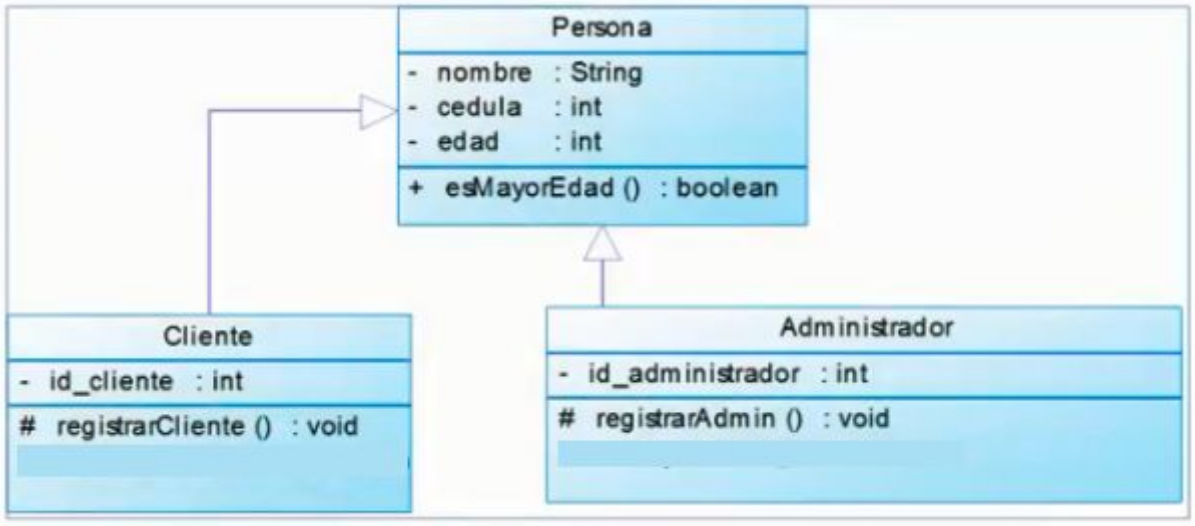
Práctica de Herencia:

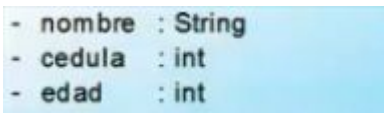
Vimos que la herencia en Java nos permite que **un objeto sea creado a partir de otro** existente, **obteniendo características** como atributos y métodos, lo que nos permite crear objetos derivados a partir de objetos bases. Simplemente **se crea una clase con funciones bases y luego se crea otra clase más (llamada clase hija) que herede de la otra clase (llamada clase padre)** y que además agregue otros atributos y métodos que la definan y la hagan una e**specialización de su clase padre**. Cabe decir también que **en java no se permite la herencia múltiple**, que es heredar de más de un padre, pero se puede arreglar mediante el uso de interfaces aunque la verdad es que si se hace bien la herencia simple no es necesaria la herencia múltiple en ningún lenguaje de programación.

Para la clase de hoy crearemos un proyecto de nombre: "**practicaDeHerencia**" y crearemos 3 clases (Recorda que las clases se nombran con la primera letra en mayúscula y en singular): **Persona, Cliente** y la clase **Administrador**.
El diagrama de clase es el siguiente:



El programa debe permitir registrar clientes solo si son mayores de edad.
Nota: Primero creamos el proyecto las 3 clases y los atributos correspondientes a cada una de las clases.

Una vez que tenemos creada la estructura de las clases; en la clase **Persona**, creamos el constructor y además vemos que los atributos son privados:
Nota: podes cambiar el atributo cedula por dni.



Por lo que debemos crear los getters y los setters para poder obtener y setear los valores de dichos atributos respectivamente.

Recorda que podemos usar la opción que brinda netbeans de insertar código. (Es una buena práctica colocar los setter y los getters juntos. Después de los constructores)

El método esMayorEdad debe retornar verdadero (true) si la edad es mayor o igual a 18.

En la clase **Cliente**, hacemos que esta herede de la clase **Persona**, esto sería: un cliente **"es una"** persona. (En UML)

Implementamos el constructor de Cliente y además incluimos la llamada al constructor del padre con el método **super.**

Nos quedaria algo asi:

```
public Cliente(String nombre, int dni, int edad, int id_cliente){
      super(nombre,dni,edad);
      this.id_cliente = id_cliente;
   }
```

Agregamos el setter y el getter correspondiente. (si no recordas porque lo hacíamos preguntale a la profe.)

Además agregamos el método registrarCliente() en este método debemos mostrar la información del cliente si y solo si la edad del cliente es mayor de edad, es decir tenemos que usar el método esMayorEdad(). En caso de que no sea mayor de edad mostrar un cartel con algún mensaje indicando que no lo es. Te animas a codificarlo sin ayuda?

Codificamos lo mismo para la Clase Administrador. Pero ojo cumplir con los nombre de los atributos y método que se piden en el diagrama de clase.

Luego creamos una clase que se llame Principal donde creamos el método main, dentro del método main creamos un cliente con los siguientes datos:
Id_cliente: 234
Nombre: Homero Simpson
DNI: 9789890
Edad: 14
Luego cambiamos la edad a 50 y llamamos al método registrarCliente().

Creamos un Administrador:
Id_administrador: 23
Nombre: Sr. Thompson
DNI: 8789890
Edad: 15
Luego cambiamos la edad a19 y llamamos al método registrarCliente().

Qué datos se muestran por pantalla?

Te animas a modificar el anterior programa para que puedas cargar una lista de clientes y de empleados que muestre cual de los clientes es el mayor y cual es el menor??

Continuamos viendo más teoría de los ArrayList:

Recorrer un ArrayList

Un ArrayList es una lista de elementos los cual se puede recorrer hacia delante o hacia atrás, ya que sus elementos están enlazados entre sí. Si bien podemos recorrer un ArrayList de la forma tradicional en la que se recorren los array normales.

Para recorrer un ArrayList lo primero será crear un ArrayList.

```
1. ArrayList<String> al = new ArrayList<String>();
```

Y, como no, rellenarlo de elementos:

```
1. al.add("Victor");
2. al.add("Luis");
3. al.add("Elena");
```

Ahora vamos a recorrer un ArrayList como si fuese un array. Para ello tenemos que obtener el tamaño de un ArrayList mediante el método .size().

```
1. int size= al.size();
```

Y ahora utilizaremos un bucle for para recorrer un ArrayList.

```
1. for(int x=0;x<al.size();x++) {
2.   System.out.println(al.get(x));
3. }
```

Para obtener el contenido de cada uno de los elementos vamos a apoyarnos en el método .get(), al cual pasaremos el indicador del índice de la posición que ocupa en el ArrayList. En este caso el índice lo encontramos en la variable x.

De esta forma tan sencilla hemos conseguido recorrer un ArrayList en Java.