Actividad # 2 - Programación Orientada a Objetos

Grupo 3

Javier Danilo Castro Faccetti

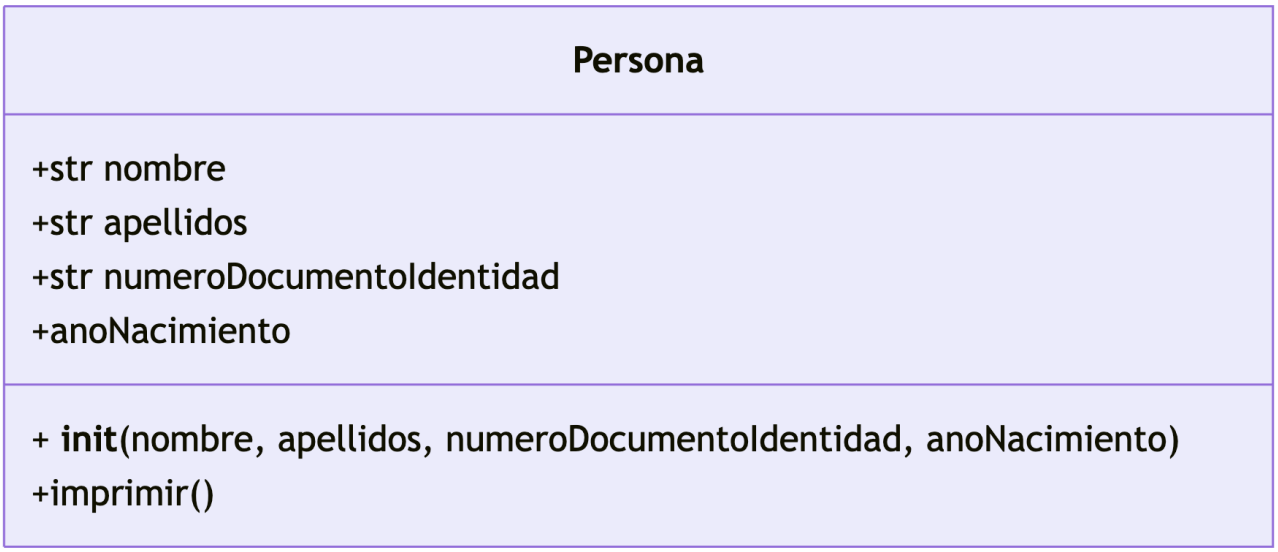
C.C.: 1047458731

Ejercicio 2.1 (Página 63)

Enlace a GitHub.

```
class Persona:

  def __init__(self, nombre, apellidos, numeroDocumentoIdentidad, anoNacimiento):
    self.nombre = nombre
    self.apellidos = apellidos
    self.numeroDocumentoIdentidad = numeroDocumentoIdentidad
    self.anoNacimiento = anoNacimiento

  def imprimir(self):
    print(f"Nombre = {self.nombre}")
    print(f"Apellidos = {self.apellidos}")
    print(f"Número de documento de identidad = {self.numeroDocumentoIdentidad}")
    print(f"Año de nacimiento = {self.anoNacimiento}")

if __name__ == "__main__":
  p1 = Persona("Pedro", "Pérez", "1053121010", 1998)
  p2 = Persona("Luis", "León", "1053223344", 2001)
  p1.imprimir()
  p2.imprimir()
```

```
Nombre = Pedro
Apellidos = Pérez
Número de documento de identidad = 1053121010
Año de nacimiento = 1998
Nombre = Luis
Apellidos = León
Número de documento de identidad = 1053223344
Año de nacimiento = 2001
```

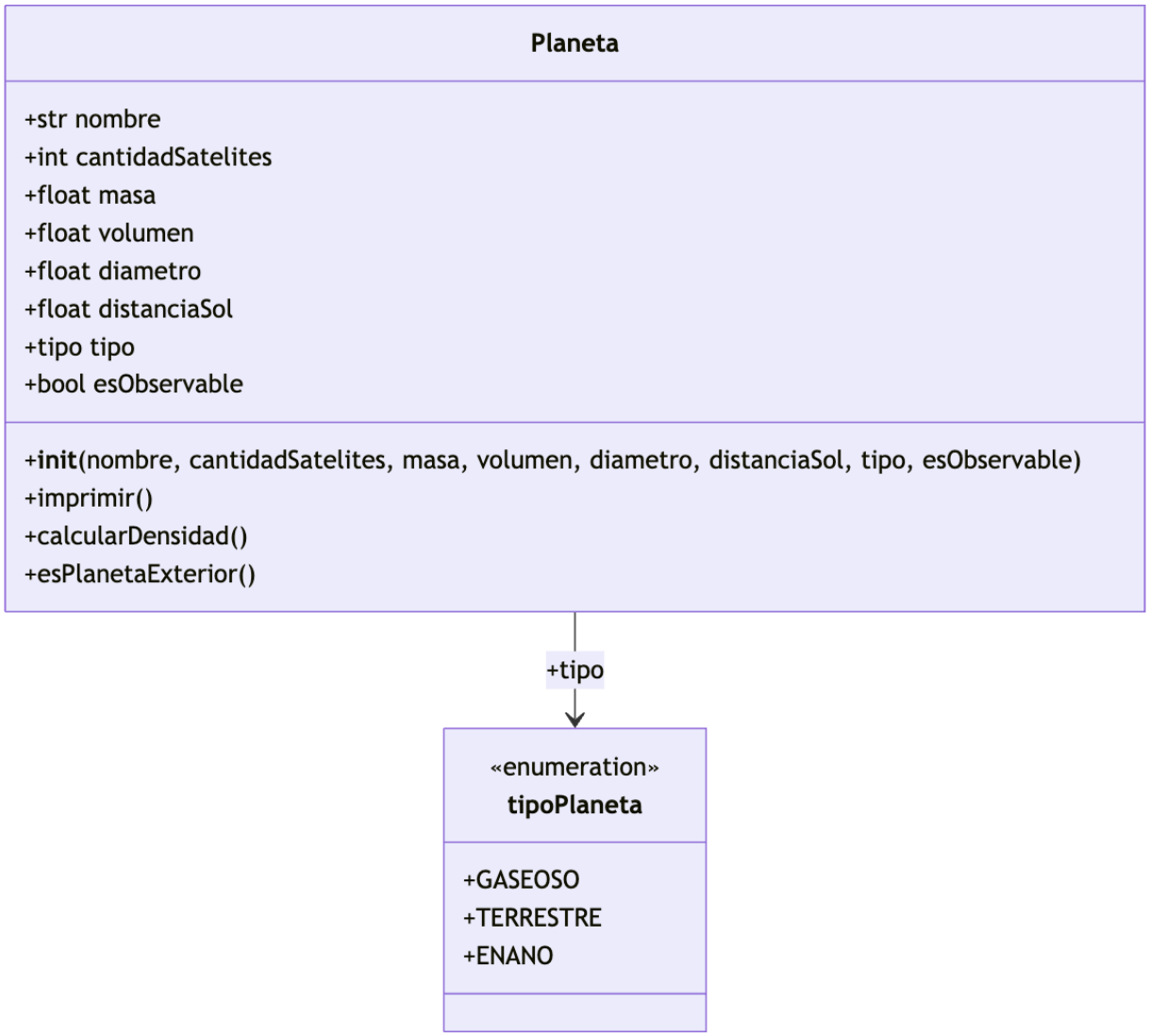


Ejercicio 2.2 (Página 66)

Enlace a GitHub.

```
from enum import Enum

class tipoPlaneta(Enum):

  GASEOSO = "GASEOSO"
  TERRESTRE = "TERRESTRE"
  ENANO = "ENANO"

class Planeta:

  def __init__(self, nombre, cantidadSatelites, masa, volumen, diametro, distanciaSol, tipo, esObservable):
    self.nombre = nombre
    self.cantidadSatelites = cantidadSatelites
    self.masa = masa
    self.volumen = volumen
    self.diametro = diametro
    self.distanciaSol = distanciaSol
    self.tipo = tipo.value
    self.esObservable = esObservable

  def imprimir(self):
    print(f"Nombre del planeta = {self.nombre}")
    print(f"Cantidad de satélites = {self.cantidadSatelites}")
    print(f"Masa del planeta = {self.masa}")
    print(f"Volumen del planeta = {self.volumen}")
    print(f"Diámetro del planeta = {self.diametro}")
    print(f"Distancia al sol = {self.distanciaSol}")
    print(f"Tipo de planeta = {self.tipo}")
    print(f"Es observable = {self.esObservable}")

  def calcularDensidad(self):
    return self.masa / self.volumen

  def esPlanetaExterior(self):
    limite= 149597870 * 3.4
    if self.distanciaSol > limite:
      return True
    else:
      return False

if __name__ == "__main__":
  p1 = Planeta("Tierra", 1, 5.9736E24, 1.08321E12, 12742, 150000000, tipoPlaneta.TERRESTRE, True)
  p1.imprimir()
  print(f"Densidad del planeta = {p1.calcularDensidad()}")
  print(f"Es planeta exterior = {p1.esPlanetaExterior()}")
  p2 = Planeta("Júpiter", 79, 1.899E27, 1.4313E15, 139820, 750000000, tipoPlaneta.GASEOSO, True)
  p2.imprimir()
  print(f"Densidad del planeta = {p2.calcularDensidad()}")
  print(f"Es planeta exterior = {p2.esPlanetaExterior()}")
```

```
Nombre del planeta = Tierra
Cantidad de satélites = 1
Masa del planeta = 5.9736e+24
Volumen del planeta = 1083210000000.0
Diámetro del planeta = 12742
Distancia al sol = 150000000
Tipo de planeta = TERRESTRE
Es observable = True
Densidad del planeta = 5514720137369.484
Es planeta exterior = False
Nombre del planeta = Júpiter
Cantidad de satélites = 79
Masa del planeta = 1.899e+27
Volumen del planeta = 1431300000000000.0
Diámetro del planeta = 139820
Distancia al sol = 750000000
Tipo de planeta = GASEOSO
Es observable = True
Densidad del planeta = 1326765877174.5964
Es planeta exterior = True
```



Ejercicio 2.3 (Página 66)

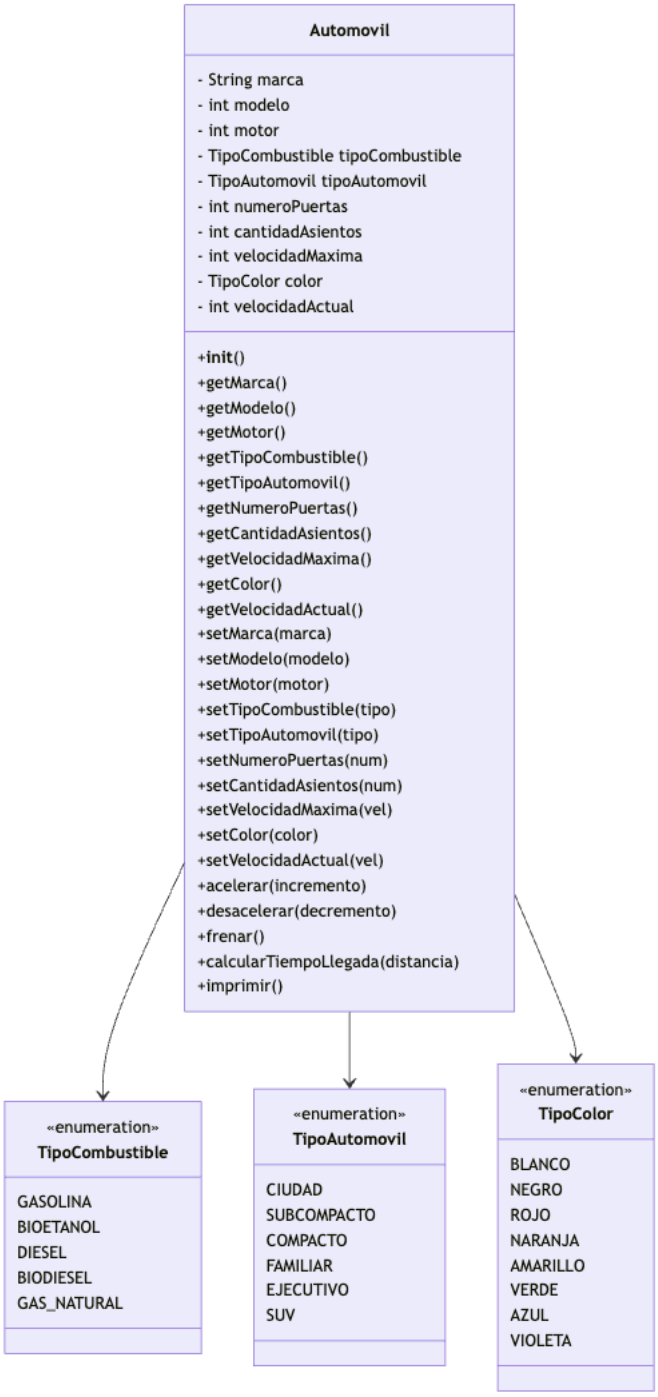
Enlace a GitHub.

```
from enum import Enum

class TipoCombustible(Enum):
    GASOLINA = 1
    BIOETANOL = 2
    DIESEL = 3
    BIODIESEL = 4
    GAS_NATURAL = 5

class TipoAutomovil(Enum):
    CIUDAD = 1
    SUBCOMPACTO = 2
    COMPACTO = 3
    FAMILIAR = 4
    EJECUTIVO = 5
    SUV = 6

class TipoColor(Enum):
    BLANCO = 1
    NEGRO = 2
    ROJO = 3
    NARANJA = 4
    AMARILLO = 5
    VERDE = 6
```

```
    AZUL = 7
    VIOLETA = 8

class Automovil:
    def __init__(self, marca, modelo, motor, tipoCombustible, tipoAutomovil,
                 numeroPuertas, cantidadAsientos, velocidadMaxima, color):
        self.marca = marca
        self.modelo = modelo
        self.motor = motor
        self.tipoCombustible = tipoCombustible
        self.tipoAutomovil = tipoAutomovil
        self.numeroPuertas = numeroPuertas
        self.cantidadAsientos = cantidadAsientos
        self.velocidadMaxima = velocidadMaxima
        self.color = color
        self.velocidadActual = 0

    def getMarca(self):
        return self.marca

    def getModelo(self):
        return self.modelo

    def getMotor(self):
        return self.motor

    def getTipoCombustible(self):
        return self.tipoCombustible

    def getTipoAutomovil(self):
        return self.tipoAutomovil

    def getNumeroPuertas(self):
        return self.numeroPuertas

    def getCantidadAsientos(self):
        return self.cantidadAsientos

    def getVelocidadMaxima(self):
        return self.velocidadMaxima

    def getColor(self):
        return self.color

    def getVelocidadActual(self):
        return self.velocidadActual

    def setMarca(self, marca):
        self.marca = marca

    def setModelo(self, modelo):
        self.modelo = modelo

    def setMotor(self, motor):
        self.motor = motor

    def setTipoCombustible(self, tipoCombustible):
        self.tipoCombustible = tipoCombustible

    def setTipoAutomovil(self, tipoAutomovil):
        self.tipoAutomovil = tipoAutomovil

    def setNumeroPuertas(self, numeroPuertas):
        self.numeroPuertas = numeroPuertas

    def setCantidadAsientos(self, cantidadAsientos):
        self.cantidadAsientos = cantidadAsientos

    def setVelocidadMaxima(self, velocidadMaxima):
        self.velocidadMaxima = velocidadMaxima

    def setColor(self, color):
        self.color = color

    def setVelocidadActual(self, velocidadActual):
        self.velocidadActual = velocidadActual
```

```
    def acelerar(self, incrementoVelocidad):
        if self.velocidadActual + incrementoVelocidad < self.velocidadMaxima:
            self.velocidadActual += incrementoVelocidad
        else:
            print("No se puede incrementar a una velocidad superior a la máxima del automóvil.")

    def desacelerar(self, decrementoVelocidad):
        if (self.velocidadActual - decrementoVelocidad) > 0:
            self.velocidadActual -= decrementoVelocidad
        else:
            print("No se puede decrementar a una velocidad negativa.")

    def frenar(self):
        self.velocidadActual = 0

    def calcularTiempoLlegada(self, distancia):
        if self.velocidadActual == 0:
            print("La velocidad actual es cero, no se puede calcular el tiempo.")
            return None
        return distancia / self.velocidadActual

    def imprimir(self):
        print(f"Marca = {self.marca}")
        print(f"Modelo = {self.modelo}")
        print(f"Motor = {self.motor}")
        print(f"Tipo de combustible = {self.tipoCombustible.name}")
        print(f"Tipo de automóvil = {self.tipoAutomovil.name}")
        print(f"Número de puertas = {self.numeroPuertas}")
        print(f"Cantidad de asientos = {self.cantidadAsientos}")
        print(f"Velocidad máxima = {self.velocidadMaxima}")
        print(f"Color = {self.color.name}")


if __name__ == "__main__":
    auto1 = Automovil("Ford", 2018, 3, TipoCombustible.DIESEL, TipoAutomovil.EJECUTIVO, 5, 6, 250, TipoColor.NEGRO)
    auto1.imprimir()
    auto1.setVelocidadActual(100)
    print(f"Velocidad actual = {auto1.getVelocidadActual()}")
    auto1.acelerar(20)
    print(f"Velocidad actual = {auto1.getVelocidadActual()}")
    auto1.desacelerar(50)
    print(f"Velocidad actual = {auto1.getVelocidadActual()}")
    auto1.frenar()
    print(f"Velocidad actual = {auto1.getVelocidadActual()}")
    auto1.desacelerar(20)
```

```
Marca = Ford
Modelo = 2018
Motor = 3
Tipo de combustible = DIESEL
Tipo de automóvil = EJECUTIVO
Número de puertas = 5
Cantidad de asientos = 6
Velocidad máxima = 250
Color = NEGRO
Velocidad actual = 100
Velocidad actual = 120
Velocidad actual = 70
Velocidad actual = 0
No se puede decrementar a una velocidad negativa.
```



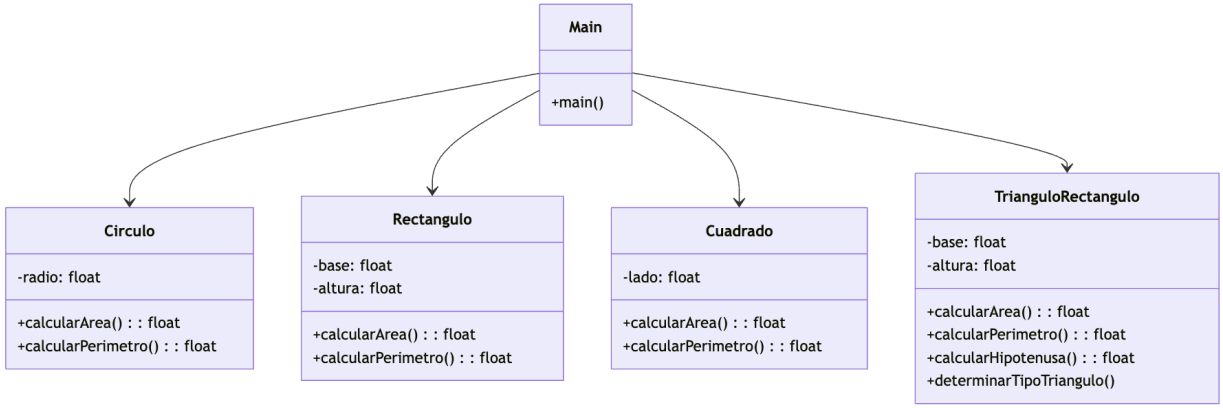
Ejercicio 2.4 (Página 86)

Enlace a GitHub.

```
import math

class Circulo:
    def __init__(self, radio):
        self.radio = radio

    def calcularArea(self):
        return math.pi * (self.radio ** 2)

    def calcularPerimetro(self):
        return 2 * math.pi * self.radio
```

```
class Rectangulo:
    def __init__(self, base, altura):
        self.base = base
        self.altura = altura

    def calcularArea(self):
        return self.base * self.altura

    def calcularPerimetro(self):
        return 2 * (self.base + self.altura)


class Cuadrado:
    def __init__(self, lado):
        self.lado = lado

    def calcularArea(self):
        return self.lado ** 2

    def calcularPerimetro(self):
        return 4 * self.lado


class TrianguloRectangulo:
    def __init__(self, base, altura):
        self.base = base
        self.altura = altura

    def calcularArea(self):
        return (self.base * self.altura) / 2

    def calcularHipotenusa(self):
        return math.sqrt(self.base ** 2 + self.altura ** 2)

    def calcularPerimetro(self):
        return self.base + self.altura + self.calcularHipotenusa()

    def determinarTipoTriangulo(self):
        hipotenusa = self.calcularHipotenusa()
        if self.base == self.altura == hipotenusa:
            print("Es un triángulo equilátero")
        elif self.base != self.altura and self.base != hipotenusa and self.altura != hipotenusa:
            print("Es un triángulo escaleno")
        else:
            print("Es un triángulo isósceles")

if __name__ == "__main__":
    figura1 = Circulo(2)
    figura2 = Rectangulo(1, 2)
    figura3 = Cuadrado(3)
    figura4 = TrianguloRectangulo(3, 5)
    print(f"El área del círculo es = {figura1.calcularArea()}")
    print(f"El perímetro del círculo es = {figura1.calcularPerimetro()}\n")
    print(f"El área del rectángulo es = {figura2.calcularArea()}")
    print(f"El perímetro del rectángulo es = {figura2.calcularPerimetro()}\n")
    print(f"El área del cuadrado es = {figura3.calcularArea()}")
    print(f"El perímetro del cuadrado es = {figura3.calcularPerimetro()}\n")
    print(f"El área del triángulo es = {figura4.calcularArea()}")
    print(f"El perímetro del triángulo es = {figura4.calcularPerimetro()}")
    figura4.determinarTipoTriangulo()
```

```
El área del círculo es = 12.566370614359172
El perímetro del círculo es = 12.566370614359172

El área del rectángulo es = 2
El perímetro del rectángulo es = 6

El área del cuadrado es = 9
El perímetro del cuadrado es = 12

El área del triángulo es = 7.5
El perímetro del triángulo es = 13.8309518948453
Es un triángulo escaleno
```



Ejercicio 2.5 (Página 95)

Enlace a GitHub.

```
from enum import Enum

class TipoCuenta(Enum):
    AHORROS = "AHORROS"
    CORRIENTE = "CORRIENTE"

class CuentaBancaria:
    def __init__(self, nombresTitular, apellidosTitular, numeroCuenta, tipoCuenta):
        self.nombresTitular = nombresTitular
        self.apellidosTitular = apellidosTitular
        self.numeroCuenta = numeroCuenta
        self.tipoCuenta = tipoCuenta
        self.saldo = 0.0

    def imprimir(self):
        print(f"Nombres del titular = {self.nombresTitular}")
        print(f"Apellidos del titular = {self.apellidosTitular}")
        print(f"Número de cuenta = {self.numeroCuenta}")
        print(f"Tipo de cuenta = {self.tipoCuenta.value}")
        print(f"Saldo = ${self.saldo:,.2f}")

    def consultarSaldo(self):
        print(f"El saldo actual es = ${self.saldo:,.2f}")

    def consignar(self, valor):
        if valor > 0:
            self.saldo += valor
            print(f"Se ha consignado ${valor:,.2f} en la cuenta. El nuevo saldo es ${self.saldo:,.2f}")
            return True
        else:
            print("El valor a consignar debe ser mayor que cero.")
            return False

    def retirar(self, valor):
        if valor > 0 and valor <= self.saldo:
            self.saldo -= valor
            print(f"Se ha retirado ${valor:,.2f} en la cuenta. El nuevo saldo es ${self.saldo:,.2f}")
            return True
        else:
            print("El valor a retirar debe ser menor o igual al saldo actual y mayor que cero.")
            return False

if __name__ == "__main__":
    cuenta = CuentaBancaria("Pedro", "Pérez", 123456789, TipoCuenta.AHORROS)
    cuenta.imprimir()
    cuenta.consignar(200000)
    cuenta.consignar(300000)
    cuenta.retirar(400000)
```

```
Nombres del titular = Pedro
Apellidos del titular = Pérez
Número de cuenta = 123456789
Tipo de cuenta = AHORROS
Saldo = $0.00
Se ha consignado $200,000.00 en la cuenta. El nuevo saldo es $200,000.00
Se ha consignado $300,000.00 en la cuenta. El nuevo saldo es $500,000.00
Se ha retirado $400,000.00 en la cuenta. El nuevo saldo es $100,000.00
```