Ejercicio 1

In [3]:

```python
class Cuenta:
    def __init__(self, saldo_inicial:float, tasa_anual:float):
        self.saldo = float(saldo_inicial)
        self.num_consignaciones = 0
        self.num_retiros = 0
        self.tasa_anual = float(tasa_anual)
        self.tasa_mensual = self.tasa_anual / 12
        self.comision_mensual = 0.0

    def consignar(self, cantidad):
        if cantidad > 0:
            self.saldo += cantidad
            self.num_consignaciones += 1
        else:
            print("La cantidad a consignar debe ser positiva.")

    def retirar(self, cantidad):
        if 0 < cantidad <= self.saldo:
            self.saldo -= cantidad
            self.num_retiros += 1
        else:
            print("Fondos insuficientes o cantidad inválida.")

    def calcular_interes_mensual(self):
        interes_mensual = self.saldo * (self.tasa_mensual / 100)
        self.saldo += interes_mensual

    def extracto_mensual(self):
        self.saldo -= self.comision_mensual
        self.calcular_interes_mensual()

    def imprimir(self):
        print("Saldo actual: ${:.2f}".format(self.saldo))
        print("Número de consignaciones:", self.num_consignaciones)
        print("Número de retiros:", self.num_retiros)
        print("Tasa anual (%):", self.tasa_anual)
        print("Comisión mensual: ${:.2f}".format(self.comision_mensual))

class CuentaAhorros(Cuenta):
    def __init__(self, saldo_inicial:float, tasa_anual:float):
        super().__init__(saldo_inicial, tasa_anual)
        self.activa = self.saldo >= 10000

    def consignar(self, cantidad):
        if self.activa:
            super().consignar(cantidad)
        else:
            print("La cuenta no está activa. No se puede consignar.")

    def retirar(self, cantidad):
        if self.activa:
            super().retirar(cantidad)
        else:
            print("La cuenta no está activa. No se puede retirar.")

    def extracto_mensual(self):
        self.comision_mensual += (self.num_retiros - 4)*1000*(self.num_retiros > 4)
        super().extracto_mensual()
        self.activa = self.saldo >= 10000

    def imprimir(self):
        print("Saldo actual: ${:.2f}".format(self.saldo))
        print("Comisión mensual: ${:.2f}".format(self.comision_mensual))
        print("Número de transacciones:", self.num_consignaciones + self.num_retiros)


class CuentaCorriente(Cuenta):
    def __init__(self, saldo_inicial:float, tasa_anual:float):
        super().__init__(saldo_inicial, tasa_anual)
        self.sobregiro = 0

    def retirar(self, cantidad):
        if cantidad > 0:
            if self.saldo >= cantidad:
                super().retirar(cantidad)
            else:
                self.sobregiro = cantidad - self.saldo
                self.saldo = 0
                self.num_retiros += 1
```

```
        else:
            print("La cantidad a retirar debe ser positiva.")

    def consignar(self, cantidad):
        if cantidad > 0:
            if self.sobregiro > 0:
                if cantidad >= self.sobregiro:
                    self.saldo += (cantidad - self.sobregiro)
                    self.sobregiro = 0
                else:
                    self.sobregiro -= cantidad
            else:
                super().consignar(cantidad)
        else:
            print("La cantidad a consignar debe ser positiva.")

    def imprimir(self):
        print("Saldo actual: ${:.2f}".format(self.saldo))
        print("Comisión mensual: ${:.2f}".format(self.comision_mensual))
        print("Número de transacciones:", self.num_consignaciones + self.num_retiros)
        print("Sobregiro: ${:.2f}".format(self.sobregiro))


if __name__ == "__main__":
    cuenta_ahorros = CuentaAhorros(15000, 12)
    cuenta_corriente = CuentaCorriente(5000, 12)

    cuenta_ahorros.consignar(2000)
    cuenta_ahorros.retirar(500)
    cuenta_ahorros.extracto_mensual()
    cuenta_ahorros.imprimir()
    print("")
    cuenta_corriente.consignar(3000)
    cuenta_corriente.retirar(10000)
    cuenta_corriente.extracto_mensual()
    cuenta_corriente.imprimir()
```

```
Saldo actual: $16665.00
Comisión mensual: $0.00
Número de transacciones: 2

Saldo actual: $0.00
Comisión mensual: $0.00
Número de transacciones: 2
Sobregiro: $2000.00
```

Ejercicio 2

In [8]:

```
from enum import Enum

class Localizacion(Enum):
    INTERNO = "Interno"
    CALLE = "Calle"

class Inmueble:
    def __init__(self, id_inmobiliario:int, area:int, direccion:str):
        self.id_inmobiliario = id_inmobiliario
        self.area = area
        self.direccion = direccion
        self.precio_venta = 0.0

    def calcular_precio_venta(self, precio_m2):
        self.precio_venta = self.area * precio_m2

    def imprimir(self):
        print("Identificador inmobiliario =", self.id_inmobiliario)
        print("Area =", self.area)
        print("Dirección =", self.direccion)
        print("Precio de venta = $", self.precio_venta)


class InmuebleVivienda(Inmueble):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int):
        super().__init__(id_inmobiliario, area, direccion)
        self.num_habitaciones = num_habitaciones
        self.num_banos = num_banos

    def imprimir(self):
        super().imprimir()
        print("Número de habitaciones =", self.num_habitaciones)
        print("Número de baños =", self.num_banos)
```

```python
class Casa(InmuebleVivienda):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, num_p
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos)
        self.num_pisos = num_pisos

    def imprimir(self):
        super().imprimir()
        print("Número de pisos =", self.num_pisos)


class CasaRural(Casa):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, num_p
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos, num_pisos)
        self.distancia_cabecera = distancia_cabecera
        self.altitud = altitud
        self.valor_area = 1500000

    def imprimir(self):
        super().imprimir()
        print("Distancia a cabecera municipal =", self.distancia_cabecera)
        print("Altitud =", self.altitud)


class CasaUrbana(Casa):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, num_p
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos, num_pisos)

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


class CasaConjuntoCerrado(CasaUrbana):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, num_p
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos, num_pisos)
        self.valor_administracion = valor_administracion
        self.tiene_piscina = tiene_piscina
        self.tiene_campo_deportivo = tiene_campo_deportivo
        self.areas_comunes = tiene_piscina or tiene_campo_deportivo
        self.valor_area = 2500000

    def imprimir(self):
        super().imprimir()
        print("Valor administración =", self.valor_administracion)
        print("Incluye áreas comunes =", self.areas_comunes)


class CasaIndependiente(CasaUrbana):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, num_p
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos, num_pisos)
        self.valor_area = 3000000

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


class Apartamento(InmuebleVivienda):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, valor_
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos)
        self.valor_administracion = valor_administracion

    def imprimir(self):
        super().imprimir()
        print("Valor administración =", self.valor_administracion)


class Apartaestudio(Apartamento):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, valor_administracion:int):
        super().__init__(id_inmobiliario, area, direccion, 1, 1, valor_administracion)
        self.valor_area = 1500000

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


class ApartamentoFamiliar(Apartamento):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, num_habitaciones:int, num_banos:int, valor_
        super().__init__(id_inmobiliario, area, direccion, num_habitaciones, num_banos, valor_administracion)
        self.valor_area = 2000000

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

```python
class Local(Inmueble):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str, localizacion: Localizacion):
        super().__init__(id_inmobiliario, area, direccion)
        self.localizacion = localizacion

    def imprimir(self):
        super().imprimir()
        print("Localización =", self.localizacion.value)

class LocalComercial(Local):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str,  localizacion: Localizacion, centro_comerci
        super().__init__(id_inmobiliario, area, direccion, localizacion)
        self.centro_comercial = centro_comercial
        self.valor_area = 3000000

    def imprimir(self):
        super().imprimir()
        print("Centro comercial =", self.centro_comercial)


class Oficina(Local):
    def __init__(self, id_inmobiliario:int, area:int, direccion:str,  localizacion: Localizacion, es_gobierno:b
        super().__init__(id_inmobiliario, area, direccion, localizacion)
        self.es_gobierno = es_gobierno
        self.valor_area = 3500000

    def imprimir(self):
        super().imprimir()
        print("Es del gobierno =", self.es_gobierno)


if __name__ == "__main__":
    casa_rural = CasaRural(1, 120, "Vereda El Sol", 3, 2, 2, 15, 1800)
    casa_rural.calcular_precio_venta(casa_rural.valor_area)
    casa_rural.imprimir()
    print("\n---\n")

    oficina = Oficina(2, 80, "Carrera 45 #12-34", Localizacion.INTERNO, True)
    oficina.calcular_precio_venta(oficina.valor_area)
    oficina.imprimir()
    print("\n---\n")

    apartaestudio = Apartaestudio(3, 45, "Calle 100 #20-15", 150000)
    apartaestudio.calcular_precio_venta(apartaestudio.valor_area)
    apartaestudio.imprimir()
```

```
Identificador inmobiliario = 1
Area = 120
Dirección = Vereda El Sol
Precio de venta = $ 180000000
Número de habitaciones = 3
Número de baños = 2
Número de pisos = 2
Distancia a cabecera municipal = 15
Altitud = 1800

---

Identificador inmobiliario = 2
Area = 80
Dirección = Carrera 45 #12-34
Precio de venta = $ 280000000
Localización = Interno
Es del gobierno = True

---

Identificador inmobiliario = 3
Area = 45
Dirección = Calle 100 #20-15
Precio de venta = $ 67500000
Número de habitaciones = 1
Número de baños = 1
Valor administración = 150000
```

Ejercicio 3

In [ ]:

```python
# Clase base Mascota
class Mascota:
    def __init__(self, nombre:str, edad:int, color:str):
        self.nombre = nombre
        self.edad = edad
        self.color = color
```

```python
# Clase Perro
class Perro(Mascota):
    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():
        print("Los perros ladran")

# Subclases de Perro según tamaño y raza
class PerroPequeno(Perro):
    pass

class Caniche(PerroPequeno):
    pass

class YorkshireTerrier(PerroPequeno):
    pass

class Schnauzer(PerroPequeno):
    pass

class Chihuahua(PerroPequeno):
    pass

class PerroMediano(Perro):
    pass

class Collie(PerroMediano):
    pass

class Dalmata(PerroMediano):
    pass

class Bulldog(PerroMediano):
    pass

class Galgo(PerroMediano):
    pass

class Sabueso(PerroMediano):
    pass

class PerroGrande(Perro):
    pass

class PastorAleman(PerroGrande):
    pass

class Doberman(PerroGrande):
    pass

class Rotweiller(PerroGrande):
    pass

# Clase Gato
class Gato(Mascota):
    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():
        print("Los gatos maúllan y ronronean")

# Subclases de Gato según tipo de pelaje y raza
class GatoSinPelo(Gato):
    pass

class Esfinge(GatoSinPelo):
    pass

class Elfo(GatoSinPelo):
    pass

class Donskoy(GatoSinPelo):
    pass

class GatoPeloLargo(Gato):
```

```
    pass

class Angora(GatoPeloLargo):
    pass

class Himalayo(GatoPeloLargo):
    pass

class Balines(GatoPeloLargo):
    pass

class Somali(GatoPeloLargo):
    pass

class GatoPeloCorto(Gato):
    pass

class AzulRuso(GatoPeloCorto):
    pass

class Britanico(GatoPeloCorto):
    pass

class Manx(GatoPeloCorto):
    pass

class DevonRex(GatoPeloCorto):
    pass


if __name__ == "__main__":
    # Crear un perro pequeño de raza Caniche
    perro1 = Caniche(nombre="Fido", edad=3, color="blanco", peso=5.0, muerde=False)
    print(f"Perro: {perro1.nombre}, Edad: {perro1.edad}, Color: {perro1.color}, Peso: {perro1.peso}kg, ¿Muerde?
    Perro.sonido()

    print("")

    # Crear un gato de raza Himalayo (pelo largo)
    gato1 = Himalayo(nombre="Misha", edad=2, color="gris", altura_salto=1.2, longitud_salto=2.0)
    print(f"Gato: {gato1.nombre}, Edad: {gato1.edad}, Color: {gato1.color}, Altura de salto: {gato1.altura_salt
    Gato.sonido()
```

```
Perro: Fido, Edad: 3, Color: blanco, Peso: 5.0kg, ¿Muerde?: False
Los perros ladran

Gato: Misha, Edad: 2, Color: gris, Altura de salto: 1.2m, Longitud de salto: 2.0m
Los gatos maúllan y ronronean
```

Ejercicio 4

In [ ]:

```
class Persona:
    def __init__(self, nombre:str, direccion:str):
        self.nombre = nombre
        self.direccion = direccion

    def getNombre(self) -> str:
        return self.nombre

    def getDireccion(self) -> str:
        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) -> str:
        return self.departamento

    def getCategoria(self) -> str:
        return self.categoria

    def setDepartamento(self, departamento:str):
        self.departamento = departamento

    def setCategoria(self, categoria:str):
        self.categoria = categoria


if __name__ == "__main__":
    estudiante = Estudiante("Ana Pérez", "Calle 123", "Ingeniería", 5)
    print("----- ESTUDIANTE -----")
    print("Nombre:", estudiante.getNombre())
    print("Dirección:", estudiante.getDireccion())
    print("Carrera:", estudiante.getCarrera())
    print("Semestre:", estudiante.getSemestre())

    estudiante.setCarrera("Matemáticas")
    estudiante.setSemestre(6)
    print("\nDespués de modificar carrera y semestre:")
    print("Carrera:", estudiante.getCarrera())
    print("Semestre:", estudiante.getSemestre())

    profesor = Profesor("Carlos Gómez", "Avenida 45", "Física", "Titular")
    print("\n----- PROFESOR -----")
    print("Nombre:", profesor.getNombre())
    print("Dirección:", profesor.getDireccion())
    print("Departamento:", profesor.getDepartamento())
    print("Categoría:", profesor.getCategoria())

    profesor.setDepartamento("Matemáticas")
    profesor.setCategoria("Asociado")
    print("\nDespués de modificar departamento y categoría:")
    print("Departamento:", profesor.getDepartamento())
    print("Categoría:", profesor.getCategoria())
```

```
----- ESTUDIANTE -----
Nombre: Ana Pérez
Dirección: Calle 123
Carrera: Ingeniería
Semestre: 5

Después de modificar carrera y semestre:
Carrera: Matemáticas
Semestre: 6

----- PROFESOR -----
Nombre: Carlos Gómez
Dirección: Avenida 45
Departamento: Física
Categoría: Titular

Después de modificar departamento y categoría:
Departamento: Matemáticas
Categoría: Asociado
```

Ejercicio 5

In [12]:

```
class Profesor:
    def imprimir(self):
        print("Es un profesor")

class ProfesorTitular(Profesor):
    def imprimir(self):
        print("Es un profesor titular")

profesores= [Profesor(), ProfesorTitular()]
for profesor in profesores:
    profesor.imprimir()
```

```
Es un profesor
Es un profesor titular
```