UNIVERSIDAD NACIONAL DE COLOMBIA

Programación orientada a objetos

Actividad 2

Integrantes:

Maria Fernanda Valencia Jimenez

Juan Felipe Muriel Mosquera

Profesor:

Walter Hugo Arboleda Mazo

Fecha:

5 de mayo del 2025

Tabla de contenido

1. Ejercicio 2.1 página 63

Código fuente

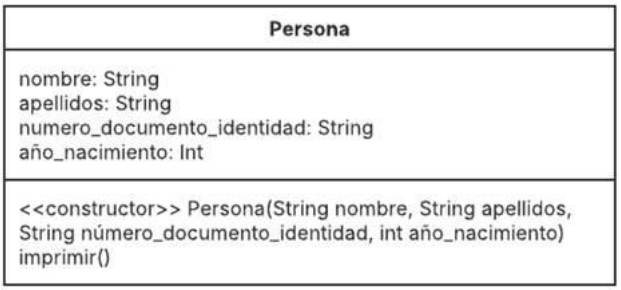
```
class Persona:
    def __init__(self, nombre, apellidos, número_documento_identidad, año_nacimiento):
        self.nombre = nombre
        self.apellidos = apellidos
        self.número_documento_identidad = número_documento_identidad
        self.año_nacimiento = año_nacimiento
```

```
    def imprimir(self):
        print("Nombre =", self.nombre)
        print("Apellidos =", self.apellidos)
        print("Número de documento de identidad =", self.número_documento_identidad)
        print("Año de nacimiento =", str(self.año_nacimiento))
        print()

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

Ejecución:

```
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
```

Diagrama de clases



2. Ejercicio 2.2 página 66

Código fuente

```
class Planeta:

        class TipoPlaneta:

        GASEOSO = "GASEOSO"

        TERRESTRE = "TERRESTRE"

        ENANO = "ENANO"


        def __init__(self, nombre = None, cantidad_satelites = 0, masa = 0.0, volúmen = 0.0, diámetro = 0, distancia_sol = 0, tipo = None, es_observable = False):

        self.nombre = nombre

        self.cantidad_satelites = cantidad_satelites

        self.masa = masa

        self.volúmen = volúmen

        self.diámetro = diámetro

        self.distancia_sol = distancia_sol

        self.tipo = tipo

        self.es_observable = es_observable
```

```
        def imprimir(self):
        print("Nombre del planeta =", self.nombre)
        print("Cantidad de satélites =", str(self.cantidad_satelites))
        print("Masa del planeta =", str(self.masa))
        print("Volúmen del planeta =", str(self.volúmen))
        print("Diámetro del planeta =", str(self.diámetro))
        print("Distancia al sol =", str(self.distancia_sol))
        print("Tipo de planeta =", self.tipo)
        print("Es observable =", self.es_observable)

        def calcular_densidad(self):
        if self.volúmen != 0:
        return self.masa / self.volúmen
        else:
        return 0

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

if __name__ == "__main__":
        p1 = Planeta(nombre="Tierra", cantidad_satelites=1, masa=5.9736e24,
volúmen=1.08321e12, diámetro=12742, distancia_sol=150000000,
tipo=Planeta.TipoPlaneta.TERRESTRE, es_observable=True)
        p1.imprimir()
        print("Densidad del planeta =", str(p1.calcular_densidad()))
```

```
        print("Es planeta exterior =", p1.es_planeta_exterior())

        print()


        p2 = Planeta(nombre="Júpiter", cantidad_satelites=79, masa=1.899e27,
volúmen=1.4313e15, diámetro=139820, distancia_sol=750000000,
tipo=Planeta.TipoPlaneta.GASEOSO, es_observable=True)

        p2.imprimir()

        print("Densidad del planeta =", str(p2.calcular_densidad()))

        print("Es planeta exterior =", p2.es_planeta_exterior())
```

Ejecución:

```
Nombre del planeta = Tierra

Cantidad de satélites = 1

Masa del planeta = 5.9736e+24

Volúmen 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

Volúmen del planeta = 1431300000000000.0

Diámetro del planeta = 139820

Distancia al sol = 750000000
```

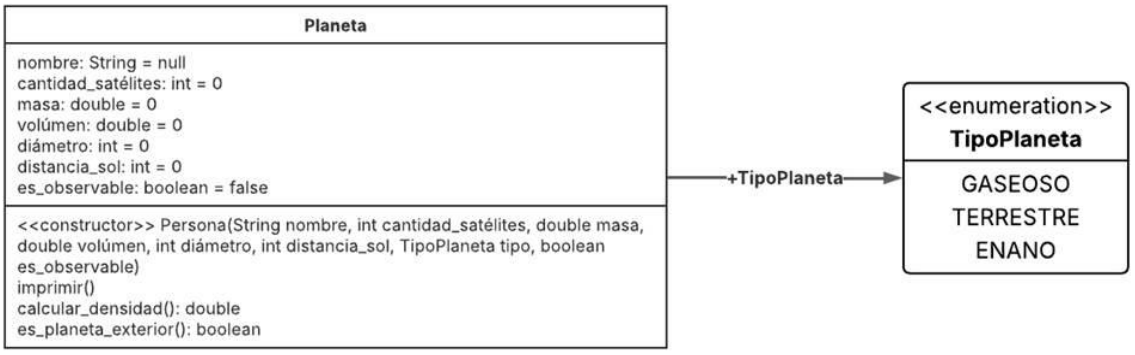
Tipo de planeta = GASEOSO

Es observable = True

Densidad del planeta = 1326765877174.5964

Es planeta exterior = True

Diagrama de clases



3. Ejercicio 2.3 página 73

Código fuente

```
class Automovil:
        class TipoCombustible:
        GASOLINA = "GASOLINA"
        BIOETANOL = "BIOETANOL"
        DIESEL = "DIESEL"
        BIODIESEL = "BIODIESEL"
        GAS_NATURAL = "GAS_NATURAL"

        class TipoAutomovil:
```

```
        CIUDAD = "CIUDAD"
        SUBCOMPACTO = "SUBCOMPACTO"
        COMPACTO = "COMPACTO"
        FAMILIAR = "FAMILIAR"
        EJECUTIVO = "EJECUTIVO"
        SUV = "SUV"

        class TipoColor:
        BLANCO = "BLANCO"
        NEGRO = "NEGRO"
        ROJO = "ROJO"
        NARANJA = "NARANJA"
        AMARILLO = "AMARILLO"
        VERDE = "VERDE"
        AZUL = "AZUL"
        VIOLETA = "VIOLETA"

        def __init__(self, marca, modelo, motor, tipo_combustible, tipo_automóvil,
número_puertas, cantidad_asientos, velocidad_máxima, color):
        self.marca = marca
        self.modelo = modelo
        self.motor = motor
      self.tipo_combustible = tipo_combustible
      self.tipo_automóvil = tipo_automóvil
      self.número_puertas = número_puertas
      self.cantidad_asientos = cantidad_asientos
      self.velocidad_máxima = velocidad_máxima
        self.color = color
      self.velocidad_actual = 0
```

```
def get_marca(self):
return self.marca

def get_modelo(self):
return self.modelo

def get_motor(self):
return self.motor

def get_tipo_combustible(self):
return self.tipo_combustible

def get_tipo_automovil(self):
return self.tipo_automovil

def get_número_puertas(self):
return self.número_puertas

def get_cantidad_asientos(self):
return self.cantidad_asientos

def get_velocidad_máxima(self):
return self.velocidad_máxima

def get_color(self):
return self.color
```

```
    def get_velocidad_actual(self):
    return self.velocidad_actual

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

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

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

    def set_tipo_combustible(self, tipo_combustible):
self.tipo_combustible = tipo_combustible

    def set_tipo_automóvil(self, tipo_automóvil):
self.tipo_automóvil = tipo_automóvil

    def set_número_puertas(self, número_puertas):
self.número_puertas = número_puertas

    def set_cantidad_asientos(self, cantidad_asientos):
self.cantidad_asientos = cantidad_asientos

    def set_velocidad_máxima(self, velocidad_máxima):
self.velocidad_máxima = velocidad_máxima

    def set_color(self, color):
```

```
        self.color = color

        def set_velocidad_actual(self, velocidad_actual):
    self.velocidad_actual = velocidad_actual

        def acelerar(self, incremento_velocidad):
        if self.velocidad_actual + incremento_velocidad < self.velocidad_máxima:
        self.velocidad_actual += incremento_velocidad
        else:
        print("No se puede incrementar a una velocidad superior a la máxima del
automóvil.")

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

        def frenar(self):
    self.velocidad_actual = 0

        def calcular_tiempo_llegada(self, distancia):
        if self.velocidad_actual != 0:
        return distancia / self.velocidad_actual
        else:
        return float('inf')

        def imprimir(self):
    print(f"Marca = {self.marca}")
```

```
        print(f"Modelo = {self.modelo}")
        print(f"Motor = {self.motor}")
        print(f"Tipo de combustible = {self.tipo_combustible}")
        print(f"Tipo de automóvil = {self.tipo_automóvil}")
        print(f"Número de puertas = {self.número_puertas}")
        print(f"Cantidad de asientos = {self.cantidad_asientos}")
        print(f"Velocida máxima = {self.velocidad_máxima}")
        print(f"Color = {self.color}")

if __name__ == "__main__":
            auto1 = Automovil(
            marca = "Ford",
            modelo = 2018,
            motor = 3,
            tipo_combustible = Automovil.TipoCombustible.DIESEL,
            tipo_automóvil = Automovil.TipoAutomovil.EJECUTIVO,
            número_puertas = 5,
            cantidad_asientos = 6,
            velocidad_máxima = 250,
            color = Automovil.TipoColor.NEGRO)

            auto1.imprimir()
    auto1.set_velocidad_actual(100)
    print(f"Velocidad actual = {auto1.get_velocidad_actual()}")
            auto1.acelerar(20)
    print(f"Velocidad actual = {auto1.get_velocidad_actual()}")
    auto1.desacelerar(50)
    print(f"Velocidad actual = {auto1.get_velocidad_actual()}")
```

```
            auto1.frenar()
    print(f"Velocidad actual = {auto1.get_velocidad_actual()}")
    auto1.desacelerar(20)
```

Ejecución:

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

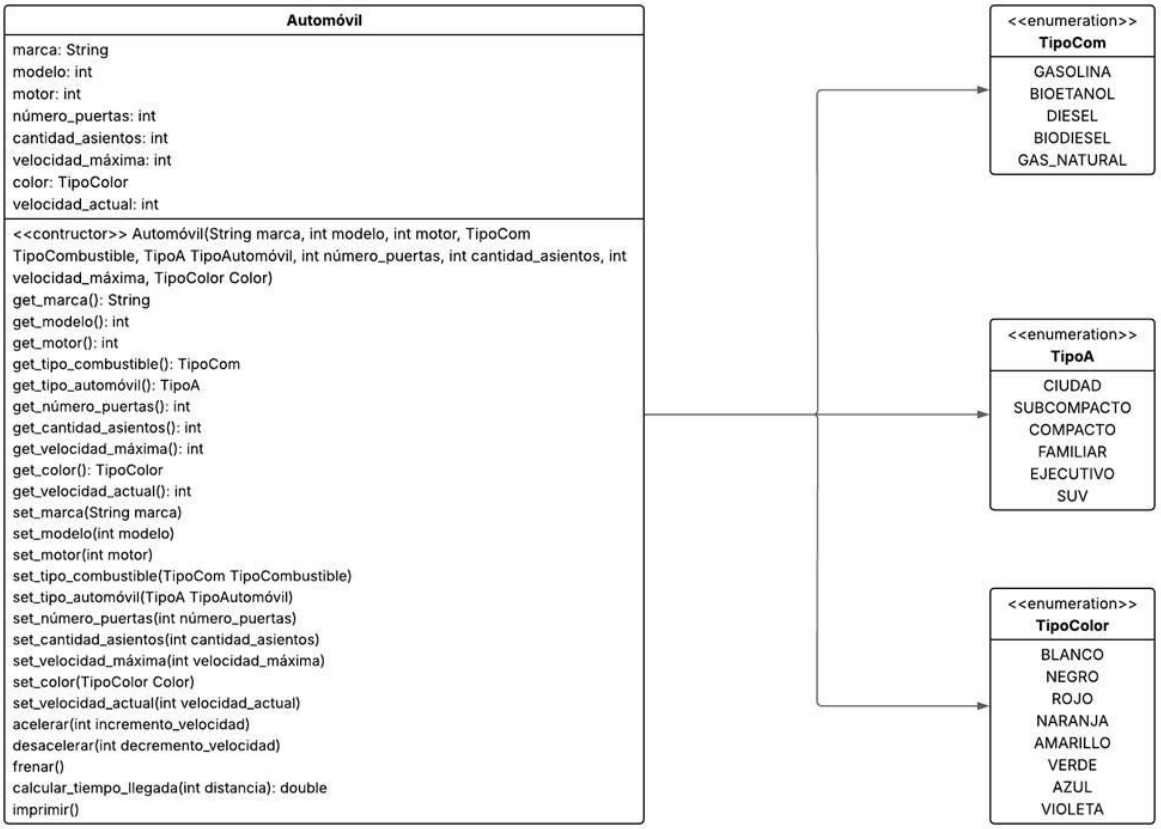
Velocidad actual = 120

Velocidad actual = 70

Velocidad actual = 0

No se puede decrementar a una velocidad negativa.

Diagrama de clases



4. Ejercicio 2.4 página 86

Código fuente

```
import math

# Clase Círculo

class Circulo:

    def __init__(self, radio: float) -> None:

        self.radio = radio

    def calcular_area(self) -> float:

        return math.pi * (self.radio ** 2)

    def calcular_perimetro(self) -> float:
```

```
        return 2 * math.pi * self.radio
# Clase Rectángulo
class Rectangulo:
    def __init__(self, base: float, altura: float) -> None:
        self.base = base
        self.altura = altura
    def calcular_area(self) -> float:
        return self.base * self.altura
    def calcular_perimetro(self) -> float:
        return 2 * (self.base + self.altura)
# Clase Cuadrado
class Cuadrado:
    def __init__(self, lado: float) -> None:
        self.lado = lado
    def calcular_area(self) -> float:
        return self.lado ** 2
    def calcular_perimetro(self) -> float:
        return 4 * self.lado
# Clase Triángulo Rectángulo
class TrianguloRectangulo:
    def __init__(self, base: float, altura: float) -> None:
        self.base = base
        self.altura = altura
    def calcular_area(self) -> float:
        return (self.base * self.altura) / 2
    def calcular_perimetro(self) -> float:
        return self.base + self.altura + self.calcular_hipotenusa()
```

```
    def calcular_hipotenusa(self) -> float:
        return math.sqrt(self.base ** 2 + self.altura ** 2)
    def tipo_triangulo(self) -> str:
        hipotenusa = self.calcular_hipotenusa()
        lados = [round(self.base, 5), round(self.altura, 5), round(hipotenusa, 5)]
        unicos = set(lados)
        if len(unicos) == 1:
            return "Equilátero"
        elif len(unicos) == 2:
            return "Isósceles"
        else:
            return "Escaleno"
def prueba_figuras_geometricas() -> None:
    # Círculo
    radio = float(input("Ingrese el radio del círculo: "))
    circulo = Circulo(radio)
    print(f"Área del círculo: {circulo.calcular_area():.2f}")
    print(f"Perímetro del círculo: {circulo.calcular_perimetro():.2f}")
    # Rectángulo
    base = float(input("\nIngrese la base del rectángulo: "))
    altura = float(input("Ingrese la altura del rectángulo: "))
    rectangulo = Rectangulo(base, altura)
    print(f"Área del rectángulo: {rectangulo.calcular_area():.2f}")
    print(f"Perímetro del rectángulo: {rectangulo.calcular_perimetro():.2f}")
    # Cuadrado
    lado = float(input("\nIngrese el lado del cuadrado: "))
    cuadrado = Cuadrado(lado)
```

```
    print(f"Área del cuadrado: {cuadrado.calcular_area():.2f}")
    print(f"Perímetro del cuadrado: {cuadrado.calcular_perimetro():.2f}")
    # Triángulo Rectángulo
    base = float(input("\nIngrese la base del triángulo rectángulo: "))
    altura = float(input("Ingrese la altura del triángulo rectángulo: "))
    triangulo = TrianguloRectangulo(base, altura)
    print(f"Área del triángulo rectángulo: {triangulo.calcular_area():.2f}")
    print(f"Perímetro del triángulo rectángulo: {triangulo.calcular_perimetro():.2f}")
    print(f"Tipo de triángulo: {triangulo.tipo_triangulo()}")
# Punto de entrada
if __name__ == "__main__":
    prueba_figuras_geometricas()
```

Ejecución:

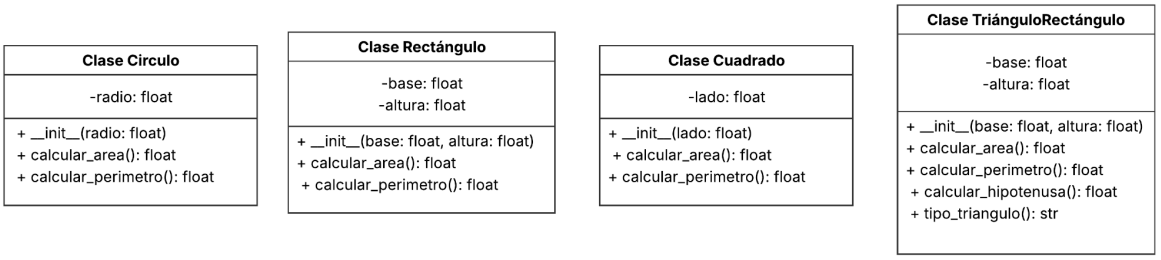
```
Ingrese el radio del círculo: 2
Área del círculo: 12.57
Perímetro del círculo: 12.57
Ingrese la base del rectángulo: 1
Ingrese la altura del rectángulo: 2
Área del rectángulo: 2.00
Perímetro del rectángulo: 6.00
Ingrese el lado del cuadrado: 3
Área del cuadrado: 9.00
Perímetro del cuadrado: 12.00
Ingrese la base del triángulo rectángulo: 3
Ingrese la altura del triángulo rectángulo: 5
```

Área del triángulo rectángulo: 7.50

Perímetro del triángulo rectángulo: 13.83

Tipo de triángulo: Escaleno

Diagrama de clases



5. Ejercicio 2.5 página 95

Código fuente

```
from enum import Enum
class TipoCuenta(Enum):
  AHORROS= "AHORROS"
  CORRIENTE="CORRIENTE"
class CuentaBancaria:
  def __init__(self,nombres_titular,apellidos_titular, numero_cuenta, tipo_cuenta):
    self.nombres_titular = nombres_titular
    self.apellidos_titular = apellidos_titular
    self.numero_cuenta = numero_cuenta
    self.tipo_cuenta = tipo_cuenta
    self.saldo = 0
  def imprimir(self):
```

```
    print(f"Nombres del titular: {self.nombres_titular}")
    print(f"Apellidos del titular: {self.apellidos_titular}")
    print(f"Número de cuenta: {self.numero_cuenta}")
    print(f"Tipo de cuenta: {self.tipo_cuenta.value}")
    print(f"Saldo:{self.saldo}")
  def consultar_saldo(self):
    print(f"El saldo actual es: {self.saldo}")
  def consignar(self,valor):
    if valor>0:
      self.saldo += valor
      print(f"Se ha consignado ${valor} en la cuenta. El nuevo saldo es ${self.saldo}")
      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} en la cuenta. El nuevo saldo es ${self.saldo}")
      return True
    else:
      print("El valor a retirar debe ser menor que el saldo actual.")
      return False
```

Ejecución

```
if __name__ == "__main__":
  cuenta = CuentaBancaria("Pedro", "Pérez", "123456789", TipoCuenta.AHORROS)
  cuenta.imprimir()
```

cuenta.consignar(200000)

cuenta.consignar(300000)

cuenta.retirar(400000)

Diagrama de clases