UNIVERSIDAD NACIONAL DE COLOMBIA

Programación orientada a objetos

Actividad 3

Estudiante:

Maria Fernanda Valencia Jimenez

Profesor:

Walter Hugo Arboleda Mazo

Fecha:

20 de mayo del 2025

Tabla de contenido

1. Ejercicio 4.1 página 194

Código fuente

```
class Cuenta:

    def __init__(self, saldo, tasa_anual):

        self.saldo = saldo

        self.tasa_anual = tasa_anual

        self.numero_consignaciones = 0

        self.numero_retiros = 0

        self.comision_mensual = 0
```

```
    def consignar(self, cantidad):
        self.saldo += cantidad
        self.numero_consignaciones += 1
    def retirar(self, cantidad):
        nuevo_saldo = self.saldo - cantidad
        if nuevo_saldo >= 0:
            self.saldo -= cantidad
            self.numero_retiros += 1
        else:
            print("La cantidad a retirar excede el saldo actual.")
    def calcular_interes(self):
        tasa_mensual = self.tasa_anual / 12
        interes_mensual = self.saldo * tasa_mensual
        self.saldo += interes_mensual
    def extracto_mensual(self):
        self.saldo -= self.comision_mensual
        self.calcular_interes()
class CuentaAhorros(Cuenta):
    def __init__(self, saldo, tasa):
        super().__init__(saldo, tasa)
        self.activa = saldo >= 10000
    def retirar(self, cantidad):
        if self.activa:
            super().retirar(cantidad)
    def consignar(self, cantidad):
        if self.activa:
            super().consignar(cantidad)
    def extracto_mensual(self):
```

```
        if self.numero_retiros > 4:
            self.comision_mensual += (self.numero_retiros - 4) * 1000
        super().extracto_mensual()
        self.activa = self.saldo >= 10000
    def imprimir(self):
        print(f"Saldo = $ {self.saldo}")
        print(f"Comisión mensual = $ {self.comision_mensual}")
        print(f"Número de transacciones = {self.numero_consignaciones + self.numero_retiros}")
        print()
class CuentaCorriente(Cuenta):
    def __init__(self, saldo, tasa):
        super().__init__(saldo, tasa)
        self.sobregiro = 0
    def retirar(self, cantidad):
        resultado = self.saldo - cantidad
        if resultado < 0:
            self.sobregiro -= resultado
            self.saldo = 0
        else:
            super().retirar(cantidad)
     def consignar(self, cantidad):
        residuo = self.sobregiro - cantidad
        if self.sobregiro > 0:
            if residuo > 0:
                self.sobregiro = 0
                self.saldo = residuo
            else:
                self.sobregiro = -residuo
```

```
            self.saldo = 0
        else:
            super().consignar(cantidad)
    def extracto_mensual(self):
        super().extracto_mensual()
    def imprimir(self):
        print(f"Saldo = $ {self.saldo}")
        print(f"Cargo mensual = $ {self.comision_mensual}")
        print(f"Número de transacciones = {self.numero_consignaciones + self.numero_retiros}")
        print(f"Valor de sobregiro = $ {self.sobregiro}")
        print()
def probar_cuentas():
    print("Cuenta de ahorros")
    try:
        saldo_inicial_ahorros = float(input("Ingrese saldo inicial= $"))
        tasa_ahorros = float(input("Ingrese tasa de interés= "))
        cuenta1 = CuentaAhorros(saldo_inicial_ahorros, tasa_ahorros)
        cantidad_depositar = float(input("Ingresar cantidad a consignar: $"))
        cuenta1.consignar(cantidad_depositar)
        cantidad_retirar = float(input("Ingresar cantidad a retirar: $"))
        cuenta1.retirar(cantidad_retirar)
        cuenta1.extracto_mensual()
        cuenta1.imprimir()
    except ValueError:
        print("Error: Por favor, ingrese valores numéricos válidos.")
if __name__ == "__main__":
    probar_cuentas()
```

Ejecución:

Cuenta de ahorros

Ingrese saldo inicial= $100000

Ingrese tasa de interés= 0.10

Ingresar cantidad a consignar: $50000

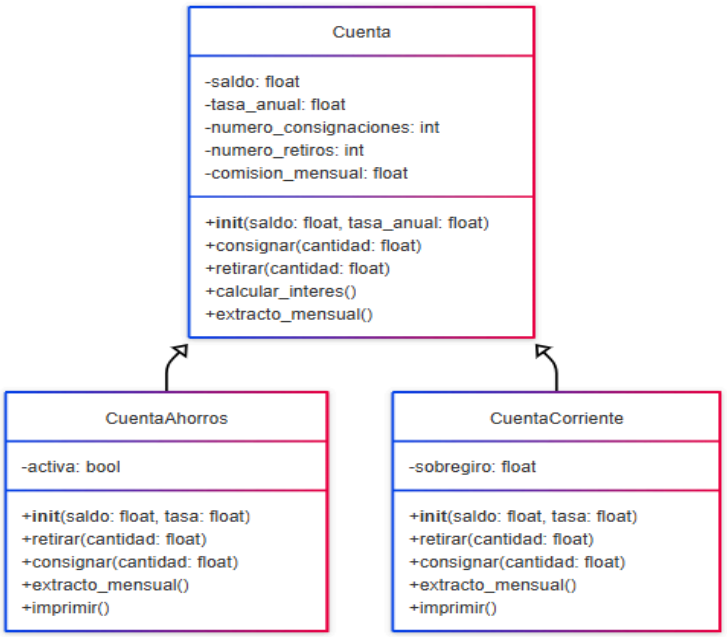
Ingresar cantidad a retirar: $70000

Saldo = $ 80666.66666666667

Comisión mensual = $ 0

Número de transacciones = 2

Diagrama de clases



2. Ejercicio 4.2 página 206

Código fuente

```
from enum import Enum

class Inmueble:

    def __init__(self, identificador_inmobiliario, area, direccion):
```

```
        self.identificador_inmobiliario = identificador_inmobiliario
        self.area = area
        self.direccion = direccion
        self.precio_venta = 0.0
    def calcular_precio_venta(self, valor_area):
        self.precio_venta = self.area * valor_area
        return self.precio_venta
    def imprimir(self):
        print(f"Identificador inmobiliario = {self.identificador_inmobiliario}")
        print(f"Area = {self.area}")
        print(f"Dirección = {self.direccion}")
        print(f"Precio de venta = ${self.precio_venta:,.2f}")
class InmuebleVivienda(Inmueble):
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos):
        super().__init__(identificador_inmobiliario, area, direccion)
        self.numero_habitaciones = numero_habitaciones
        self.numero_banos = numero_banos
    def imprimir(self):
        super().imprimir()
        print(f"Número de habitaciones = {self.numero_habitaciones}")
        print(f"Número de baños = {self.numero_banos}")
class Casa(InmuebleVivienda):
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_banos)
        self.numero_pisos = numero_pisos
```

```
    def imprimir(self):
        super().imprimir()
        print(f"Número de pisos = {self.numero_pisos}")
class Apartamento(InmuebleVivienda):
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_banos)
    def imprimir(self):
        super().imprimir()
class CasaRural(Casa):
    valor_area = 1500000
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos, numero_pisos,
                 distancia_cabecera, altitud):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_banos, numero_pisos)
        self.distancia_cabecera = distancia_cabecera
        self.altitud = altitud
    def imprimir(self):
        super().imprimir()
        print(f"Distancia a la cabecera municipal = {self.distancia_cabecera} km.")
        print(f"Altitud sobre el nivel del mar = {self.altitud} metros.")
        print()
class ApartamentoFamiliar(Apartamento):
    valor_area = 2000000
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos, valor_administracion):
```

```
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_banos)
        self.valor_administracion = valor_administracion
    def imprimir(self):
        super().imprimir()
        print(f"Valor de la administración = ${self.valor_administracion:,.2f}")
        print()
class Apartaestudio(Apartamento):
    valor_area = 1500000
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos):
        super().__init__(identificador_inmobiliario, area, direccion, 1, 1)
    def imprimir(self):
        super().imprimir()
        print()
class CasaUrbana(Casa):
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_banos, numero_pisos)
    def imprimir(self):
        super().imprimir()
class CasaConjuntoCerrado(CasaUrbana):
    valor_area = 2500000
    def __init__(self, identificador_inmobiliario, area, direccion,
                 numero_habitaciones, numero_banos, numero_pisos,
                 valor_administracion, tiene_piscina, tiene_campos_deportivos):
        super().__init__(identificador_inmobiliario, area, direccion,
```

```
                    numero_habitaciones, numero_banos, numero_pisos)
        self.valor_administracion = valor_administracion
        self.tiene_piscina = tiene_piscina
        self.tiene_campos_deportivos = tiene_campos_deportivos
    def imprimir(self):
        super().imprimir()
        print(f"Valor de la administración = {self.valor_administracion}")
        print(f"Tiene piscina? = {self.tiene_piscina}")
        print(f"Tiene campos deportivos? = {self.tiene_campos_deportivos}")
        print()
class CasaIndependiente(CasaUrbana):
    valor_area = 3000000
    def __init__(self, identificador_inmobiliario, area, direccion,
                numero_habitaciones, numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion,
                    numero_habitaciones, numero_banos, numero_pisos)
    def imprimir(self):
        super().imprimir()
        print()
class Local(Inmueble):
    class TipoLocal(Enum):
        INTERNO = 1
        CALLE = 2
    def __init__(self, identificador_inmobiliario, area, direccion, tipo_local):
        super().__init__(identificador_inmobiliario, area, direccion)
        self.tipo_local = tipo_local
    def imprimir(self):
        super().imprimir()
```

```
        print(f"Tipo de local = {self.tipo_local.name}")
class LocalComercial(Local):
    valor_area = 3000000
    def __init__(self, identificador_inmobiliario, area, direccion,
                 tipo_local, centro_comercial):
        super().__init__(identificador_inmobiliario, area, direccion, tipo_local)
        self.centro_comercial = centro_comercial
    def imprimir(self):
        super().imprimir()
        print(f"Centro comercial = {self.centro_comercial}")
        print()
class Oficina(Local):
    valor_area = 3500000
    def __init__(self, identificador_inmobiliario, area, direccion,
                 tipo_local, es_gobierno):
        super().__init__(identificador_inmobiliario, area, direccion, tipo_local)
        self.es_gobierno = es_gobierno
    def imprimir(self):
        super().imprimir()
        print(f"Es oficina gubernamental = {self.es_gobierno}")
        print()
if __name__ == "__main__":
    apto1 = ApartamentoFamiliar(103067, 120, "Avenida Santander 45-45", 3, 2, 200000)
    print("Datos apartamento")
    apto1.calcular_precio_venta(apto1.valor_area)
    apto1.imprimir()
    print("Datos apartaestudio")
    aptestudio1 = Apartaestudio(12354, 50, "Avenida Caracas 30-15", 1, 1)
```

```
aptestudio1.calcular_precio_venta(aptestudio1.valor_area)
aptestudio1.imprimir()
```

Ejecución:

Datos apartamento

Identificador inmobiliario = 103067

Area = 120

Dirección = Avenida Santander 45-45

Precio de venta = $240,000,000.00

Número de habitaciones = 3

Número de baños = 2

Valor de la administración = $200,000.00

Datos apartaestudio

Identificador inmobiliario = 12354

Area = 50

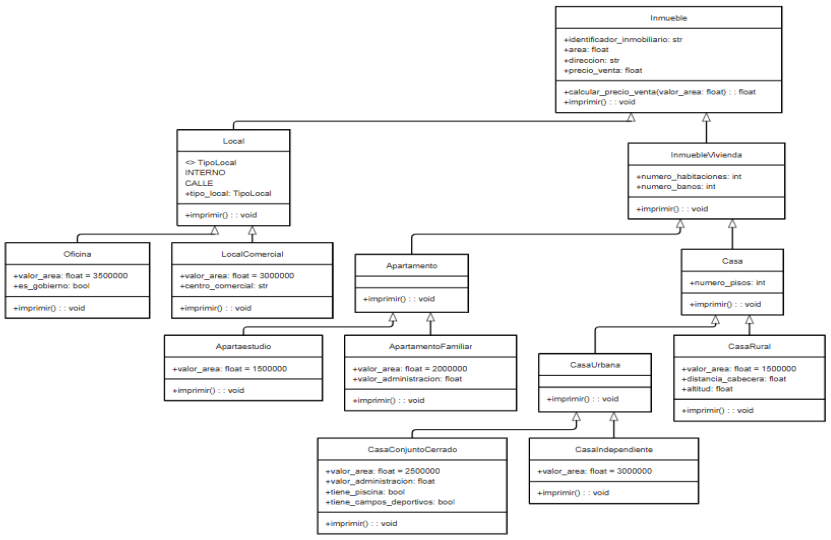
Dirección = Avenida Caracas 30-15

Precio de venta = $75,000,000.00

Número de habitaciones = 1

Número de baños = 1

Diagrama de clases



3. Ejercicio propuesto página 227

Código fuente

```
 class Mascota:
    """Clase base para todos los animales de la tienda de mascotas."""
    def __init__(self, nombre: str, edad: int, color: str):
        self.nombre = nombre
        self.edad = edad
        self.color = color
     def __str__(self):
        """Representación en cadena de la Mascota."""
        return f"Nombre: {self.nombre}, Edad: {self.edad} años, Color: {self.color}"
# --- TiendaMascotas/perros.py ---
class Perro(Mascota):
    """Clase base para todos los perros."""
     def __init__(self, nombre: str, edad: int, color: str, peso: float, muerde: bool):
        super().__init__(nombre, edad, color)
        self.peso = peso
        self.muerde = muerde
    @staticmethod
    def sonido():
        """Método estático que imprime el sonido de los perros."""
        print("Los perros ladran")
    def __str__(self):
        """Representación en cadena del Perro."""
        return (f"{super().__str__()}, Peso: {self.peso} kg, "
                f"¿Muerde?: {'Sí' if self.muerde else 'No'}")
```

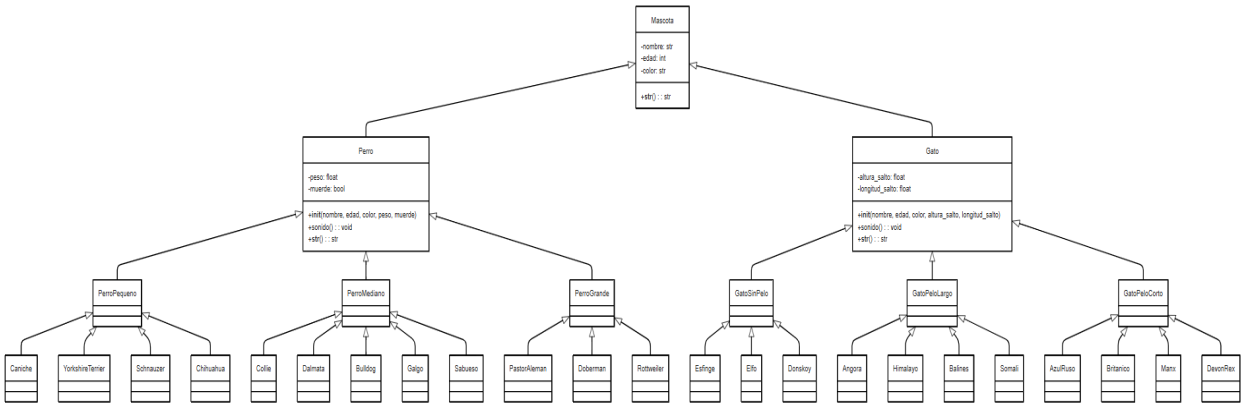
```
# --- Categorías de Perros ---
class PerroGrande(Perro):
    """Categoría para perros grandes."""
    pass
class PerroMediano(Perro):
    """Categoría para perros medianos."""
    pass
class PerroPequeño(Perro):
    """Categoría para perros pequeños."""
    pass
# --- Razas de Perros Pequeños ---
class Caniche(PerroPequeño):
    pass
class YorkshireTerrier(PerroPequeño):
    pass
class Schnauzer(PerroPequeño):
    pass
class Chihuahua(PerroPequeño):
    pass
# --- Razas de Perros Medianos ---
class Collie(PerroMediano):
    pass
class Dalmata(PerroMediano):
    pass
class Bulldog(PerroMediano):
    pass
class Galgo(PerroMediano):
    pass
```

```
class Sabueso(PerroMediano):
    pass
# --- Razas de Perros Grandes ---
class PastorAleman(PerroGrande):
    pass
class Doberman(PerroGrande):
    pass
class Rottweiler(PerroGrande):
    pass
# --- TiendaMascotas/gatos.py ---
class Gato(Mascota):
    """Clase base para todos los gatos."""
    def __init__(self, nombre: str, edad: int, color: str, altura_salto: float, longitud_salto: float):
        super().__init__(nombre, edad, color)
        self.altura_salto = altura_salto
        self.longitud_salto = longitud_salto
    @staticmethod
    def sonido():
        """Método estático que imprime el sonido de los gatos."""
        print("Los gatos maúllan y ronronean")
    def __str__(self):
        """Representación en cadena del Gato."""
        return (f"{super().__str__()}, Altura de salto: {self.altura_salto} metros, "
                f"Longitud de salto: {self.longitud_salto} metros")
# --- Categorías de Gatos ---
class GatoSinPelo(Gato):
    """Categoría para gatos sin pelo."""
    pass
```

```
class GatoPeloLargo(Gato):
    """Categoría para gatos de pelo largo."""
    pass
class GatoPeloCorto(Gato):
    """Categoría para gatos de pelo corto."""
    pass
# --- Razas de Gatos Sin Pelo ---
class Esfinge(GatoSinPelo):
    pass
class Elfo(GatoSinPelo):
    pass
class Donskoy(GatoSinPelo):
    pass
# --- Razas de Gatos de Pelo Largo ---
class Angora(GatoPeloLargo):
    pass
class Himalayo(GatoPeloLargo):
    pass
class Balines(GatoPeloLargo):
    pass
class Somali(GatoPeloLargo):
    pass
# --- Razas de Gatos de Pelo Corto ---
class AzulRuso(GatoPeloCorto):
    pass
class Britanico(GatoPeloCorto):
    pass
```

```
class Manx(GatoPeloCorto):
    pass
class DevonRex(GatoPeloCorto):
    pass.
```

Diagrama de clases



4. Ejercicio propuesto página 231

Código fuente

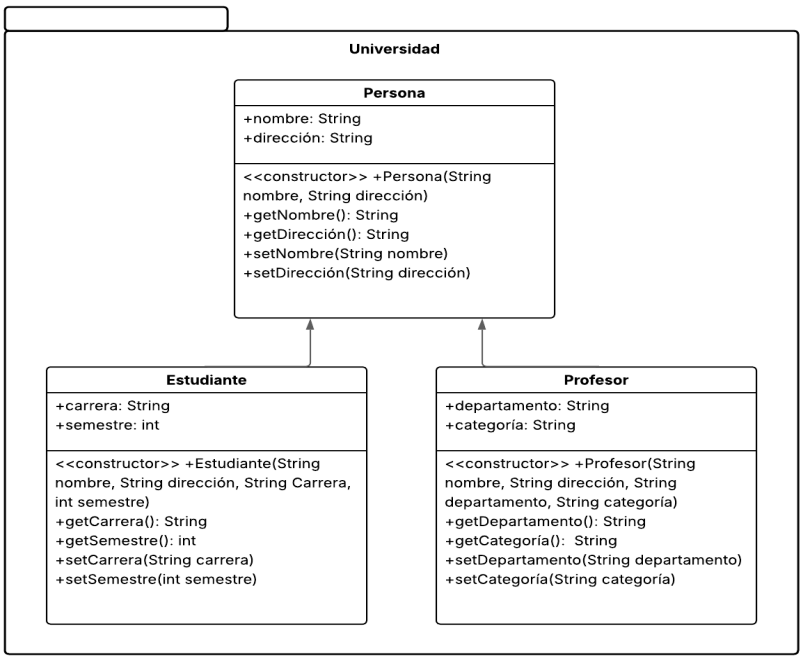
```
class Persona:
    def __init__(self, nombre: str, direccion: str):
        self._nombre = nombre
        self._direccion = direccion
    def getNombre(self):
        return self._nombre
    def getDireccion(self):
        return self._direccion
    def setNombre(self, nombre: str):
        self._nombre = nombre
```

```
    def setDireccion(self, direccion: str):
        self._direccion = direccion
class Estudiante(Persona):
    def __init__(self, nombre: str, direccion: str, carrera: str, semestre: int):
        super().__init__(nombre, direccion)
        self._carrera = carrera
        self._semestre = semestre
    def getCarrera(self) -> str:
        return self._carrera
    def getSemestre(self) -> int:
        return self._semestre
    def setCarrera(self, carrera: str):
        self._carrera = carrera
    def setSemestre(self, semestre: int):
        self._semestre = semestre
class Profesor(Persona):
    def __init__(self, nombre: str, direccion: str, departamento: str, categoria: str):
        super().__init__(nombre, direccion)
        self._departamento = departamento
        self._categoria = categoria
    def getDepartamento(self):
        return self._departamento
    def getCategoria(self):
        return self._categoria
    def setDepartamento(self, departamento: str):
        self._departamento = departamento
    def setCategoria(self, categoria: str):
        self._categoria = categoria
```

Ejecución:

```
if __name__ == "__main__":
    estudiante1 = Estudiante("Maria Fernanda", "Calle 123", "Ingeniería administrativa", 1)
    Estudiante.setCarrera(estudiante1, "Estadística")
    Estudiante.setSemestre(estudiante1, 7)
    print(f"Nombre del estudiante: {estudiante1.getNombre()}")
    print(f"Carrera del estudiante: {estudiante1.getCarrera()}")
    profesor1 = Profesor("Manuel Rojas", "Avenida 456", "Zootécnia", "Ocasional")
    Profesor.setNombre(profesor1, "Raúl Velez")
    Profesor.setDepartamento(profesor1, "Matemáticas")
    Profesor.setCategoria(profesor1, "Titular")
    print(f"Nombre del profesor: {profesor1.getNombre()}")
    print(f"Departamento del profesor: {profesor1.getDepartamento()}")
```

Diagrama de clases



5. Ejercicio 4.4 página 231

Código fuente

```
class Profesor:
    def imprimir(self):
        print("Es un profesor.")
class ProfesorTitular(Profesor): # Indicamos la herencia de la clase Profesor
    def imprimir(self):
        print("Es un profesor titular.")
class Prueba:
    @staticmethod
    def main():
        profesor1 = ProfesorTitular()
        profesor1.imprimir()
if __name__ == "__main__":
    Prueba.main()
```

Diagrama de clases