UNIVERSIDAD NACIONAL DE COLOMBIA

Programación orientada a objetos

Actividad 2

Integrantes:

Juan Felipe Muriel Mosquera

Profesor:

Walter Hugo Arboleda Mazo

Fecha:

20 de mayo del 2025

Punto 1. Página 194

Código fuente:

```
class CuentaBancaria:

  def __init__(self, balance, interes_anual):
    self.balance = balance
    self.interes_anual = interes_anual
    self.depositos = 0
    self.retiros = 0
    self.cargo_mensual = 0

  def agregar_fondos(self, valor):
    self.balance += valor
    self.depositos += 1

  def extraer_fondos(self, valor):
    if self.balance >= valor:
      self.balance -= valor
      self.retiros += 1
    else:
      print("No hay fondos suficientes para realizar el retiro.")

  def aplicar_interes(self):
    interes_mensual = self.interes_anual / 12
    ganancia = self.balance * interes_mensual
    self.balance += ganancia

  def resumen_mensual(self):
    self.balance -= self.cargo_mensual
```

```
        self.aplicar_interes()


class CajaAhorro(CuentaBancaria):

    def __init__(self, balance, interes):
        super().__init__(balance, interes)
        self.estado = balance >= 10000

    def extraer_fondos(self, valor):
        if self.estado:
            super().extraer_fondos(valor)

    def agregar_fondos(self, valor):
        if self.estado:
            super().agregar_fondos(valor)

    def resumen_mensual(self):
        if self.retiros > 4:
            self.cargo_mensual += (self.retiros - 4) * 1000
        super().resumen_mensual()
        self.estado = self.balance >= 10000

    def mostrar_info(self):
        print(f"Saldo actual: $ {self.balance}")
        print(f"Cargo del mes: $ {self.cargo_mensual}")
        print(f"Total de movimientos: {self.depositos + self.retiros}")
        print()


class CuentaCheque(CuentaBancaria):
```

```
def __init__(self, balance, interes):
    super().__init__(balance, interes)
    self.descubierto = 0

def extraer_fondos(self, valor):
    resultado = self.balance - valor
    if resultado < 0:
        self.descubierto -= resultado
        self.balance = 0
    else:
        super().extraer_fondos(valor)

def agregar_fondos(self, valor):
    if self.descubierto > 0:
        restante = self.descubierto - valor
        if restante > 0:
            self.descubierto = restante
            self.balance = 0
        else:
            self.descubierto = 0
            self.balance = -restante
    else:
        super().agregar_fondos(valor)

def resumen_mensual(self):
    super().resumen_mensual()

def mostrar_info(self):
    print(f"Saldo actual: $ {self.balance}")
    print(f"Cargo del mes: $ {self.cargo_mensual}")
```

```
        print(f"Total de movimientos: {self.depositos + self.retiros}")
        print(f"Descubierto actual: $ {self.descubierto}")
        print()


def ejecutar_simulacion():
    print("Simulación de Caja de Ahorro")
    try:
        monto_inicial = float(input("Ingrese el saldo inicial: $"))
        tasa = float(input("Ingrese la tasa de interés anual: "))
        ahorro = CajaAhorro(monto_inicial, tasa)

        deposito = float(input("¿Cuánto desea consignar?: $"))
        ahorro.agregar_fondos(deposito)

        retiro = float(input("¿Cuánto desea retirar?: $"))
        ahorro.extraer_fondos(retiro)

        ahorro.resumen_mensual()
        ahorro.mostrar_info()

    except ValueError:
        print("Error: Ingrese solo números válidos.")

if __name__ == "__main__":
    ejecutar_simulacion()
```

Ejecución:

```
Simulación de Caja de Ahorro
Ingrese el saldo inicial: $100000
```

Ingrese la tasa de interés anual: 0.10

¿Cuánto desea consignar?: $50000

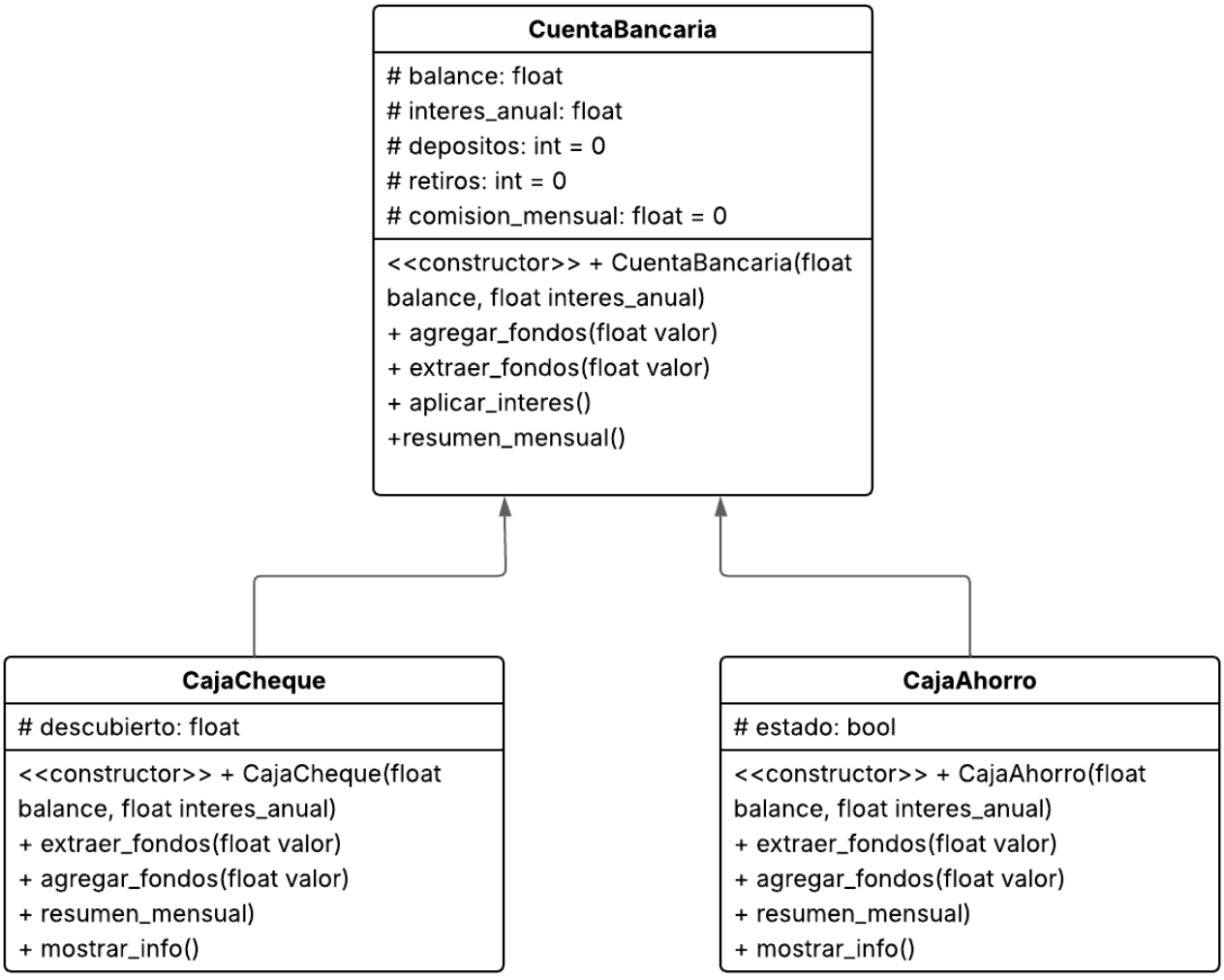
¿Cuánto desea retirar?: $70000

Saldo actual: $ 80666.66666666667

Cargo del mes: $ 0

Total de movimientos: 2

Diagrama de clases:



Punto 2. Página 206

Código fuente:

```
from enum import Enum


class Inmueble:
    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str):
        self._identificador_inmobiliario = identificador_inmobiliario
        self._area = area
        self._direccion = direccion
        self._precio_venta = 0.0

    def calcular_precio_venta(self, valor_area: float):
        self._precio_venta = self._area * valor_area
        return self._precio_venta

    def imprimir(self):
        print(f"Identificador inmobiliario = {self._identificador_inmobiliario}")
        print(f"Área = {self._area}")
        print(f"Dirección = {self._direccion}")
        print(f"Precio de venta = ${self._precio_venta:,.2f}")



class InmuebleVivienda(Inmueble):
    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str, numero_habitaciones:
int, numero_baños: int):
        super().__init__(identificador_inmobiliario, area, direccion)
        self._numero_habitaciones = numero_habitaciones
        self._numero_baños = numero_baños
```

```
    def imprimir(self):
        super().imprimir()
        print(f"Número de habitaciones = {self._numero_habitaciones}")
        print(f"Número de baños = {self._numero_baños}")


class Casa(InmuebleVivienda):
    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str, numero_habitaciones:
int, numero_baños: int, numero_pisos: int):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_baños)
        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: int, area: int, direccion: str, numero_habitaciones:
int, numero_baños: int):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_baños)


    def imprimir(self):
        super().imprimir()


class CasaRural(Casa):
    valor_area = 1500000 # Atributo de clase (equivalente a static en Java)


    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str, numero_habitaciones:
int, numero_baños: int, numero_pisos: int, distancia_cabecera: int, altitud: int):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,
numero_baños, 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 CasaUrbana(Casa):
    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str, numero_habitaciones: int, numero_baños: int, numero_pisos: int):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones, numero_baños, numero_pisos)

    def imprimir(self):
        super().imprimir()

class ApartamentoFamiliar(Apartamento):
    valor_area = 2000000 # Atributo de clase

    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str, numero_habitaciones: int, numero_baños: int, valor_administracion: int):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones, numero_baños)
        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 # Atributo de clase

  def __init__(self, identificador_inmobiliario: int, area: int, direccion: str):
    # Los apartaestudios tienen una sola habitación y un solo baño por defecto
    super().__init__(identificador_inmobiliario, area, direccion, 1, 1)

  def imprimir(self):
    super().imprimir()
    print()

class CasaConjuntoCerrado(CasaUrbana):
  valor_area = 2500000 # Atributo de clase

  def __init__(self, identificador_inmobiliario: int, area: int, direccion: str,
        numero_habitaciones: int, numero_baños: int, numero_pisos: int,
        valor_administracion: int, tiene_piscina: bool, tiene_campos_deportivos: bool):
    super().__init__(identificador_inmobiliario, area, direccion,
          numero_habitaciones, numero_baños, 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:,.2f}")
    print(f"¿Tiene piscina? = {self._tiene_piscina}")
    print(f"¿Tiene campos deportivos? = {self._tiene_campos_deportivos}")
    print()

class CasaIndependiente(CasaUrbana):
```

```
    valor_area = 3000000 # Atributo de clase

    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str,
                 numero_habitaciones: int, numero_baños: int, numero_pisos: int):
        super().__init__(identificador_inmobiliario, area, direccion,
                         numero_habitaciones, numero_baños, numero_pisos)

    def imprimir(self):
        super().imprimir()
        print()

class Local(Inmueble):
    class TipoLocal(Enum):
        INTERNO = "Interno"
        CALLE = "Calle"

    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str, tipo_local: TipoLocal):
        super().__init__(identificador_inmobiliario, area, direccion)
        self._tipo_local = tipo_local

    def imprimir(self):
        super().imprimir()
        print(f"Tipo de local = {self._tipo_local.value}")

class LocalComercial(Local):
    valor_area = 3000000 # Atributo de clase

    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str,
                 tipo_local: Local.TipoLocal, centro_comercial: str):
        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 # Atributo de clase

    def __init__(self, identificador_inmobiliario: int, area: int, direccion: str,
                 tipo_local: Local.TipoLocal, es_gobierno: bool):
        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__":
    print("Datos apartamento")
    apto1 = ApartamentoFamiliar(103067, 120, "Avenida Santander 45-45", 3, 2, 200000)
    apto1.calcular_precio_venta(apto1.valor_area)
    apto1.imprimir()

    print("Datos apartaestudio")
    apto2 = Apartaestudio(12354, 50, "Avenida Caracas 30-15")
    apto2.calcular_precio_venta(apto2.valor_area)
    apto2.imprimir()
```

Ejecución:

Datos apartamento

Identificador inmobiliario = 103067

Área = 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 aparta estudio

Identificador inmobiliario = 12354

Área = 50

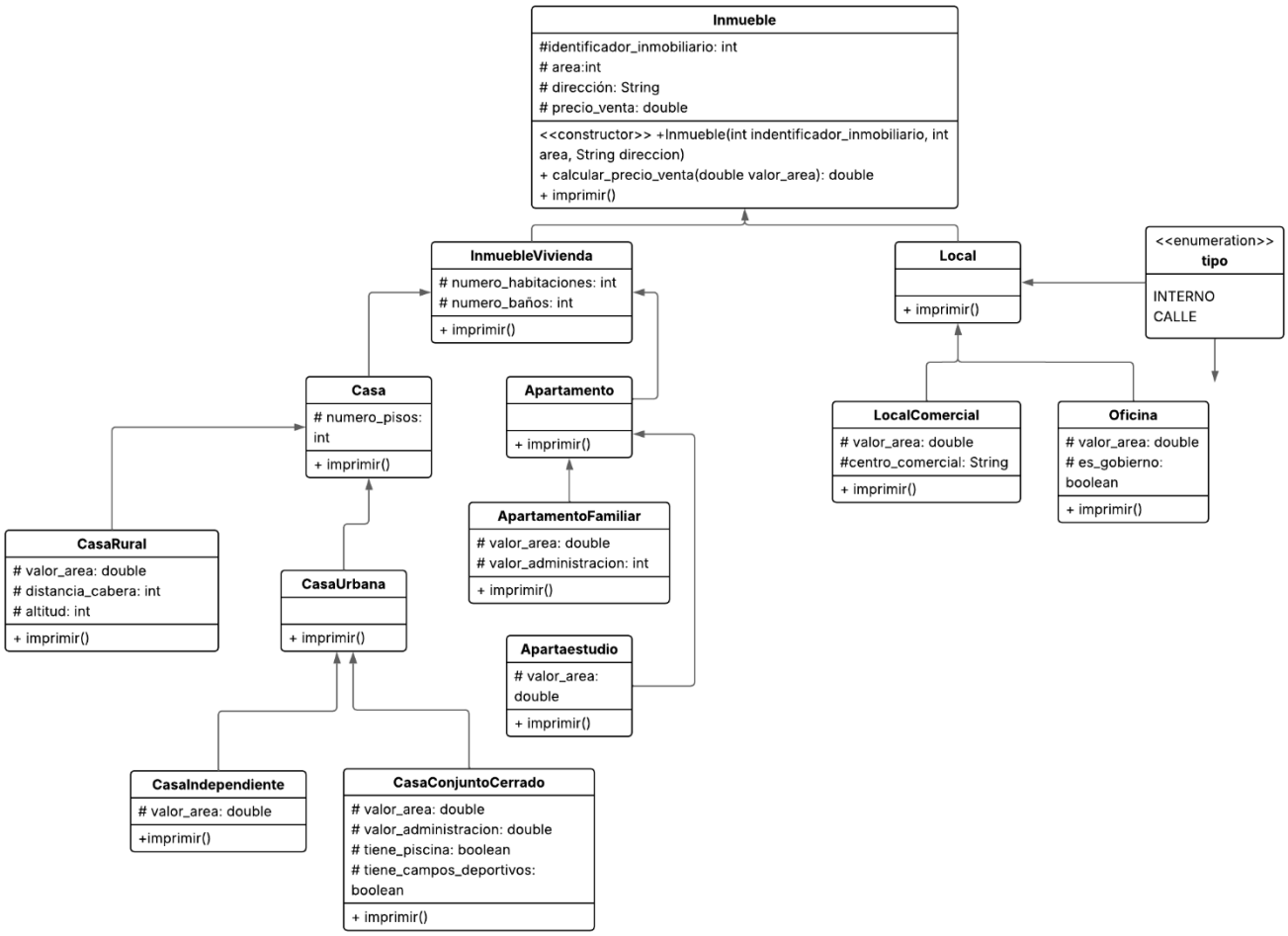
Dirección = Avenida Caracas 30-15

Precio de venta = $75,000,000.00

Número de habitaciones = 1

Número de baños = 1

Diagrama de clases:



Punto 3. Página 227

Código fuente:

```
class Mascota:
  def __init__(self, nombre, edad, color):
    self.nombre = nombre
    self.edad = edad
    self.color = color
```

```
class Perro(Mascota):
    def __init__(self, nombre, edad, color, peso, muerde):
        super().__init__(nombre, edad, color)
        self.peso = peso
        self.muerde = muerde

    @staticmethod
    def sonido():
        print("Los perros ladran")



# Categorías de perros según el tamañp
class PerroPequeno(Perro):
    pass

class PerroMediano(Perro):
    pass

class PerroGrande(Perro):
    pass

# Razas de perros pequeños
class Caniche(PerroPequeno):
    pass

class YorkshireTerrier(PerroPequeno):
    pass
```

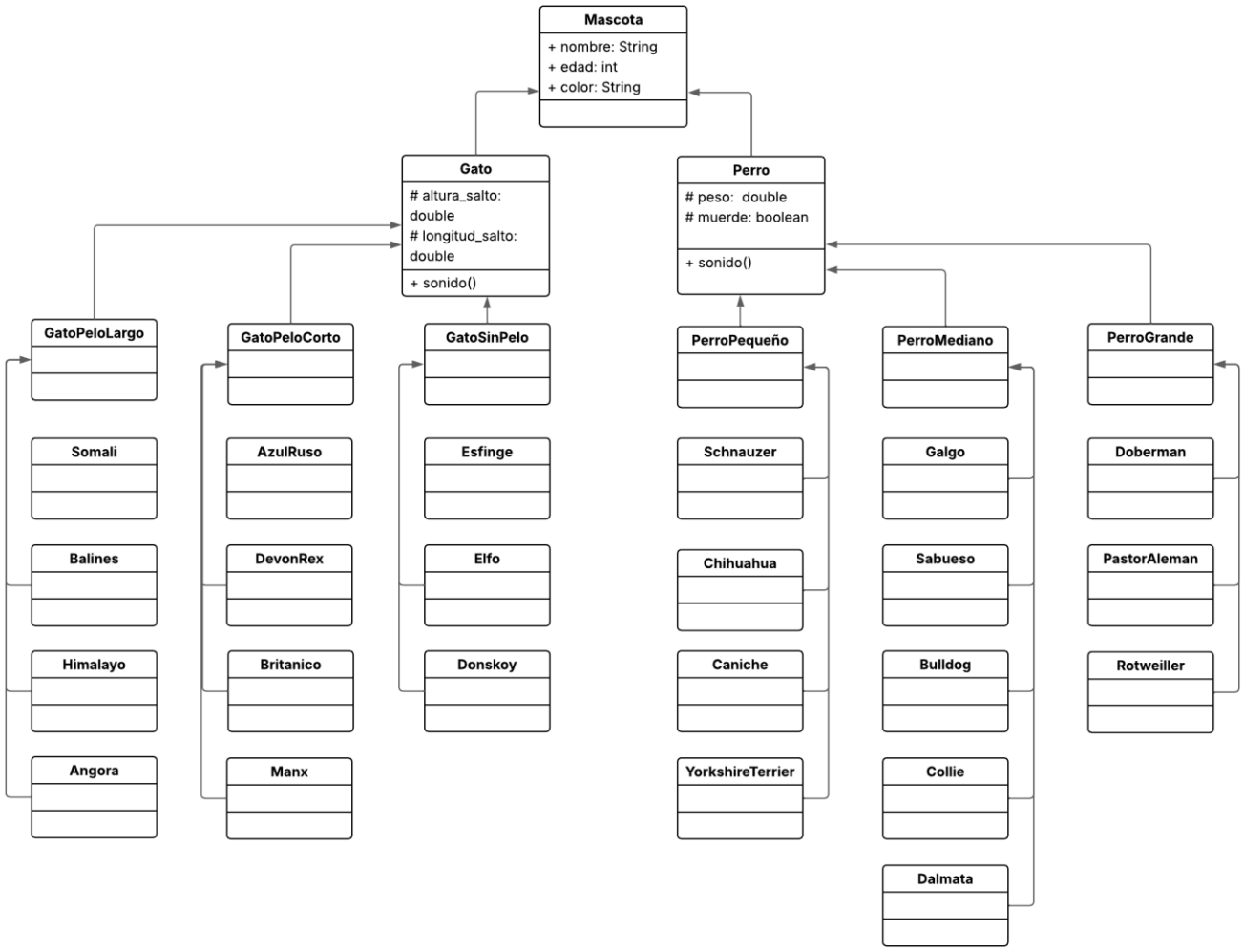
```
class Schnauzer(PerroPequeno):
    pass

class Chihuahua(PerroPequeno):
    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 Rotweiller(PerroGrande):
  pass

class Gato(Mascota):
  def __init__(self, nombre, edad, color, altura_salto, longitud_salto):
    super().__init__(nombre, edad, color)
    self.altura_salto = altura_salto
    self.longitud_salto = longitud_salto

  @staticmethod
  def sonido():
    print("Los gatos maúllan y ronronean")



# Categorías de gatos según el tipo de pelaje
class GatoSinPelo(Gato):
  pass

class GatoPeloLargo(Gato):
  pass

class GatoPeloCorto(Gato):
  pass

# Razas de gatos sin pelaje
class Esfinge(GatoSinPelo):
  pass
```

```
class Elfo(GatoSinPelo):
  pass

class Donskoy(GatoSinPelo):
  pass

# Razas de gatos con el pelaje corto
class Angora(GatoPeloLargo):
  pass

class Himalayo(GatoPeloLargo):
  pass

class Balines(GatoPeloLargo):
  pass

class Somali(GatoPeloLargo):
  pass

# Razas de gatos con el pelaje corto
class AzulRuso(GatoPeloCorto):
  pass

class Britanico(GatoPeloCorto):
  pass

class Manx(GatoPeloCorto):
```

```
    pass

class DevonRex(GatoPeloCorto):
    pass
```

Diagrama de clases:



Punto 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
```

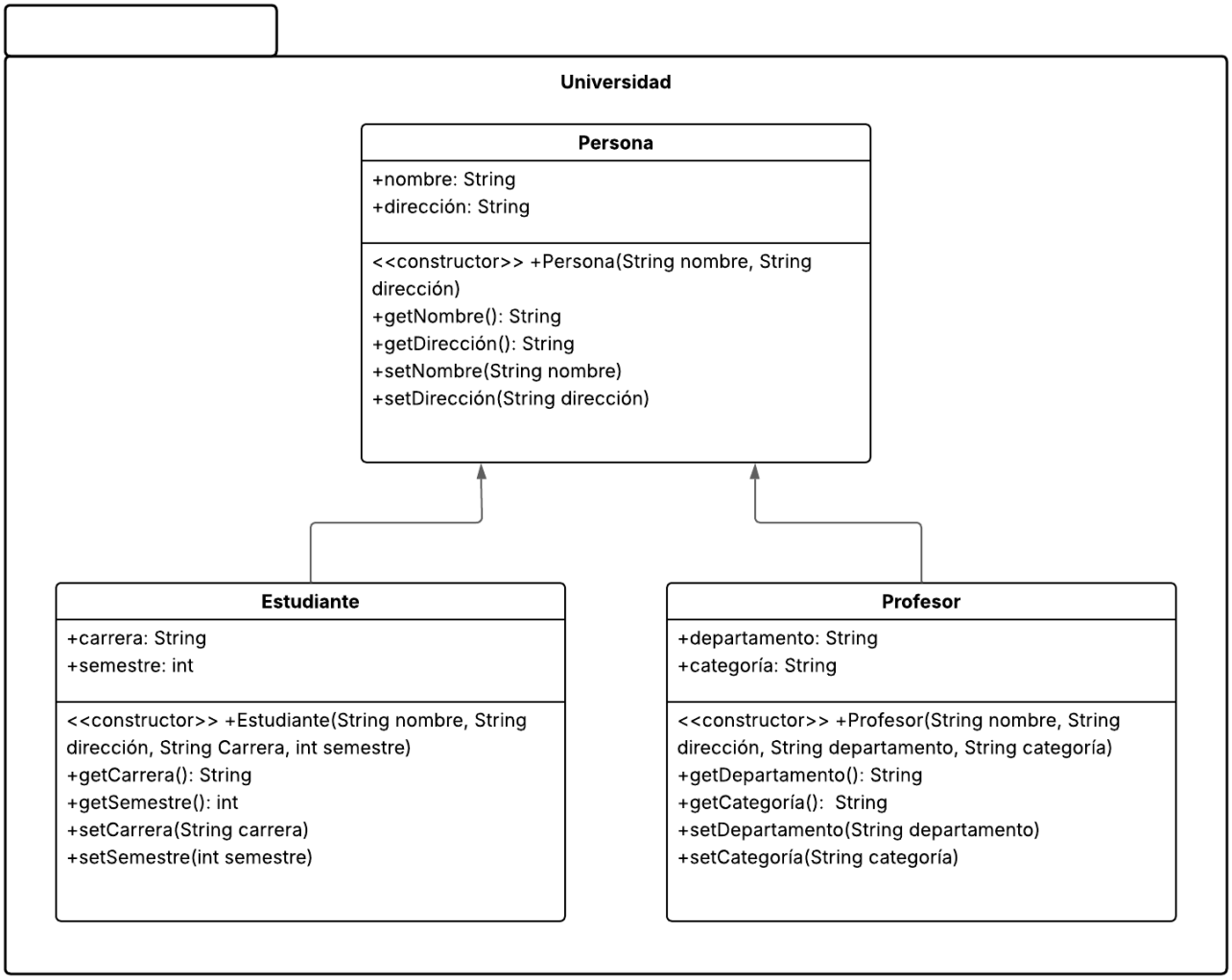
```
    def setCarrera(self, carrera: str):
        self._carrera = carrera

    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

# Ejemplo de uso:
if __name__ == "__main__":
    estudiante1 = Estudiante("Juan Muriel", "Calle 123", "Ingeniería administrativa", 1)
    Estudiante.setCarrera(estudiante1, "Matemáticas")
```

```
Estudiante.setSemestre(estudiante1, 7)
print(f"Nombre del estudiante: {estudiante1.getNombre()}")
print(f"Carrera del estudiante: {estudiante1.getCarrera()}")

profesor1 = Profesor("Manuela Bastidas", "Avenida 456", "Zootécnia", "Ocasional")
Profesor.setNombre(profesor1, "Rodney Jaramillo")
Profesor.setDepartamento(profesor1, "Matemáticas")
Profesor.setCategoria(profesor1, "Titular")
print(f"Nombre del profesor: {profesor1.getNombre()}")
print(f"Departamento del profesor: {profesor1.getDepartamento()}")
```

Ejecución:

```
Nombre del estudiante: Juan Muriel
Carrera del estudiante: Matemáticas
Nombre del profesor: Rodney Jaramillo
Departamento del profesor: Matemáticas
```

Diagrama de clases:



Punto 5. 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()
```

Ejecución:

Es un profesor titular.

Diagrama de clases: