

En la programación orientada a objetos, un concepto muy importante y ampliamente utilizado es la herencia. En Java es posible heredar atributos y métodos de una clase a otra.

Cuando se diseña un programa con herencia, se coloca código común en una clase y, luego, se determina a otras clases más específicas que la clase común es su superclase. Cuando una clase hereda de otra, la subclase hereda de la superclase y, consecuentemente, sus atributos y métodos.

Hay dos conceptos importantes cuando una clase hereda de otra clase

- **Subclase (clase hija):** la clase que hereda de otra clase.
- **Superclase (clase padre):** la clase de la que se hereda.

Para heredar de una clase se utiliza la palabra clave **extends**, con el siguiente formato:

```
class ClaseHija extends ClasePadre
```

Los atributos y métodos heredados se pueden utilizar tal como están, reemplazarlos o complementarlos con nuevos atributos y métodos.

Es posible declarar nuevos atributos y métodos en la subclase que no están en la superclase. Se puede reescribir el constructor en la subclase, el cual invoca al constructor de la superclase, ya sea implícitamente o usando la palabra clave **super**.


## Ejercicio 1

Definir una clase Libro para manejar la información asociada a un libro. La información de interés para un libro es: el título, el autor y el precio. Los métodos necesarios son:

- Un constructor para crear un objeto libro, con título y autor como parámetros.
- Sobrecargar el método toString para imprimir en pantalla el título, los autores y el precio del libro.
- Métodos get y set para cada atributo de un libro.

Se debe extender la clase Libro definiendo las siguientes clases:

- **Libros de texto** con un nuevo atributo que especifica el curso al cual está asociado el libro.
- **Libros de texto de la Universidad de Oviedo** : subclase de la clase anterior. Esta subclase tiene un atributo que especifica cuál facultad lo publicó.

- **Novelas:** pueden ser de diferente tipo, histórica, romántica, policíaca, realista, ciencia ficción o aventuras. 
- Para cada una de las clases anteriores se debe definir su constructor y sobrecargar adecuadamente el método toString para visualizar del objeto

## Ejercicio 2

Desarrollar un programa que modele una cuenta bancaria que tiene los siguientes atributos, que deben ser de acceso protegido:

- Saldo, de tipo float.
- Número de consignaciones con valor inicial cero, de tipo int.
- Número de retiros con valor inicial cero, de tipo int.
- Tasa anual (porcentaje), de tipo float.
- Comisión mensual con valor inicial cero, de tipo float.

La clase Cuenta tiene un constructor que inicializa los atributos saldo y tasa anual con valores pasados como parámetros. La clase Cuenta tiene los siguientes métodos:

- Asignar una cantidad de dinero en la cuenta actualizando su saldo.
- Retirar una cantidad de dinero en la cuenta actualizando su saldo. El valor a retirar no debe superar el saldo.
- Calcular el interés mensual de la cuenta y actualiza el saldo correspondiente.
- Extracto mensual: actualiza el saldo restándole la comisión mensual y calculando el interés mensual correspondiente (invoca el método anterior).
- toString: muestra en pantalla los valores de los atributos.

La clase Cuenta tiene dos clases hijas:

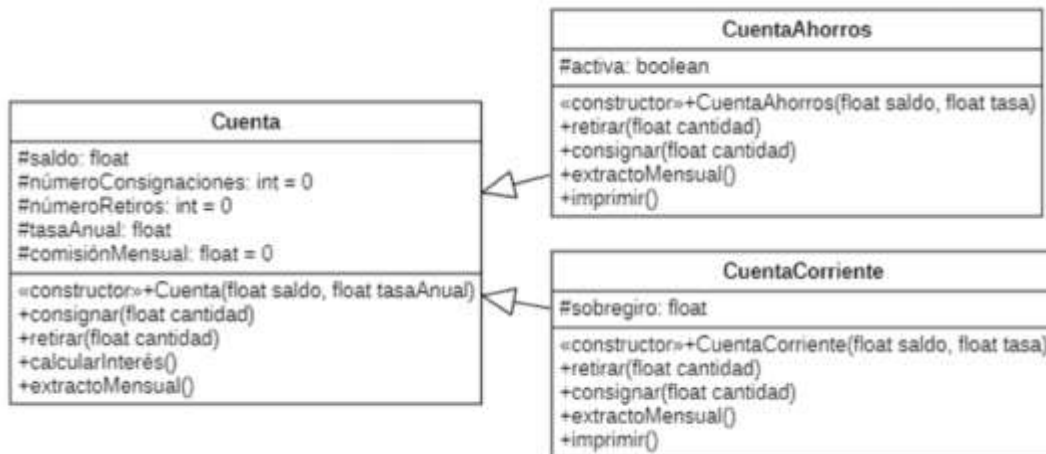
**Cuenta de ahorros:** posee un atributo para determinar si la cuenta de ahorros está activa (tipo boolean). Si el saldo es menor a \$10000, la cuenta está inactiva, en caso contrario se considera activa. Los siguientes métodos se redefinen:

- Consignar: se puede consignar dinero si la cuenta está activa. Debe invocar al método heredado.
- Retirar: es posible retirar dinero si la cuenta está activa. Debe invocar al método heredado.
- Extracto mensual: si el número de retiros es mayor que 4, por cada retiro adicional, se cobra \$1000 como comisión mensual. Al generar el extracto, se determina si la cuenta está activa o no con el saldo.
- Un nuevo método imprimir que muestra en pantalla el saldo de la cuenta, la comisión mensual y el número de transacciones realizadas (suma de cantidad de consignaciones y retiros).

**Cuenta corriente:** posee un atributo de sobregiro, el cual se inicializa en cero. Se redefinen los siguientes métodos:

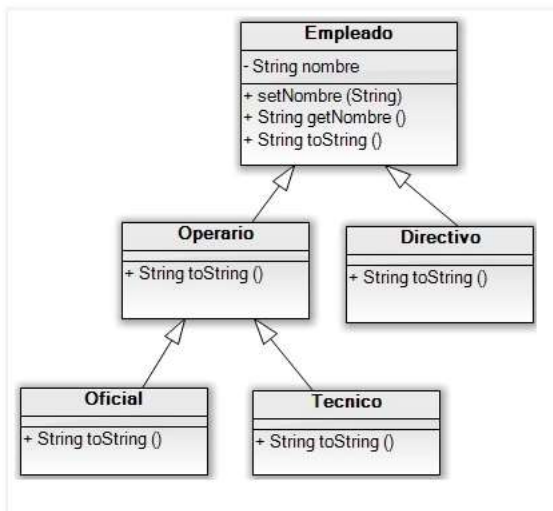
- Retirar: se retira dinero de la cuenta actualizando su saldo. Se puede retirar dinero superior al saldo. El dinero que se debe queda como sobregiro.
- Consignar: invoca al método heredado. Si hay sobregiro, la cantidad consignada reduce el sobregiro.
- Extracto mensual: invoca al método heredado.
- Un nuevo método imprimir que muestra en pantalla el saldo de la cuenta, la comisión mensual, el número de transacciones realizadas (suma de cantidad de consignaciones y retiros) y el valor de sobregiro.

Realizar un método main que implemente un objeto Cuenta de ahorros y llame a los métodos correspondientes.



### Ejercicio 3

Codifica la siguiente jerarquía de clases java representada en el diagrama UML



- La clase base es la clase Empleado. Esta clase contiene:
- Un atributo privado nombre de tipo String que heredan el resto de clases.
- Un constructor por defecto.
- Un constructor con parámetros que inicializa el nombre con el String que recibe.

- Método set y get para el atributo nombre.
- Un método toString() que devuelve el String: "Empleado " + nombre.

El resto de clases solo deben sobrescribir el método toString() en cada una de ellas y declarar el constructor adecuado de forma que cuando la ejecución de las siguientes instrucciones:

Main.java	Sailda
<pre>Empleado E1 = new Empleado("Rafa"); Directivo D1 = new Directivo("Mario"); Operario OP1 = new Operario("Alfonso"); Oficial OF1 = new Oficial("Luis"); Tecnico T1 = new Tecnico("Pablo"); System.out.println(E1); System.out.println(D1); System.out.println(OP1); System.out.println(OF1); System.out.println(T1);</pre>	<pre>Empleado Rafa Empleado Mario -&gt; Directivo Empleado Alfonso -&gt; Operario Empleado Luis -&gt; Operario -&gt; Oficial Empleado Pablo -&gt; Operario -&gt; Tecnico</pre>

## Ejercicio 4

Se plantea desarrollar un programa Java que permita la gestión de una empresa agroalimentaria que trabaja con tres tipos de productos:

- productos frescos,
- productos refrigerados
- productos congelados.

Todos los productos llevan esta información común:

- fecha de caducidad
- número de lote.

A su vez, cada tipo de producto lleva alguna información específica.

- Los productos frescos:
  - la fecha de envasado
  - el país de origen.
- Los productos refrigerados
  - el código del organismo de supervisión alimentaria,

- la fecha de envasado,
- la temperatura de mantenimiento recomendada
- el país de origen.
- Los productos congelados
  - la fecha de envasado, el país de origen
  - la temperatura de mantenimiento recomendada.

Hay tres tipos de productos congelados:

- congelados por aire,
- congelados por agua
- congelados por nitrógeno.

Los productos congelados por aire deben llevar la información de la composición del aire con que fue congelado (% de nitrógeno, % de oxígeno, % de dióxido de carbono y % de vapor de agua).

Los productos congelados por agua deben llevar la información de la salinidad del agua con que se realizó la congelación en gramos de sal por litro de agua.

Los productos congelados por nitrógeno deben llevar la información del método de congelación empleado y del tiempo de exposición al nitrógeno expresada en segundos.

## TAREA

Crear el código de las clases Java implementando una relación de herencia siguiendo estas indicaciones:

1. En primer lugar realizar un esquema con papel y bolígrafo donde se represente cómo se van a organizar las clases cuando escribamos el código. Estudiar los atributos de las clases y trasladar a la superclase todo atributo que pueda ser trasladado.
2. Crear superclases intermedias (aunque no se correspondan con la descripción dada de la empresa) para agrupar atributos y métodos cuando sea posible. Esto corresponde a “realizar

abstracciones” en el ámbito de la programación, que pueden o no corresponderse con el mundo real.

3. Cada clase debe disponer de constructor y permitir establecer (set) y recuperar (get) el valor de sus atributos y tener un método que permita mostrar la información del objeto cuando sea procedente.
4. Crear una clase testHerencia3 con el método main donde se creen: dos productos frescos, tres productos refrigerados y cinco productos congelados (2 de ellos congelados por agua, otros 2 por aire y 1 por nitrógeno). Mostrar la información de cada producto por pantalla.

## Ejercicio 5

Se trata de crear una pequeña base de datos de personas de una universidad. De momento definiremos y probaremos las siguientes clases:

- Direccion: o atributos: calle, ciudad, código postal, país o Constructores predeterminado y parametrizado.
- Persona: Clase ya creada (con nombre, apellidos y NIF, ver ejercicio anterior) a la que añadiremos el atributo dirección y sus métodos get y set. Esta clase implementa la interface Humano, con un método indentificate(), que muestra el tipo de la clase que lo implementa (el tipo de persona, en este caso).
- Estudiante: Subclase de Persona.
  - Atributos: ID de estudiante
  - Constructores : predeterminado y constructor parametrizado que admita el ID.
  - Métodos set , get y toString().
- Profesor: Subclase de Persona.
  - Atributos : despacho o Constructores: predeterminado y constructor parametrizado que admita el despacho.
  - Métodos get, set y toString()

Crea una lista de personas (con la clase Vector) y prueba a añadir varios alumnos y varios profesores a la lista y sus operaciones.