

Guía de ejercicios N° 3

**Cátedra: “Prácticas III”**

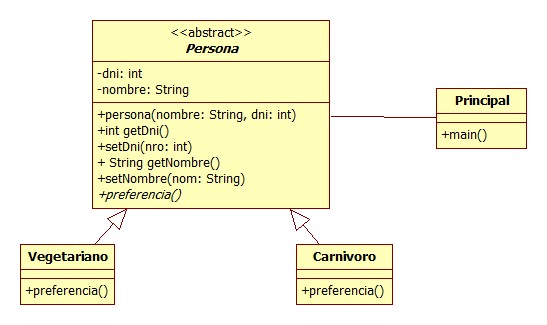
**Docente/s: “Cosgaya, Carolina”**

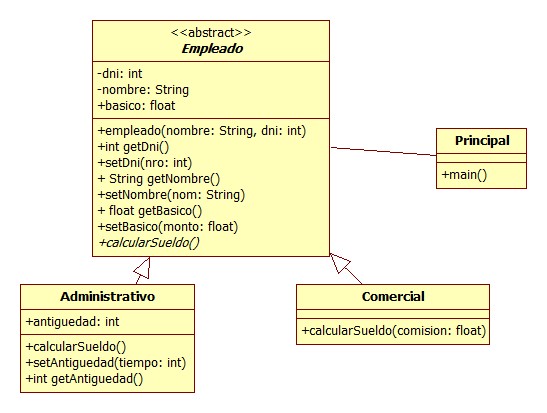
**ANALISTA DE SISTEMAS**

**Ciclo Lectivo 2025**

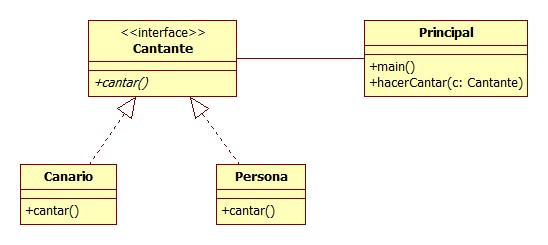
# Unidad N°2: Clases en java y su representación en UML:

Ejercicio N° 1: Clases abstractas y de interface

1. Cree el paquete ejemploClaseAbtracta, dentro del mismo realizar las clases que se observan en el siguiente diagrama teniendo en cuenta lo siguiente:
   1. La clase Persona es abstracta y no tiene definido el método preferencia()
   2. La clase Vegetariano debe redefinir el método preferencia() para que imprima “Verdura”.
   3. La clase Carnivoro debe redefinir el método preferencia() para que imprima “Carne”.
   4. La clase Principal debe contener el método **public** **static** **void** main(String[] args) y dentro de main crear una instancia de Carnivoro y otra de Vegetariano, y para cada una mostrar nombre, dni y preferencia.
2. Cree el paquete ejemplo2ClaseAbtracta, dentro del mismo realizar las clases que se observan en el siguiente diagrama teniendo en cuenta lo siguiente:
   1. La clase Empleado es abstracta y no tiene definido el método calcularSueldo()
   2. La clase Administrativo redefine el método calcularSueldo() y el cálculo es el siguiente:
      1. Se le debe sumar al básico un porcentaje del mismo en base al siguiente cuadro de antigüedad: entre 1 y 3 años 5 %, de 4 a 10 años 15%, de 11 a 20 años 50%, más de 20 años 75%
      2. Imprimir el sueldo
   3. La clase Comercial redefine el método calcularBasico(float comision) y el cálculo es el siguiente:
      1. Se le debe sumar al básico lo ganado por comisión.
      2. Imprimir el sueldo.
   4. La clase Principal debe contener el método **public** **static** **void** main(String[] args) y dentro de main crear una instancia de Administrativo y una de Comercial imprimir sus atributos y el sueldo.



1. Cree el paquete ejemplo3ClaseAbtracta, dentro del mismo realizar las clases que se observan en el siguiente diagrama teniendo en cuenta lo siguiente:
   1. La clase Cantante es de tipo interface y tiene el método abstracto cantar().
   2. La clase Canario redefine el método cantar() que imprime por pantalla “pio pio pio”
   3. La clase Persona redefine el método cantar() que imprime por pantalla “ la la la”
   4. La clase Principal debe contener el método **public** **static** **void** hacerCantar(Cantate c), el cual realiza la instrucción c.cantar();
   5. También debe contener el método **public** **static** **void** main(String[] args) y dentro de main crear una instancia de las clases Canario y Persona. Luego llamar al hacerCantar () pasándole cómo parámetro los objetos creados.



## Ejercicio N° 2: Casos de Uso y Relaciones entre clases (Trabajo integrador)

1. Considere un sistema de Reservas de un Hotel del cual se requieren las siguientes funcionalidades:
   1. CRUD de Habitaciones considerando la capacidad de personas que alojan y tipos de camas (dobles o singles) características (estandar o lujo) y precios, además del estado (Libre, Reservada u Ocupada)
   2. CRUD de Clientes, los clientes pueden ser habituales u ocasionales.
   3. CRUD de usuarios y login de acceso.
   4. Gestión de Reservas y Cancelaciones.
   5. Listados (cada alumno tendrá una consigna particular en este punto)

**Consideraciones para la Gestión de Reservas.**

1. Registrar una reserva, asociando al pasajero registrado en el sistema, las habitaciones requeridas, ingresando además, la cantidad de habitaciones reservadas, la fecha de llegada y salida, si realizó un abono inicial al total de la reserva, previo depósito bancario, confirmando así la reserva realizada.

2. Una reserva puede estar en uno de los siguientes estados: pendientes, confirmadas o anuladas. Pendiente es cuando todavía no se ha realizado en pago por la misma y confirmada cuando se ha realizado dicho pago.

3. Ingresar un abono en dinero para pagar de la reserva y cambiar su estado a confirmada.

4. Anular una reserva buscada según el nombre o apellido del pasajero, cambiando su estado de pendiente a anulada.

5. Consultar la disponibilidad de habitaciones de acuerdo al tipo de habitación y el rango de fechas ingresado por el pasajero.

6.Cambiar el estado de la habitación cuando la reserva se hace efectiva, es decir, cuando llega el cliente. O bien, cuando se libera, se ocupa o se reserva.

1. Realizar el Diagrama de Casos de Uso.
2. Realizar el Diagrama de Clases.
3. Programar las clases identificadas en el punto anterior.

## Ejercicio N° 3: Retomando el ejercicio integrador, implementar las clases abstractas e interfaces necesarias