Nombre: Miguel angel Mendoza Espinoza

Practica de auxiliatura Herencia:

Ejercicio1:Diagrama de clases:

+-------------------+

| Vehiculo |

+-------------------+

| - marca |

| - modelo |

| - anio |

| - precio\_base |

+-------------------+

| + mostrar\_info() |

| + get\_anio() |

+-------------------+

▲

+-------+--------+----------------+

| |

+-------------+ +-------------+

| Coche | | Moto |

+-------------+ +-------------+

| - num\_puertas | | - cilindrada |

| - tipo\_combustible | | - tipo\_motor |

+-------------+ +-------------+

| + mostrar\_info()| | + mostrar\_info() |

| + get\_num\_puertas() | | |

+-------------+ +-------------+

Seudocódigo:

Clase Vehiculo:

Constructor (marca, modelo, anio, precio\_base)Asignar valores a atributos

Método mostrar\_info():

Imprimir información del vehículo

Método get\_anio():

Retornar el año del vehículo

Clase Coche hereda de Vehiculo:

Constructor (marca, modelo, anio, precio\_base, num\_puertas, tipo\_combustible)

Llamar al constructor de Vehiculo

Asignar valores a num\_puertas y tipo\_combustible

Método mostrar\_info():Llamar a mostrar\_info() de Vehiculo

Imprimir número de puertas y tipo de combustible

Método get\_num\_puertas():

Retornar número de puertas

Clase Moto hereda de Vehiculo:

Constructor (marca, modelo, anio, precio\_base, cilindrada, tipo\_motor)

Llamar al constructor de Vehiculo

Asignar valores a cilindrada y tipo\_motor

Método mostrar\_info():Llamar a mostrar\_info() de Vehiculo

Imprimir cilindrada y tipo de motor

Crear lista de vehículos con instancias de Coche y Moto

Imprimir "--- Información de vehículos ---"

Para cada vehículo en la lista:

Llamar mostrar\_info()

Imprimir "--- Coches con más de 4 puertas ---"

Para cada vehículo en la lista:

Si es instancia de Coche y tiene más de 4 puertas:

Llamar mostrar\_info()

Imprimir "--- Vehículos del año 2025 ---"

Para cada vehículo en la lista:

Si el año es 2025:

Llamar mostrar\_info()

Ejercicio3:

+--------------------------+

| Persona |

+--------------------------+

| - ci |

| - nombre |

| - apellido |

| - celular |

| - fecha\_Nac |

+--------------------------+

| + edad() |

| + mostrar() |

+--------------------------+

▲

+------+------+----------+

| |

+----------------+ +----------------+

| Estudiante | | Docente |

+----------------+ +----------------+

| - ru | | - nit |

| - fecha\_Ingreso| | - profesion |

| - semestre | | - especialidad |

| | | - sexo |

+----------------+ +----------------+

| + mostrar() | | + mostrar() |

+----------------+ +----------------+

Seudocodigo:

Clase Persona:

Constructor(ci, nombre, apellido, celular, fecha\_Nac)

Convertir fecha\_Nac a tipo date

Asignar atributos

Método edad():

Calcular la diferencia de años con la fecha actual

Ajustar si aún no ha cumplido años este año

Método mostrar():

Imprimir nombre completo, CI, celular, fecha de nacimiento y edad

Clase Estudiante hereda de Persona:

Constructor(ci, nombre, apellido, celular, fecha\_Nac, ru, fecha\_Ingreso, semestre)

Llamar constructor de Persona

Convertir fecha\_Ingreso a tipo date

Asignar ru y semestre

Método mostrar():

Llamar mostrar() de Persona

Imprimir RU, fecha de ingreso y semestre

Clase Docente hereda de Persona:

Constructor(ci, nombre, apellido, celular, fecha\_Nac, nit, profesion, especialidad, sexo)

Llamar constructor de Persona

Asignar nit, profesion, especialidad y sexo

Método mostrar():

Llamar mostrar() de Persona

Imprimir NIT, profesión, especialidad y sexo

Crear lista de estudiantes

Crear lista de docentes

Imprimir "--- Estudiantes mayores de 25 años ---"

Para cada estudiante:

Si su edad es mayor a 25:

Llamar mostrar()

Imprimir "--- Docente masculino ingeniero de mayor edad ---"

Filtrar docentes de sexo masculino cuya profesión sea "Ingeniero"

Si la lista no está vacía:

Obtener el docente con mayor edad usando max()

Llamar mostrar()

Imprimir "--- Estudiantes y Docentes con el mismo apellido ---"

Para cada estudiante:

Para cada docente:

Si los apellidos coinciden:Imprimir apellido

Llamar mostrar() del estudiante

Llamar mostrar() del docente

Ejercicio5:

+-------------------------------+

| Empleado |

+-------------------------------+

| - nombre : str |

| - apellido : str |

| - salario\_base : float |

| - anios\_antiguedad : int |

+-------------------------------+

| + get\_nombre\_completo() : str|

| + calcular\_salario() : float |

| + mostrar() : void |

+-------------------------------+

▲

+--------+--------+--------------+

| |

+-------------------+ +----------------------------+

| Gerente | | Desarrollador |

+-------------------+ +----------------------------+

| - departamento : str| | - lenguaje\_programacion : str |

| - bono\_gerencial : float| | - horas\_extras : int |

+-------------------+ +----------------------------+

| + calcular\_salario() | | + calcular\_salario() |

| + mostrar() | | + mostrar() |

+-------------------+ +----------------------------+

Agregación y composiciuon: ejercicio1:

+------------------------+

| Habitacion |

+------------------------+

| - nombre : str |

| - tamano : float |

+------------------------+

| + get\_nombre() |

| + get\_tamano() |

| + set\_nombre(nombre) |

| + set\_tamano(tamano) |

| + mostrar\_info() |

+------------------------+

+------------------------+

| Casa |

+------------------------+

| - direccion : str |

| - habitaciones : list |

+------------------------+

| + get\_direccion() |

| + set\_direccion(d) |

| + agregar\_habitacion() |

| + mostrar\_casa() |

+------------------------+

Seudocódigo:

Clase Habitacion:

- Constructor(nombre, tamano)

- Asigna nombre y tamaño

- Método get\_nombre()

- Retorna el nombre

- Método get\_tamano()

- Retorna el tamaño

- Método set\_nombre(nombre)

- Asigna un nuevo nombre

- Método set\_tamano(tamano)

- Asigna un nuevo tamaño

- Método mostrar\_info()

- Imprime: "Habitación: {nombre}, Tamaño: {tamano} m²"

Clase Casa:

- Constructor(direccion)

- Asigna dirección

- Inicializa lista vacía de habitaciones

- Método get\_direccion()

- Retorna dirección

- Método set\_direccion(direccion)

- Asigna nueva dirección

- Método agregar\_habitacion(habitacion)

- Agrega una habitación a la lista

- Método mostrar\_casa()

- Imprime dirección

- Imprime la información de cada habitación usando mostrar\_info()

Programa principal:

- Crear objeto habitacion1 = Habitacion("Dormitorio", 12)

- Crear objeto habitacion2 = Habitacion("Sala", 20)

- Crear objeto habitacion3 = Habitacion("Cocina", 10)

- Crear objeto casa = Casa("Calle Falsa 123")

- Agregar habitaciones a la casa:

- casa.agregar\_habitacion(habitacion1)

- casa.agregar\_habitacion(habitacion2)

- casa.agregar\_habitacion(habitacion3)

- Mostrar toda la información de la casa:

- casa.mostrar\_casa()

Ejercicio3:

+-----------------------+

| Parte |

+-----------------------+

| - nombre : str |

| - peso : float |

+-----------------------+

| + get\_nombre() |

| + get\_peso() |

| + set\_nombre(nombre) |

| + set\_peso(peso) |

| + mostrar\_info() |

+-----------------------+

+----------------------------+

| Avion |

+----------------------------+

| - modelo : str |

| - fabricante : str |

| - partes : list |

+----------------------------+

| + get\_modelo() |

| + get\_fabricante() |

| + set\_modelo(modelo) |

| + set\_fabricante(fab) |

| + agregar\_parte(parte) |

| + mostrar\_avion() |

+----------------------------+

Seudocódigo:

Clase Parte:

- Constructor(nombre, peso)

- Asignar nombre y peso

- get\_nombre()

- Retornar nombre

- get\_peso()

- Retornar peso

- set\_nombre(nombre)

- Asignar nuevo nombre

- set\_peso(peso)

- Asignar nuevo peso

- mostrar\_info()

- Imprimir "Parte: {nombre}, Peso: {peso} kg"

Clase Avion:

- Constructor(modelo, fabricante)

- Asignar modelo y fabricante

- Inicializar lista de partes vacía

- get\_modelo()

- Retornar modelo

- get\_fabricante()

- Retornar fabricante

- set\_modelo(modelo)

- Asignar nuevo modelo

- set\_fabricante(fabricante)

- Asignar nuevo fabricante

- agregar\_parte(parte)

- Añadir objeto Parte a la lista de partes

- mostrar\_avion()

- Imprimir modelo y fabricante

- Para cada parte en la lista:

- Llamar parte.mostrar\_info()

Programa principal:

- Crear motor = Parte("Motor", 1200)

- Crear alas = Parte("Alas", 800)

- Crear tren = Parte("Tren de aterrizaje", 600)

- Crear avion = Avion("Boeing 747", "Boeing")

- Agregar partes:

- avion.agregar\_parte(motor)

- avion.agregar\_parte(alas)

- avion.agregar\_parte(tren)

- Mostrar datos del avión:

- avion.mostrar\_avion()

Ejercicio5:  
+-----------------------------+

| Jugador |

+-----------------------------+

| - nombre : str |

| - numero : int |

| - posicion : str |

+-----------------------------+

| + mostrar\_info() |

+-----------------------------+

▲ ▲ ▲ ▲

| | | |

| | | |

+--------+ +-------------+ +---------------+ +-------------+

| Portero| | Defensa | | Mediocampista | | Delantero |

+--------+ +-------------+ +---------------+ +-------------+

| - habilidad\_especial : str |

+-------------------------------------------------------+

| + mostrar\_info() |

+-------------------------------------------------------+

+-----------------------------+

| Equipo |

+-----------------------------+

| - nombre : str |

| - jugadores : list |

+-----------------------------+

| + agregar\_jugador(jugador) |

| + mostrar\_equipo() |

+-----------------------------+

Seudocódigo:

Clase Jugador:

- Constructor(nombre, numero, posicion)

- Asignar atributos

- mostrar\_info()

- Imprimir nombre, número, posición

Subclases: Portero, Defensa, Mediocampista, Delantero

- Constructor(nombre, numero, habilidad\_especial)

- Llamar al constructor de Jugador con la posición correspondiente

- Asignar habilidad\_especial

- mostrar\_info()

- Llamar mostrar\_info() de Jugador

- Imprimir habilidad\_especial

Clase Equipo:

- Constructor(nombre)

- Asignar nombre

- Inicializar lista de jugadores vacía

- agregar\_jugador(jugador)

- Agregar jugador a la lista

- mostrar\_equipo()

- Imprimir nombre del equipo

- Para cada jugador:

- Llamar jugador.mostrar\_info()

Programa Principal:

- Crear un objeto Equipo con nombre "Los Invencibles"

- Agregar:

- Portero("Carlos", 1, "Atajadas")

- Defensa("Luis", 4, "Marcaje")

- Mediocampista("Pedro", 8, "Pases precisos")

- Delantero("Juan", 9, "Goleador nato")

- Llamar mostrar\_equipo()

Ejercico7:

+------------------------------+

| Estudiante |

+------------------------------+

| - nombre : String |

| - carrera : String |

| - semestre : int |

+------------------------------+

| + getNombre() : String |

| + getCarrera() : String |

| + getSemestre() : int |

| + setNombre(nombre : String)|

| + setCarrera(carrera : Str) |

| + setSemestre(semestre : int)|

| + mostrarInfo() |

+------------------------------+

+------------------------------+

| Universidad |

+------------------------------+

| - nombre : String |

| - estudiantes : ArrayList |

+------------------------------+

| + getNombre() : String |

| + setNombre(nombre : String)|

| + agregarEstudiante(est) |

| + mostrarUniversidad() |

+------------------------------+

Seudocódigo:

Clase Estudiante:

- Atributos: nombre, carrera, semestre

- Constructor(nombre, carrera, semestre)

- Métodos:

- getNombre()

- getCarrera()

- getSemestre()

- setNombre(nombre)

- setCarrera(carrera)

- setSemestre(semestre)

- mostrarInfo():

Imprimir nombre, carrera y semestre

Clase Universidad:

- Atributos: nombre, lista de estudiantes

- Constructor(nombre)

- Inicializar nombre

- Inicializar lista vacía de estudiantes

- Métodos:

- getNombre()

- setNombre(nombre)

- agregarEstudiante(estudiante)

- mostrarUniversidad():

Imprimir nombre

Para cada estudiante en la lista:

llamar estudiante.mostrarInfo()

Programa principal:

- Crear tres objetos Estudiante con diferentes datos

- Crear un objeto Universidad con nombre "Universidad Nacional"

- Agregar los estudiantes a la universidad usando agregarEstudiante()

- Llamar a mostrarUniversidad()

Ejercicio9:

+----------------------------+

| Producto |

+----------------------------+

| - nombre : String |

| - precio : double |

+----------------------------+

| + getNombre() : String |

| + getPrecio() : double |

| + setNombre(nombre:String) |

| + setPrecio(precio:double) |

| + mostrarInfo() |

+----------------------------+

+------------------------------+

| CarritoCompras |

+------------------------------+

| - productos : ArrayList |

| - LIMITE : int (constante 10)|

+------------------------------+

| + agregarProducto(producto) |

| + mostrarCarrito() |

+------------------------------+

Seudocódigo:

Clase Producto:

- Atributos: nombre, precio

- Constructor(nombre, precio)

- Métodos:

- getNombre()

- getPrecio()

- setNombre(nombre)

- setPrecio(precio)

- mostrarInfo():

Imprimir nombre y precio del producto

Clase CarritoCompras:

- Atributos: productos (lista), LIMITE = 10

- Constructor:

- Inicializa la lista de productos vacía

- Métodos:

- agregarProducto(producto):

Si hay menos de 10 productos en la lista:

Agregar el producto

Imprimir que se ha agregado

Si no:

Imprimir que se alcanzó el límite

- mostrarCarrito():

Imprimir encabezado "Carrito de Compras"

Si la lista está vacía:

Imprimir que el carrito está vacío

Sino:

Para cada producto en la lista:

Mostrar su información

Programa principal:

- Crear 11 productos diferentes con nombre y precio

- Crear un objeto CarritoCompras

- Agregar los 11 productos uno por uno al carrito

- Mostrar el contenido del carrito con mostrarCarrito()