Actividad 3 grupal POO

Estudiantes

David Esteban Cartagena Mejia
C.C: 1001361568

Docente

Walter Hugo Arboleda Mazo

Asignatura

Programación Orientada a Objetos (POO)

Universidad Nacional de Colombia Sede Medellín
Mayo de 2025

Tabla de contenidos

Listado de Figuras

Punto 1

```python
class Cuenta:
    """
    Clase que modela una cuenta bancaria con atributos básicos como saldo,
    número de consignaciones, número de retiros, tasa anual y comisión mensual.
    """
    def __init__(self, saldo, tasa_anual):
        self.saldo = saldo
        self.numero_consignaciones = 0
        self.numero_retiros = 0
        self.tasa_anual = tasa_anual
        self.comision_mensual = 0

    def consignar(self, cantidad):
        """
        Consigna una cantidad de dinero en la cuenta, actualizando el saldo
        y aumentando el número de consignaciones.
        """
        self.saldo += cantidad
        self.numero_consignaciones += 1

    def retirar(self, cantidad):
```

```
        """
        Retira una cantidad de dinero de la cuenta si el saldo es suficiente.
        """
        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):
        """
        Calcula el interés mensual basado en la tasa anual y lo añade al saldo.
        """
        tasa_mensual = self.tasa_anual / 12
        interes_mensual = self.saldo * tasa_mensual
        self.saldo += interes_mensual

    def extracto_mensual(self):
        """
        Genera el extracto mensual, aplicando la comisión mensual
        y calculando el interés.
        """
        self.saldo -= self.comision_mensual
        self.calcular_interes()


class CuentaAhorros(Cuenta):
    """
    Clase que modela una cuenta de ahorros como subclase de Cuenta.
    Añade un atributo adicional para determinar si la cuenta está activa.
    """
    def __init__(self, saldo, tasa_anual):
        super().__init__(saldo, tasa_anual)
        self.activa = saldo >= 10000

    def retirar(self, cantidad):
        """
        Retira una cantidad de dinero si la cuenta está activa.
        """
        if self.activa:
            super().retirar(cantidad)
```

```
            if self.saldo < 10000:
                self.activa = False
        else:
            print("La cuenta está inactiva. No se puede retirar.")

    def consignar(self, cantidad):
        """
        Consigna una cantidad de dinero si la cuenta está activa.
        """
        if self.activa:
            super().consignar(cantidad)
        else:
            print("La cuenta está inactiva. No se puede consignar.")

    def extracto_mensual(self):
        """
        Genera el extracto mensual, aplicando comisiones adicionales
        si hay más de 4 retiros.
        """
        if self.numero_retiros > 4:
            self.comision_mensual += (self.numero_retiros - 4) * 1000
        super().extracto_mensual()
        if self.saldo < 10000:
            self.activa = False

    def imprimir(self):
        """
        Muestra el estado actual de la cuenta.
        """
        print(f"Saldo: ${self.saldo:.2f}")
        print(f"Comisión mensual: ${self.comision_mensual:.2f}")
        print(f"Número de transacciones: {self.numero_consignaciones + self.numero_retiros}")
        print(f"Cuenta activa: {self.activa}")


class CuentaCorriente(Cuenta):
    """
    Clase que modela una cuenta corriente como subclase de Cuenta.
    Añade un atributo adicional para manejar sobregiros.
    """
    def __init__(self, saldo, tasa_anual):
        super().__init__(saldo, tasa_anual)
```

```
        self.sobregiro = 0

    def retirar(self, cantidad):
        """
        Permite realizar un retiro incluso si el saldo no es suficiente,
        generando un sobregiro.
        """
        resultado = self.saldo - cantidad
        if resultado < 0:
            self.sobregiro -= resultado
            self.saldo = 0
        else:
            super().retirar(cantidad)

    def consignar(self, cantidad):
        """
        Permite consignar dinero, cubriendo primero el sobregiro si lo hay.
        """
        if self.sobregiro > 0:
            residuo = self.sobregiro - cantidad
            if residuo > 0:
                self.sobregiro = residuo
            else:
                self.sobregiro = 0
                self.saldo += -residuo
        else:
            super().consignar(cantidad)

    def imprimir(self):
        """
        Muestra el estado actual de la cuenta.
        """
        print(f"Saldo: ${self.saldo:.2f}")
        print(f"Comisión mensual: ${self.comision_mensual:.2f}")
        print(f"Número de transacciones: {self.numero_consignaciones + self.numero_retiros}
        print(f"Sobregiro: ${self.sobregiro:.2f}")


# Prueba del programa
if __name__ == "__main__":
    print("Cuenta de Ahorros:")
    saldo_inicial = float(input("Ingrese el saldo inicial: "))
```

```
    tasa_anual = float(input("Ingrese la tasa anual (%): "))
    cuenta_ahorros = CuentaAhorros(saldo_inicial, tasa_anual)

    cantidad_consignar = float(input("Ingrese la cantidad a consignar: "))
    cuenta_ahorros.consignar(cantidad_consignar)

    cantidad_retirar = float(input("Ingrese la cantidad a retirar: "))
    cuenta_ahorros.retirar(cantidad_retirar)

    cuenta_ahorros.extracto_mensual()
    cuenta_ahorros.imprimir()
```

Punto 2

```
class Inmueble:
    """
    Clase base que modela un inmueble con identificador, área, dirección y precio de venta
    """
    def __init__(self, identificador_inmobiliario, area, direccion):
        self.identificador_inmobiliario = identificador_inmobiliario
        self.area = area
        self.direccion = direccion
        self.precio_venta = 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"Área: {self.area} m²")
        print(f"Dirección: {self.direccion}")
        print(f"Precio de venta: ${self.precio_venta}")


class InmuebleVivienda(Inmueble):
    """
    Subclase de Inmueble que representa una vivienda con número de habitaciones y baños.
    """
```

```
    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):
    """
    Subclase de InmuebleVivienda que representa una casa con número de pisos.
    """
    def __init__(self, identificador_inmobiliario,
    area, direccion, numero_habitaciones, numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,
        self.numero_pisos = numero_pisos

    def imprimir(self):
        super().imprimir()
        print(f"Número de pisos: {self.numero_pisos}")


class Apartamento(InmuebleVivienda):
    """
    Subclase de InmuebleVivienda que representa un apartamento.
    """
    def __init__(self, identificador_inmobiliario,
     area, direccion, numero_habitaciones, numero_banos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,


class Apartaestudio(Apartamento):
    """
    Subclase de Apartamento que representa un apartaestudio.
    """
    VALOR_AREA = 1500000
```

```
    def __init__(self, identificador_inmobiliario,
     area, direccion):
        super().__init__(identificador_inmobiliario, area, direccion, 1, 1)


class CasaRural(Casa):
    """
    Subclase de Casa que representa una casa rural con atributos adicionales de distancia a
    municipal y altitud sobre el nivel del mar.
    """
    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,
        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")


class CasaUrbana(Casa):
    """
    Subclase de Casa que representa una casa urbana.
    """
    def __init__(self, identificador_inmobiliario,
     area, direccion, numero_habitaciones,
     numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,


class CasaConjuntoCerrado(CasaUrbana):
    """
    Subclase de CasaUrbana que representa una casa en un conjunto cerrado.
    """
    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,
        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 administración: ${self.valor_administracion}")
        print(f"Tiene piscina: {'Sí' if self.tiene_piscina else 'No'}")
        print(f"Tiene campos deportivos: {'Sí' if self.tiene_campos_deportivos else 'No'}"


class CasaIndependiente(CasaUrbana):
    """
    Subclase de CasaUrbana que representa una casa independiente.
    """
    VALOR_AREA = 3000000

    def __init__(self, identificador_inmobiliario,
    area, direccion, numero_habitaciones,
    numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,


class Local(Inmueble):
    """
    Subclase de Inmueble que representa un local.
    """
    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}")
```

```python
class LocalComercial(Local):
    """
    Subclase de Local que representa un local comercial.
    """
    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}")


class Oficina(Local):
    """
    Subclase de Local que representa una oficina.
    """
    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: {'Sí' if self.es_gobierno else 'No'}")


# Ejemplo de uso
if __name__ == "__main__":
    print("\nDatos Apartamento")
    apartamento_familiar = Apartamento(103067, 120, "Avenida Santander 45-45", 3, 2)
    apartamento_familiar.calcular_precio_venta(2000000)
    apartamento_familiar.imprimir()

    print("\nDatos Apartaestudio")
    apartaestudio = Apartaestudio(123454, 50, "Avenida Caracas 30-15")
    apartaestudio.calcular_precio_venta(Apartaestudio.VALOR_AREA)
    apartaestudio.imprimir()
```

```
    print("\nDatos Casa Conjunto Cerrado")
    casa_conjunto = CasaConjuntoCerrado(56789, 200, "Calle Principal 123", 4, 3, 2, 150000
    casa_conjunto.calcular_precio_venta(CasaConjuntoCerrado.VALOR_AREA)
    casa_conjunto.imprimir()
```

Punto 2

```
class Inmueble:
    """
    Clase base que modela un inmueble con identificador, área, dirección y precio de venta
    """
    def __init__(self, identificador_inmobiliario,
     area, direccion):
        self.identificador_inmobiliario = identificador_inmobiliario
        self.area = area
        self.direccion = direccion
        self.precio_venta = 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"Área: {self.area} m²")
        print(f"Dirección: {self.direccion}")
        print(f"Precio de venta: ${self.precio_venta}")


class InmuebleVivienda(Inmueble):
    """
    Subclase de Inmueble que representa una vivienda con número de habitaciones y baños.
    """
    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):
    """
    Subclase de InmuebleVivienda que representa una casa con número de pisos.
    """
    def __init__(self, identificador_inmobiliario,
     area, direccion, numero_habitaciones,
     numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,
        self.numero_pisos = numero_pisos

    def imprimir(self):
        super().imprimir()
        print(f"Número de pisos: {self.numero_pisos}")


class Apartamento(InmuebleVivienda):
    """
    Subclase de InmuebleVivienda que representa un apartamento.
    """
    def __init__(self, identificador_inmobiliario,
    area, direccion, numero_habitaciones, numero_banos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,


class Apartaestudio(Apartamento):
    """
    Subclase de Apartamento que representa un apartaestudio.
    """
    VALOR_AREA = 1500000

    def __init__(self, identificador_inmobiliario,
    area, direccion):
        super().__init__(identificador_inmobiliario, area, direccion, 1, 1)


class CasaRural(Casa):
```

```
    """
    Subclase de Casa que representa una casa rural con atributos adicionales de distancia a
    municipal y altitud sobre el nivel del mar.
    """
    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,
        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")


class CasaUrbana(Casa):
    """
    Subclase de Casa que representa una casa urbana.
    """
    def __init__(self, identificador_inmobiliario,
    area, direccion, numero_habitaciones,
    numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,


class CasaConjuntoCerrado(CasaUrbana):
    """
    Subclase de CasaUrbana que representa una casa en un conjunto cerrado.
    """
    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,
        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 administración: ${self.valor_administracion}")
        print(f"Tiene piscina: {'Sí' if self.tiene_piscina else 'No'}")
        print(f"Tiene campos deportivos: {'Sí' if self.tiene_campos_deportivos else 'No'}"


class CasaIndependiente(CasaUrbana):
    """
    Subclase de CasaUrbana que representa una casa independiente.
    """
    VALOR_AREA = 3000000

    def __init__(self, identificador_inmobiliario,
    area, direccion, numero_habitaciones,
    numero_banos, numero_pisos):
        super().__init__(identificador_inmobiliario, area, direccion, numero_habitaciones,


class Local(Inmueble):
    """
    Subclase de Inmueble que representa un local.
    """
    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}")


class LocalComercial(Local):
    """
    Subclase de Local que representa un local comercial.
    """
    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}")


class Oficina(Local):
    """
    Subclase de Local que representa una oficina.
    """
    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: {'Sí' if self.es_gobierno else 'No'}")


# Ejemplo de uso
if __name__ == "__main__":
    print("\nDatos Apartamento")
    apartamento_familiar = Apartamento(103067, 120, "Avenida Santander 45-45", 3, 2)
    apartamento_familiar.calcular_precio_venta(2000000)
    apartamento_familiar.imprimir()

    print("\nDatos Apartaestudio")
    apartaestudio = Apartaestudio(123454, 50, "Avenida Caracas 30-15")
    apartaestudio.calcular_precio_venta(Apartaestudio.VALOR_AREA)
    apartaestudio.imprimir()

    print("\nDatos Casa Conjunto Cerrado")
    casa_conjunto = CasaConjuntoCerrado(56789, 200, "Calle Principal 123", 4, 3, 2, 150000
    casa_conjunto.calcular_precio_venta(CasaConjuntoCerrado.VALOR_AREA)
    casa_conjunto.imprimir()
```

Punto 3

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

# -------------------------
# PERROS
# -------------------------
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")

# Tamaños
class PerroPequeño(Perro): pass
class PerroMediano(Perro): pass
class PerroGrande(Perro): 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
```

```python
class PastorAleman(PerroGrande): pass
class Doberman(PerroGrande): pass
class Rottweiler(PerroGrande): pass

# ------------------------
# GATOS
# ------------------------
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")

# Tipos por pelaje
class GatoSinPelo(Gato): pass
class GatoPeloLargo(Gato): pass
class GatoPeloCorto(Gato): 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

# ------------------------
# EJEMPLO
# ------------------------
```

```python
# Crear instancias
perro = Bulldog("Rex", 4, "Marrón", 25.0, True)
gato = AzulRuso("Michi", 2, "Gris", 1.0, 2.5)

# Mostrar sonidos
Bulldog.sonido()    # Los perros ladran
AzulRuso.sonido()   # Los gatos maúllan y ronronean

# Acceder a atributos
print(f"{perro.nombre}, {perro.edad} años, color {perro.color}, muerde: {perro.muerde}")
print(f"{gato.nombre}, salta {gato.altura_salto}m de alto y mide {gato.longitud_salto}m de
```

Punto 4

```python
# Clase base
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


# Subclase Estudiante
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


# Subclase Profesor
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


# Ejemplo

# Crear un estudiante
est = Estudiante("Ana García", "Calle 123", "Ingeniería", 3)

# Acceder a sus datos
print("Estudiante:")
print("Nombre:", est.getNombre())
print("Dirección:", est.getDireccion())
```

```
print("Carrera:", est.getCarrera())
print("Semestre:", est.getSemestre())

# Modificar semestre
est.setSemestre(4)
print("Semestre actualizado:", est.getSemestre())

print()

# Crear un profesor
prof = Profesor("Dr. Luis Pérez", "Av. Principal 456", "Matemáticas", "Titular")

# Acceder a sus datos
print("Profesor:")
print("Nombre:", prof.getNombre())
print("Dirección:", prof.getDireccion())
print("Departamento:", prof.getDepartamento())
print("Categoría:", prof.getCategoria())

# Cambiar categoría
prof.setCategoria("Asociado")
print("Categoría actualizada:", prof.getCategoria())
```

Punto 5

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

class ProfesorTitular(Profesor):
    """
    Subclase ProfesorTitular que hereda de Profesor.
    """
    def imprimir(self):
        print("Es un profesor titular.")
```

```
def main():
    """
    Función principal que demuestra el uso de polimorfismo.
    """
    profesor1 = ProfesorTitular()  # Se declara como Profesor pero se instancia como Profes
    profesor1.imprimir()           # Salida: "Es un profesor titular."

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