En este caso práctico, vamos a modelar un sistema de gestión de empleados para una empresa utilizando la programación orientada a objetos en Python.

1. Creación de la clase base `Empleado`

```python
class Empleado:
    def __init__(self, nombre, identificacion, salario):
        self.nombre = nombre
        self.identificacion = identificacion
        self.salario = salario

    def mostrar_informacion(self):
        print(f"Empleado: {self.nombre}\nID: {self.identificacion}\nSalario: {self.salario}")
```

2. Creación de clases derivadas

Clase `Gerente`

```python
class Gerente(Empleado):
    def __init__(self, nombre, identificacion, salario, departamento):
        super().__init__(nombre, identificacion, salario)
        self.departamento = departamento

    def mostrar_informacion(self):
        super().mostrar_informacion()
        print(f"Departamento: {self.departamento}")
```

Clase `Vendedor`

```python
class Vendedor(Empleado):
    def __init__(self, nombre, identificacion, salario, ventas):
        super().__init__(nombre, identificacion, salario)
        self.ventas = ventas

    def mostrar_informacion(self):
        super().mostrar_informacion()
        print(f"Ventas: {self.ventas}")

    def calcular_comision(self, porcentaje_comision):
        comision = self.ventas * porcentaje_comision
        print(f"Comisión: {comision}")
```

3. Creación de objetos y demostración de polimorfismo

```python
gerente = Gerente("Laura", "G123", 5000, "Marketing")
vendedor = Vendedor("Carlos", "V456", 3000, 15000)

print("Información del gerente:")
gerente.mostrar_informacion()
print("\nInformación del vendedor:")
vendedor.mostrar_informacion()
print("\nCálculo de comisión del vendedor:")
vendedor.calcular_comision(0.10)
```

Salida:

```
Información del gerente:
Empleado: Laura
ID: G123
Salario: 5000
Departamento: Marketing

Información del vendedor:
Empleado: Carlos
ID: V456
Salario: 3000
Ventas: 15000

Cálculo de comisión del vendedor:
Comisión: 1500.0
```

En este ejemplo, hemos creado una clase base `Empleado` y dos clases derivadas, `Gerente` y `Vendedor`. La clase `Gerente` hereda todos los métodos y atributos de la clase `Empleado` y agrega un atributo adicional (`departamento`). La clase `Vendedor` hereda también todos los métodos y atributos de la clase `Empleado`, pero agrega un atributo adicional (`ventas`) y un nuevo método (`calcular_comision`).

Este ejemplo demuestra cómo la programación orientada a objetos en Python puede ser utilizada para modelar un sistema de gestión de empleados en un negocio. La herencia y el polimorfismo permiten reutilizar y extender código de una clase base en clases derivadas, lo que facilita la creación de un sistema modular y fácil de mantener.