Línea horizontal

**ITU**

**Profesor: Lic. Ignacio G Fontaine Digier**

**Email: i.fontaine.digier@gmail.com**

Ejercicios POO - Java

28 de agosto de 2024

# Ejercicio 1: Crear una clase "Persona"

## Descripción:

Crea una clase llamada **Persona** que tenga los siguientes atributos: **nombre** (String), **edad** (int) y **género** (String). La clase debe tener un constructor que inicialice todos los atributos. Además, incluye los métodos **mostrarInformacion()** para mostrar en consola los detalles de la persona y **esMayorDeEdad()** que retorne true si la persona tiene 18 años o más, y false en caso contrario.

## Pasos:

1. Define la clase **Persona** con sus atributos privados.
2. Implementa un constructor que acepte parámetros para inicializar los atributos.
3. Crea los métodos **mostrarInformacion()** y **esMayorDeEdad()**.
4. En la clase principal, crea una instancia de Persona y llama a los métodos para verificar su funcionamiento.

# Ejercicio 2: Crear una clase "Coche"

## **Descripción:**

Crea una clase llamada **Coche** que tenga los atributos **marca** (String), **modelo** (String), **año** (int) y **velocidad** (int). La clase debe permitir **encender** el coche (aumentar la velocidad), **frenar** (disminuir la velocidad) y **mostrar** el estado del coche.

## Pasos:

1. Define la clase **Coche** con sus atributos privados.
2. Implementa un constructor que inicialice los atributos, exceptuando velocidad, que debe inicializarse en 0.
3. Crea los métodos **acelerar()**, **frenar()** y **mostrarEstado()**.
4. En la clase principal, crea una instancia de Coche y simula un pequeño recorrido.

# Ejercicio 3: Crear una clase "CuentaBancaria"

## **Descripción:**

Crea una clase llamada **CuentaBancaria** que tenga los atributos **numeroCuenta** (String), **titular** (String) y **saldo** (double). La clase debe permitir depositar dinero, retirar dinero (solo si hay saldo suficiente) y consultar el saldo.

## Pasos:

1. Define la clase **CuentaBancaria** con sus atributos privados.
2. Implementa un constructor que inicialice los atributos.
3. Crea los métodos **depositar(double cantidad)**, **retirar(double cantidad)** y **consultarSaldo()**.
4. En la clase principal, crea una instancia de CuentaBancaria, realiza algunas operaciones y muestra el saldo final.

# Prueba Técnica

# Ejercicio 4: Tienda de Tecnología

## Contexto:

En una tienda de tecnología, se gestionan varios productos que tienen un código único, un nombre y un precio. La tienda quiere implementar un sistema que le permita gestionar los productos, incluyendo la posibilidad de agregar productos, mostrar la lista de productos y calcular el total del valor del inventario.

## Problema:

Debes desarrollar un programa en Java utilizando Programación Orientada a Objetos que cumpla con los siguientes requerimientos:

1. Crear una clase llamada **Producto** con los atributos:
   * codigo (String)
   * nombre (String)
   * precio (double)
2. Crear una clase llamada **Tienda** que contenga:
   * Una lista de productos (puede usar ArrayList<Producto>).
   * Un método para agregar un producto a la lista.
   * Un método para mostrar todos los productos.
   * Un método para calcular y devolver el valor total de todos los productos en la tienda.
3. Crear una clase **principal** donde:
   * Se agregan al menos 3 productos a la tienda.
   * Se muestra la lista de productos.
   * Se muestra el valor total del inventario.

## Requerimientos:

* Usa los conceptos de clases, objetos, encapsulamiento y métodos.
* Usa un array o lista para almacenar los productos.

# Ejercicio 5: Sala de reuniones

## Contexto:

Una empresa quiere implementar un sistema simple para gestionar la reserva de sus salas de reunión. Cada sala tiene un identificador único y una capacidad máxima. El sistema debe permitir registrar salas, reservarlas si están disponibles, y liberar la reserva una vez que la reunión haya terminado.

## Descripción del problema:

1. Crear una clase **Sala** con los atributos: ***número de sala*** y ***capacidad***.
2. Crear una clase **Reserva** con los atributos: ***número de sala*** y un ***estado*** que indique si está reservada o no.
3. Implementar métodos en Reserva para reservar una sala **reservarSala()** y liberar una sala ***liberarSala()***.

## Explicación POO:

* Clases: Sala y Reserva.
* Relación: Reserva tiene una relación de composición con Sala, ya que cada reserva está vinculada a una sala.
* Métodos:
  + reservarSala() marca la sala como reservada si está disponible.
  + liberarSala() cambia el estado de la sala a disponible una vez que se libera.

### Ejercicio 6: Gestión de Recursos Humanos en una Empresa

## Consigna:

En una empresa, se desea gestionar a sus empleados, departamentos y los roles que desempeñan. Se te solicita diseñar un sistema que permita realizar estas tareas. Para ello, debes crear las siguientes clases:

### Clase Empleado:

#### Atributos:

* + - **nombre**: Nombre completo del empleado.
    - **id**: Identificador único del empleado.
    - **sueldoBase**: Sueldo base mensual del empleado.
    - **rol**: Objeto de tipo Rol, que representa el rol que desempeña el empleado.
    - **departamento**: Objeto de tipo Departamento, que representa el departamento donde trabaja el empleado.

#### Métodos:

* + - **calcularBono()**: Calcula un bono del 10% del sueldo base.
    - **mostrarInformacion()**: Muestra la información detallada del empleado (nombre, ID, sueldo base, rol y departamento).

### Clase Departamento:

#### Atributos:

* + - **nombre**: Nombre del departamento.
    - **id**: Identificador único del departamento.
    - **jefe**: Objeto de tipo Empleado, que es el jefe del departamento.

#### Métodos:

* + - **cambiarJefe(Empleado nuevoJefe)**: Cambia al jefe del departamento.
    - **mostrarDepartamento()**: Muestra la información del departamento (nombre, ID, nombre del jefe).

### Clase Rol:

#### Atributos:

* + - **nombre**: Nombre del rol (ej. "Desarrollador", "Gerente").
    - **nivel**: Nivel jerárquico del rol (ej. 1 para empleados junior, 2 para senior, etc.).

#### Métodos:

* + - **mostrarRol()**: Muestra la información del rol (nombre y nivel).

## Requisitos:

1. Implementa los constructores en cada clase para inicializar correctamente los objetos.
2. Implementa los métodos indicados en las clases Empleado, Departamento y Rol.
3. En la clase principal, crea varios empleados con diferentes roles y departamentos. Asigna un jefe a cada departamento y muestra la información del empleado y su departamento.
4. Realiza operaciones como cambiar al jefe de un departamento y calcular el bono de los empleados.

## Objetivo:

* Manejar múltiples clases relacionadas entre sí, utilizando la composición de objetos.
* Aplicar relaciones de asociación (empleado-departamento) y agregación (empleado-rol).
* Practicar la implementación de métodos que operan sobre objetos relacionados.

# Ejercicio 7: Gestión de Proyectos y Equipos de Trabajo

## Consigna:

Una empresa está trabajando en varios proyectos, y cada proyecto tiene asignados equipos de trabajo que incluyen empleados con diferentes roles. Se te pide implementar un sistema para gestionar esta información. Deberás crear las siguientes clases:

### **Clase Proyecto**:

#### Atributos:

* + - **nombre**: Nombre del proyecto.
    - **presupuesto**: Presupuesto total asignado al proyecto.
    - **fechaInicio**: Fecha de inicio del proyecto.
    - **equipo**: Objeto de tipo Equipo, que representa el equipo de empleados asignado al proyecto.

#### Métodos:

* + - **agregarEmpleadoAlEquipo(Empleado empleado)**: Agrega un empleado al equipo del proyecto.
    - **calcularCostoProyecto()**: Calcula el costo total del proyecto, sumando los sueldos de los empleados asignados.
    - **mostrarProyecto()**: Muestra la información del proyecto (nombre, presupuesto, fecha de inicio, costo actual).

### **Clase Empleado**:

#### Atributos:

* + - **nombre**: Nombre completo del empleado.
    - **id**: Identificador único del empleado.
    - **sueldo**: Sueldo mensual del empleado.
    - **rol**: Objeto de tipo Rol, que representa el rol del empleado dentro del proyecto.

#### Métodos:

* + - **mostrarEmpleado()**: Muestra la información del empleado (nombre, ID, sueldo, rol).

### **Clase Equipo**:

#### Atributos:

* + - **nombre**: Nombre del equipo (ej. "Equipo A", "Equipo B").
    - **empleados**: Lista de empleados que forman parte del equipo.
    - **jefeEquipo**: Objeto de tipo Empleado, que representa el jefe del equipo.

#### Métodos:

* + - **agregarEmpleado(Empleado empleado)**: Agrega un empleado al equipo.
    - **mostrarEquipo()**: Muestra la información del equipo y los empleados asignados.

### **Clase Rol**:

#### Atributos:

* + - **nombre**: Nombre del rol (ej. "Desarrollador", "Tester").
    - **nivel**: Nivel jerárquico del rol.

#### Métodos:

* + - **mostrarRol()**: Muestra la información del rol.

## Requisitos:

1. Implementa los constructores en todas las clases para inicializar correctamente los objetos.
2. Implementa los métodos indicados en las clases Proyecto, Empleado, Equipo y Rol.
3. En la clase principal, crea varios proyectos con diferentes presupuestos y fechas de inicio. Crea equipos y asigna empleados a cada equipo. Calcula el costo total de cada proyecto en función de los sueldos de los empleados.
4. Simula el cambio de un jefe de equipo y muestra los empleados asignados a cada equipo.

## Objetivo:

* Trabajar con relaciones de composición y agregación, donde un Proyecto está compuesto por un Equipo y un Equipo está compuesto por Empleados.
* Aplicar la lógica empresarial en la gestión de proyectos y su presupuesto.
* Manejar colecciones (como listas) para representar a los empleados dentro de un equipo.