La **herencia** significa que se pueden crear nuevas clases partiendo de clases existentes, que tendrá todos los atributos, propiedades y los métodos de su 'superclase' o 'clase padre' y además se le podrán añadir otros atributos, propiedades y métodos propios.

Herencia

Las clases se pueden derivar desde otras clases. La clase derivada (la clase que proviene de otra clase) se llama **subclase**. La clase de la que está derivada se denomina **superclase**.

Una subclase es una clase que desciende de otra clase. Una subclase hereda el estado y el comportamiento de todos sus ancestros. El término superclase se refiere a la clase que es el ancestro más directo, así como a todas las clases ascendentes.

De hecho, en Java, todas las clases deben derivar de alguna clase. La clase más alta, la clase de la que todas las demás descienden, es la clase Object, definida en java.lang. Object es la raíz de la herencia de todas las clases.

Las subclases heredan el estado y el comportamiento en forma de las variables y los métodos de su superclase. La subclase puede utilizar los ítems heredados de su superclase tal y como son, o puede modificarlos o sobreescribirlos. Por eso, según se va bajando por el árbol de la herencia, las clases se convierten en más y más especializadas.

Una clase Java sólo puede tener una superclase directa. Java no soporta la herencia multiple.

```
class SubClass extends SuperClass {
   . . .
}
```

La palabra extends indica que Subclase es extendida de Superclase. Una subclase hereda todas las variables miembros de su superclase que puedan ser accesibles desde la subclase (a menos que la variable miembro esté oculta en la subclase).

Las subclases **heredan** variables miembro declaradas:

- **public**
- **protected**
- sin especificador de acceso siempre que la subclase esté en el mismo paquete que la clase.

Las subclases **NO heredan** variables miembro declaradas:

- con el mismo nombre que en la superclase. La variable miembro de la subclase se dice que oculta a la variable miembro de la superclase.
- **private**

clase padre

Clase de la que desciende o deriva una clase. Las clases hijas (descendientes) heredan (incorporan) automáticamente los atributos, propiedades y métodos de la la clase padre.

Subclase
Clase desciendiente de otra. Hereda automáticamente los atributos, propiedades y métodos de su superclase. Es una especialización de otra clase. Admiten la definición de nuevos atributos y métodos para aumentar la especialización de la clase.

Ventajas de la herencia

- Fácil modificación de código
- Reutilización de código existente
- Adaptación de programas para trabajar en situaciones similares pero diferentes
- Extracción de elementos comunes de clases diferentes
- Organización de objetos en jerarquías

MISCELÁNEA DE EJERCICIOS EJE TEMÁTICO RELACIONES ENTRE CLASES EN JAVA

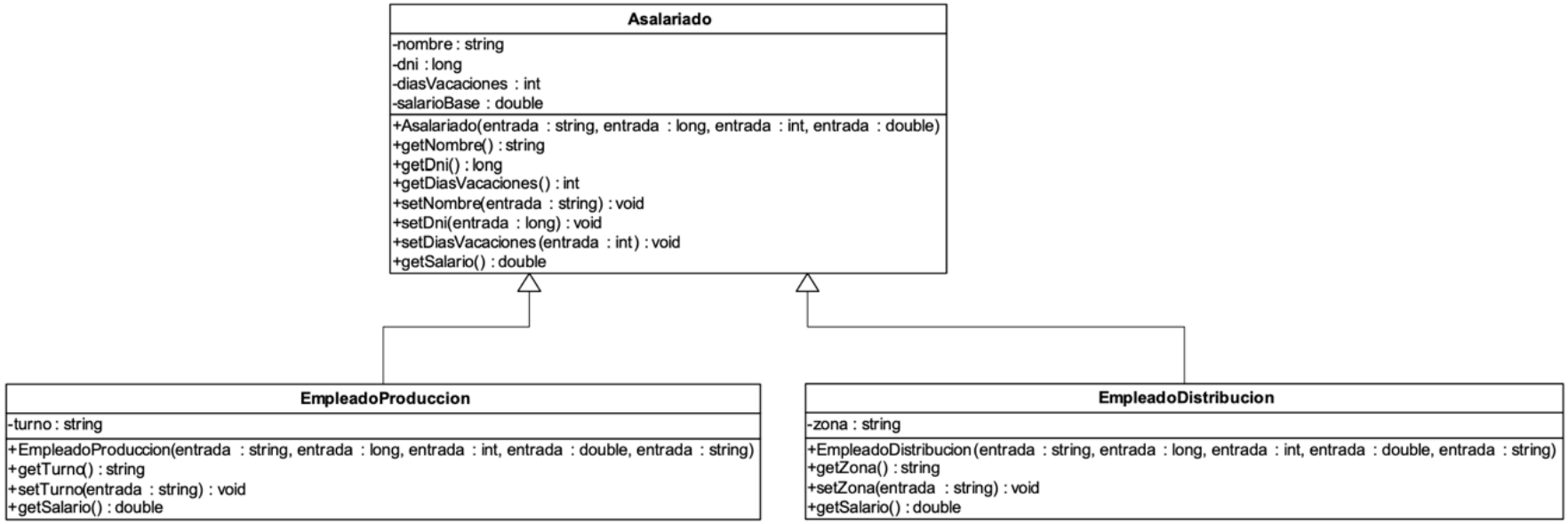
1. Confeccionar una clase Persona que tenga como atributos el nombre y la edad (definir las propiedades para poder acceder a dichos atributos). Definir como responsabilidad un método para imprimir.

    - Plantear una segunda clase Empleado que herede de la clase Persona. Añadir un atributo sueldo (y su propiedad) y el método para imprimir su sueldo.

    - Definir un objeto de la clase Persona y llamar a sus métodos y propiedades. También crear un objeto de la clase Empleado y llamar a sus métodos y propiedades.

2. Plantear una clase Club y otra clase Socio.
    - La clase Socio debe tener los siguientes atributos privados: nombre y la antigüedad en el club (en años). En el constructor pedir la carga del nombre y su antigüedad.
    - La clase Club debe tener como atributos 3 objetos de la clase Socio. Definir una responsabilidad para imprimir el nombre del socio con mayor antigüedad en el club.

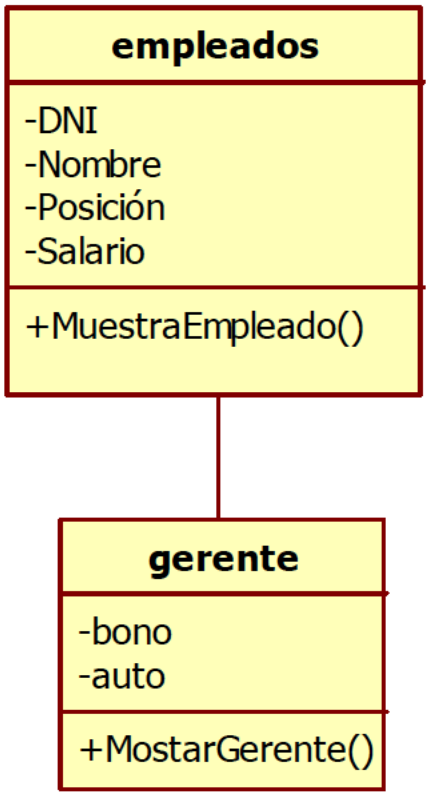
3. De acuerdo al siguiente UML



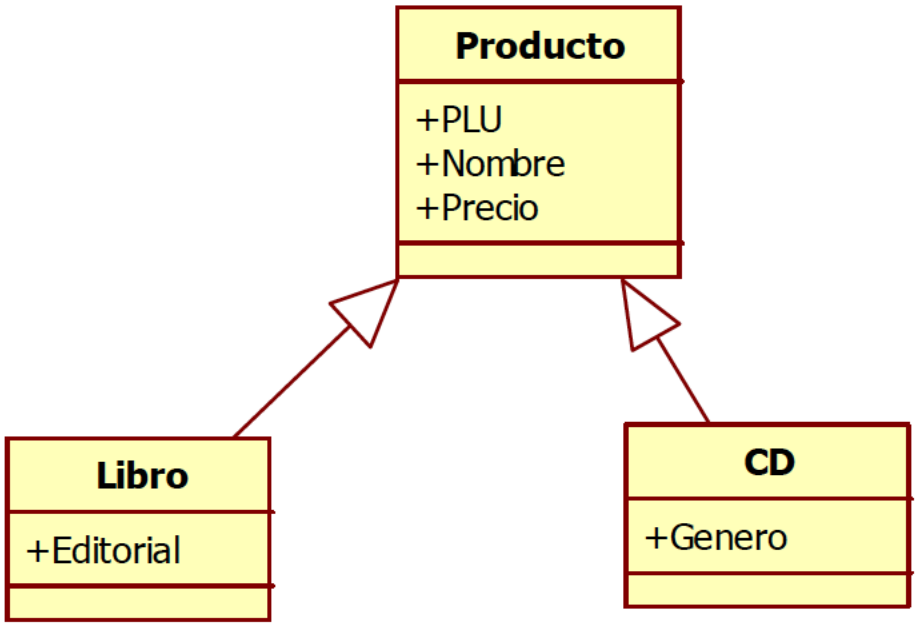
Realice:

- La creación de las tres clases Asalariado con sus atributos, constructor y métodos respectivos, la clase EmpleadoProduccion que herede de asalariado y la clase EmpleadoDistribucion que herede de la clase asalariado. Los empleados de producción reciben sobre el atributo salarioBase un incremento del 15%. Los empleados de distribucion, sin embargo, reciben un aumento de solamente del 10%.

4. Crea una clase empleados y clase gerente con la siguiente estructura, cada uno con su constructor y método correspondiente, crear un objeto de empleado y un objeto de gerente, visualizarlos por consola:



5. Realizar la creación de las clases según el siguiente modelo:



- En la clase producto crear un método verProducto() para visualizar los atributos de dicha clase y en las subclases libro, cd que heredan de la clase producto crear un método en cada una de dichas subclases, para visualizar un objeto de libro y cd con la información heredada de producto más el atributo propio de cada subclase.