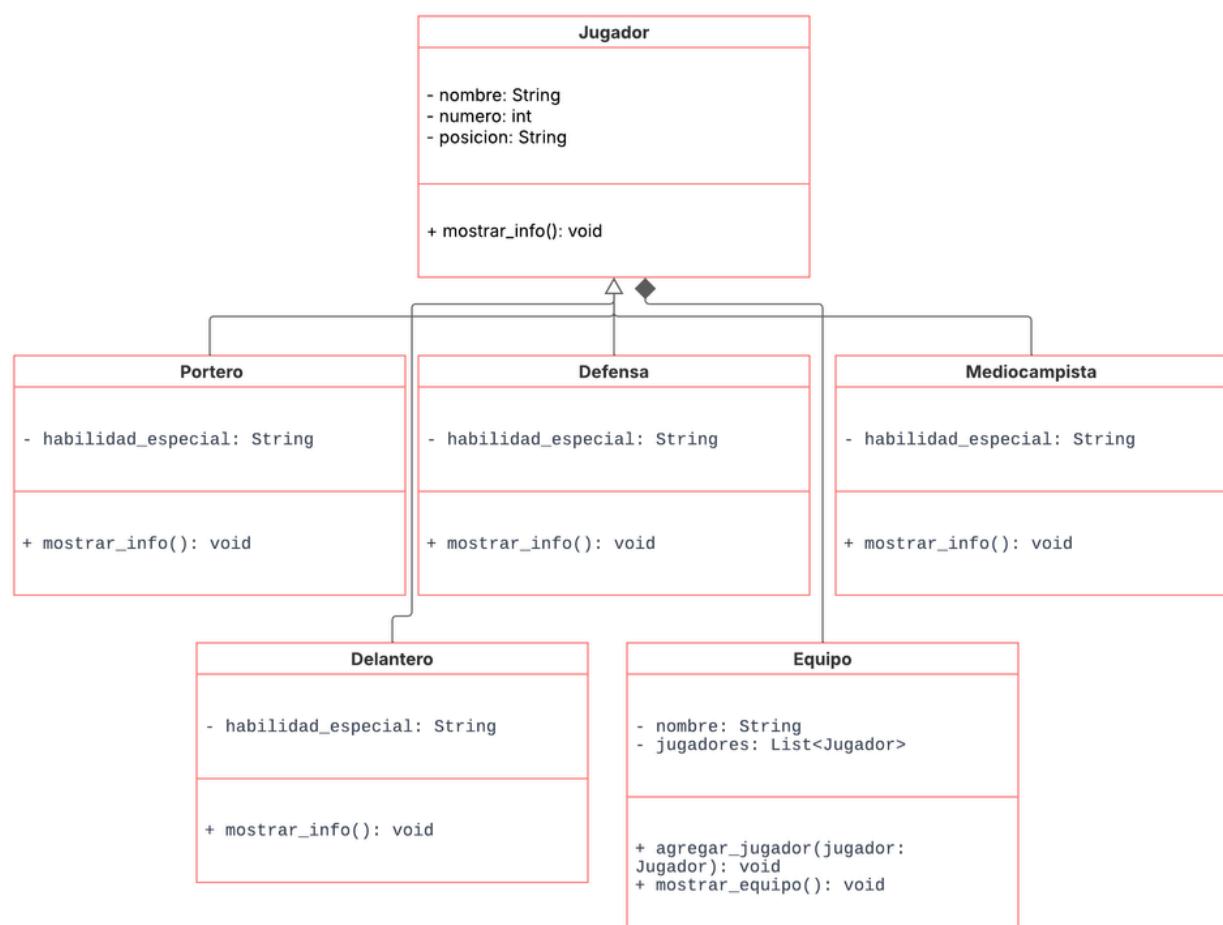
Atributos adicionales: habilidad\_especial (ej: "Atajadas", "Marcaje", "Pases", "Goles")

**Clase:** Equipo<nombre, jugadores (lista de objetos de tipo Jugador)>

Métodos: agregar\_jugador(jugador), mostrar\_equipo() (muestra el nombre del equipo y la información de todos los jugadores)

- Implementa las clases con sus constructores, getters y setters.
- Crea un equipo y agrega varios jugadores de diferentes tipos.
- Muestra la información del equipo y sus jugadores.

## Diagrama de clases Uml



# Ejecucion

Python

```
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux> & C:/Users/teran/OneDrive/Desktop/Progra II/Practica 2 aux/Composicion y Agregacion/ej
Equipo: Los shiposters
Jugador: Pedro
Número: 1
Posición: Portero
Habilidad especial: Atajadas
Jugador: Carlos
Número: 4
Posición: Defensa
Habilidad especial: Marcaje
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```

# Ejercicio 7.

7. Crea un POO para una universidad y sus estudiantes. La universidad contiene estudiantes, pero los estudiantes pueden existir independientemente de la universidad.

**Estudiante**< nombre, carrera, semestre>

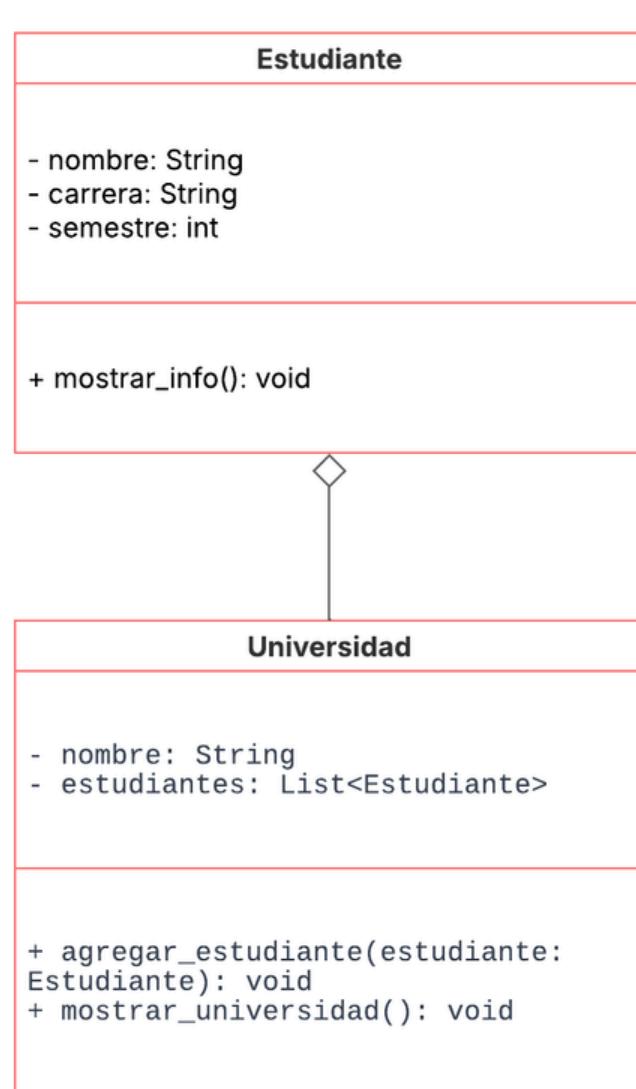
Métodos: mostrar\_info() (muestra el nombre, carrera y semestre del estudiante)

**Universidad**<nombre, estudiantes (lista de objetos de tipo Estudiante)>

Métodos: agregar\_estudiante(estudiante), mostrar\_universidad() (muestra el nombre de la universidad y la información de todos los estudiantes)

- Implementa las clases con sus constructores, getters y setters.
- Crea una universidad y agrega varios estudiantes.
- Muestra la información de la universidad y sus estudiantes.

## Diagrama de clases Uml



# Ejecucion

## Python

```
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux> & C:/Users/teran/OneDrive/Desktop/Progra II/Practica 2 aux/Composicion y Agregacion/EUniversidad: UPB
Nombre: Carlos
Carrera: Ingeniería
Semestre: 3
Nombre: María
Carrera: Arquitectura
Semestre: 1
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```



# Ejercicio 9.

9. Crea un POO para un carrito de compras y sus productos. El carrito contiene productos, pero los productos pueden existir independientemente del carrito. Además, el carrito no puede contener más de 10 productos.

**Producto**<nombre, precio>

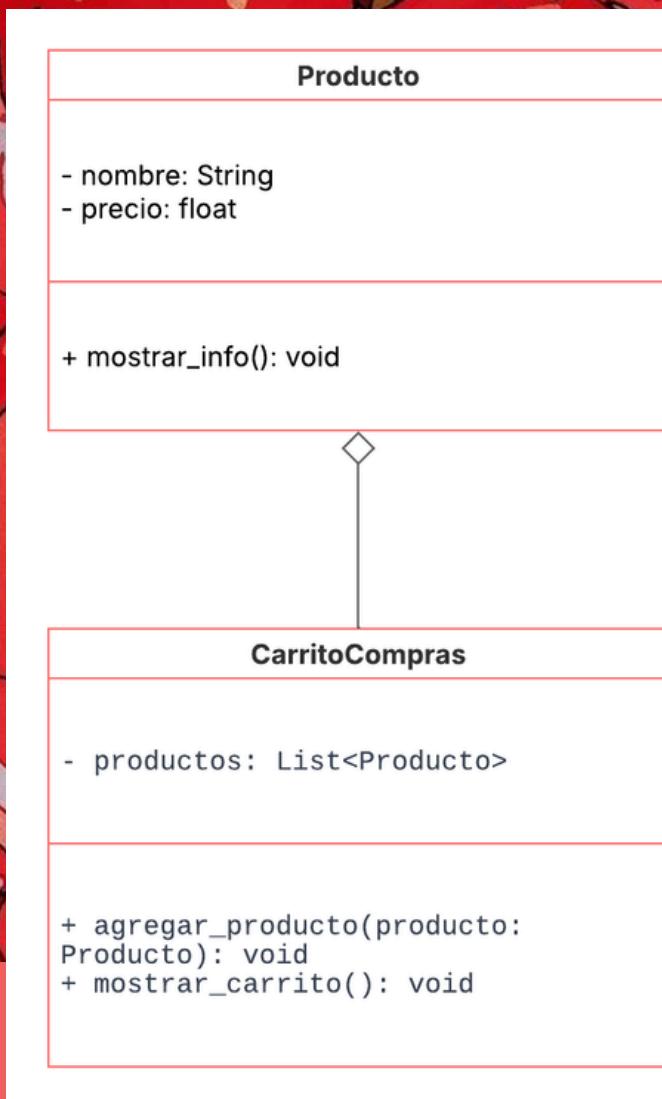
Métodos: mostrar\_info() (muestra el nombre y el precio del producto)

**CarritoCompras**<productos (lista de objetos de tipo Producto)>

Métodos: agregar\_producto(producto), mostrar\_carrito() (muestra la información de todos los productos en el carrito)

- Implementa las clases con sus constructores, getters y setters.
- Crea un carrito de compras y agrega varios productos, validando que no se exceda el límite de 10 productos.
- Muestra la información del carrito y sus productos.

## Diagrama de clases Uml



# Ejecucion

## Python



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux> & C:/Users/teran/OneDrive/Desktop/Progra II/Practica 2 aux/Composicion y Agregacion/E
No se pueden agregar más productos
No se pueden agregar más productos
Carrito de Compras:
Producto: Producto1
Precio: 0
Producto: Producto2
Precio: 10
Producto: Producto3
Precio: 20
Producto: Producto4
Precio: 30
Producto: Producto5
Precio: 40
Producto: Producto6
Precio: 50
Producto: Producto7
Precio: 60
Producto: Producto8
Precio: 70
Producto: Producto9
Precio: 80
Producto: Producto10
Precio: 90
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```

