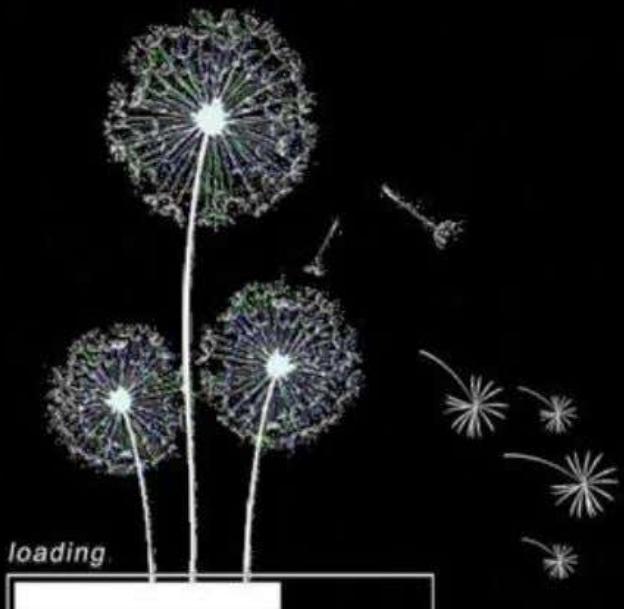




HERENCIA



loading

61%

Ejercicio 1.

1. **Modelar diferentes tipos de vehículos.** Las clases deben tener las siguientes características:

Vehículo<marca, modelo, año, precio_base>

Métodos: mostrar_info() muestra la información básica del vehículo

Coche (hereda de Vehículo)< num_puertas, tipo_combustible>

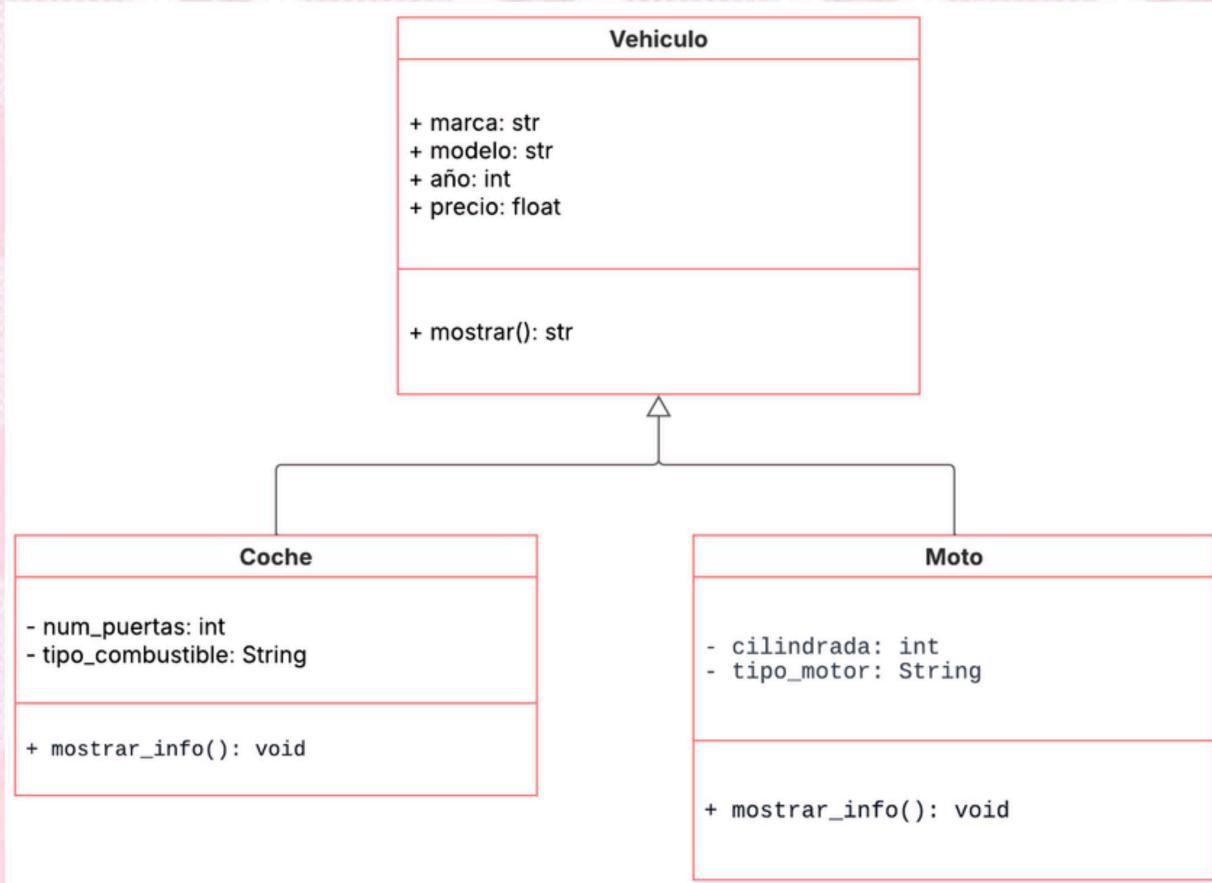
Métodos: mostrar_info() debe mostrar la información básica más los atributos adicionales

Moto (hereda de Vehículo)< cilindrada, tipo_motor>

Métodos: mostrar_info() debe mostrar la información básica más los atributos adicionales

- a) Implementa las clases con sus constructores, getters y setters.
- b) Crea instancias de Coche y Moto y muestra su información usando el método mostrar_info().
- c) Muestra todos los coches que tienen más de 4 puertas.
- d) Mostrar los coches y motos actuales (gestión 2025)

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/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/teran/OneDrive/Herencia/ejercicio 1/Python/main.py"
Vehiculos 2025
({'Marca': 'Honda', 'Modelo': 'Civic', 'Año': 2024, 'Precio Base': 22000}, {'Puertas': 4, 'Combustible': 'Gasolina'})
({'Marca': 'Chevrolet', 'Modelo': 'Tahoe', 'Año': 2024, 'Precio Base': 55000}, {'Puertas': 7, 'Combustible': 'Gasolina'})
({'Marca': 'Kawasaki', 'Modelo': 'Ninja ZX-10R', 'Año': 2024, 'Precio Base': 20000}, {'Cilindrada': 1000, 'cc', 'Tipo Motor': 'RGB'})
({'Marca': 'Suzuki', 'Modelo': 'V-Strom 650', 'Año': 2024, 'Precio Base': 9500}, {'Cilindrada': 650, 'cc', 'Tipo Motor': 'RGBD'})
tiene 4 puertas
({'Marca': 'Chevrolet', 'Modelo': 'Tahoe', 'Año': 2024, 'Precio Base': 55000}, {'Puertas': 7, 'Combustible': 'Gasolina'})
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```

Java

PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Modelo: Corolla
Año: 2025
Precio base: $22000.0
Número de puertas: 4
Tipo de combustible: Gasolina

Todas las motos:
Marca: Yamaha
Modelo: R1
Año: 2025
Precio base: $15000.0
Marca: Honda
Modelo: CBR
Año: 2025
Precio base: $14000.0
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Herencia\ejercicio 1\Java>
```

Ejercicio 3.

3. Definir las clases:

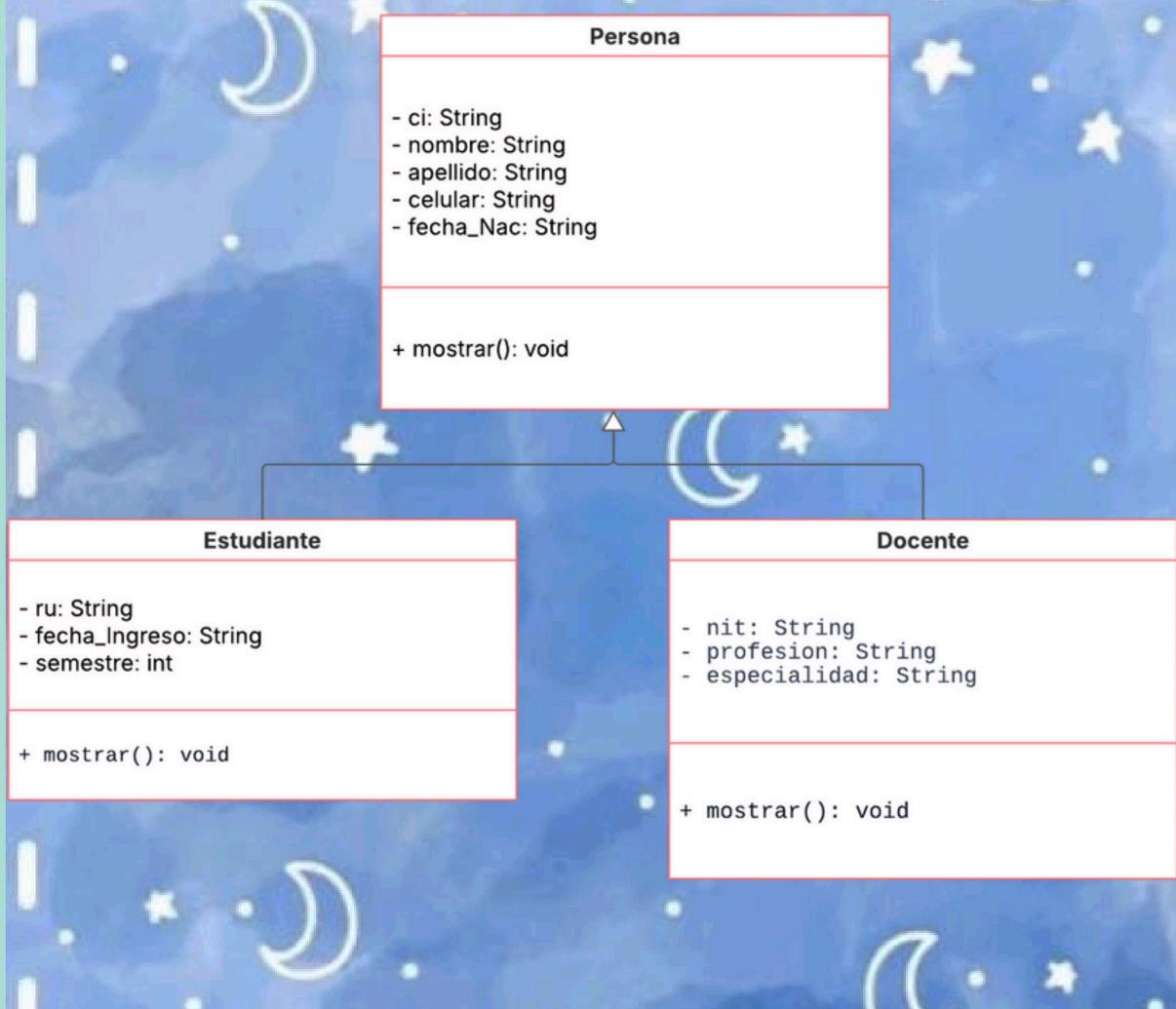
Persona <ci, nombre, apellido, celular, fecha_Nac>

Estudiante (heredado de persona) <ru, fecha_Ingreso, semestre>

Docente (heredado de persona) <nit, profesión, especialidad>

- a) Diseñar el diagrama UML de las clases anteriores.
- b) Implementa las clases con sus constructores, datos por defecto y mostrar.
- c) Mostrar los estudiantes mayores de 25 años.
- d) Mostrar al docente que tiene la profesión de "Ingeniero", es del sexo masculino y es el mayor de todos.
- e) Mostrar a los estudiantes y docentes que tienen el mismo apellido.

Diagrama de clases Uml



Ejecución

Python

```
aux\Herencia\ejercicio 3\Python\main.py
Estudiantes mayores de 25 años:
Mario López, CI: 130, Edad: 30
RU: RU010, Semestre: 10

Docente ingeniero, masculino y el mayor:

Personas con el mismo apellido:
Mario López, CI: 130, Edad: 30
RU: RU010, Semestre: 10
Javier López, CI: 210, Edad: 50
Profesión: Arquitecto, Especialidad: Diseño
Sofía Ramírez, CI: 131, Edad: 22
RU: RU011, Semestre: 2
Elena Ramírez, CI: 211, Edad: 34
Profesión: Bióloga, Especialidad: Genética
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```

Java

```
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Herencia\ejercicio 3\Java> & 'C:\Program Files\Java\jdk-17.0.2\bin\java' '-cp' 'C:\Users\teran\AppData\Roaming\Code\User\workspaces\Java\jdt_ws\Java_1bca8633\bin' 'Main'
Estudiantes mayores de 25:
Mario López, CI: 130, Edad: 30
RU: RU010, Semestre: 10

Docente Ingeniero, masculino, mayor:

Personas con el mismo apellido:
Mario López, CI: 130, Edad: 30
RU: RU010, Semestre: 10
Javier López, CI: 210, Edad: 50
Profesión: Arquitecto, Especialidad: Diseño
Sofía Ramírez, CI: 131, Edad: 22
RU: RU011, Semestre: 2
Elena Ramírez, CI: 211, Edad: 34
Profesión: Bióloga, Especialidad: Genética
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux\Herencia\ejercicio 3\Java>
```

Ejercicio 5.

5. Definir las siguientes clases:

Empleado<nombre, apellido, salario_base, años_antigüedad>

Métodos: calcular_salario() (retorna el salario base más un bono del 5% por cada año de antigüedad)

Gerente (hereda de Empleado)< departamento, bono_gerencial>

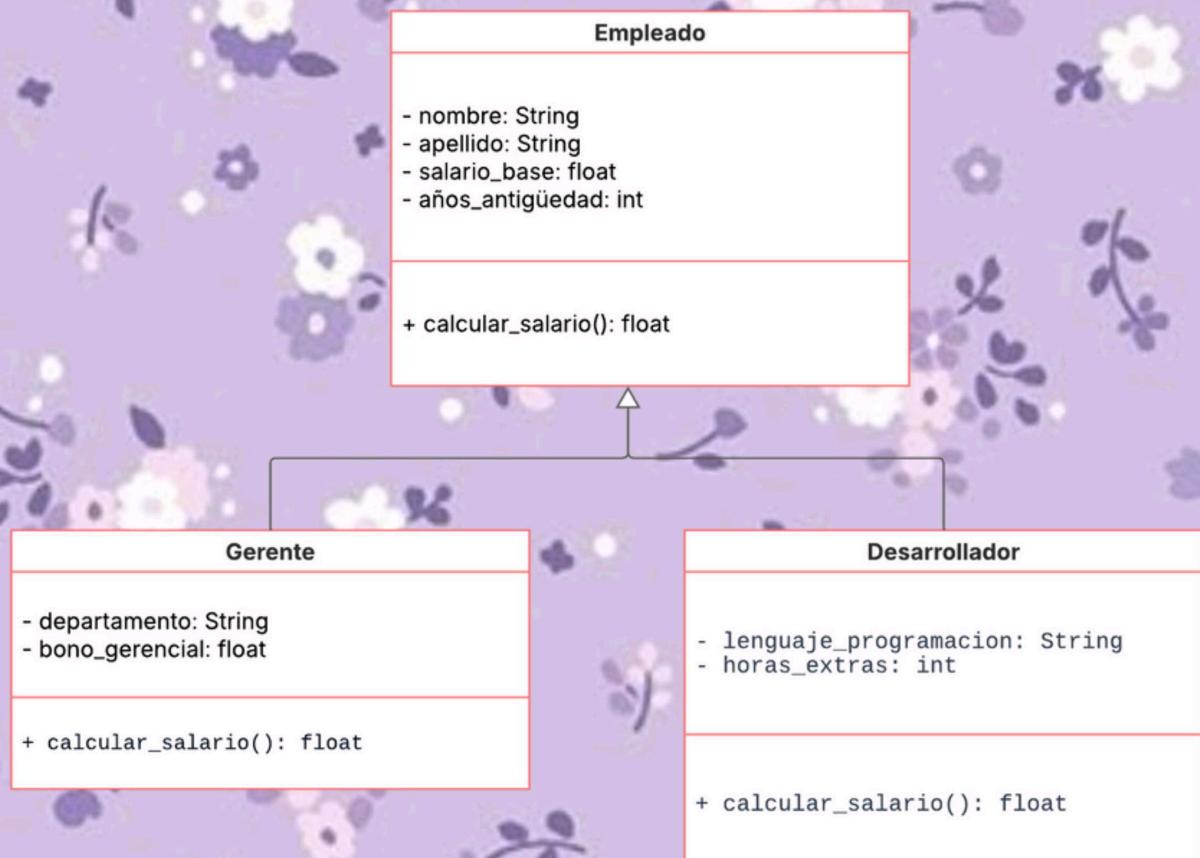
Métodos: calcular_salario() (debe sumar el bono gerencial al salario calculado en la clase base)

Desarrollador (hereda de Empleado) <lenguaje_programación, horas_extras>

Métodos: calcular_salario() (debe sumar un monto adicional por horas extras al salario calculado en la clase base)

- a) Implementa las clases con sus constructores, getters y setters.
- b) Crea instancias de Gerente y Desarrollador y muestra su salario calculado.
- c) Muestra todos los gerentes que tienen un bono gerencial mayor a 1000.
- d) Muestra todos los desarrolladores que trabajan más de 10 horas extras.

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/t
aux/Herencia/ejercicio 5/Python/Main.py"
Gerente: María - Salario: 4750.0
Bono gerencial mayor a 1000: 1500
Gerente: Jorge - Salario: 3620.0
Bono gerencial mayor a 1000: 1100
Desarrollador: Elena - Salario: 2485.0
Trabaja más de 10 horas extras: 15
Desarrollador: Miguel - Salario: 2367.5
PS C:\Users\teran\OneDrive\Desktop\Progra II\Practica 2 aux>
```