

UNIDAD 8:

HERENCIA

CLASE 4: EJERCICIOS

Ejercicio 1: **profesores**

Define la clase **Fecha** que tendrá los atributos **día**, **mes** y **año**. Sus métodos **constructor** con parámetros y **toString()**.

Define la clase **Profesor** que cuenta con:

- Los atributos **nombre**, **especialidad**, **nombre del centro** y **años consolidados**.
- El método **obtenerSalarioBase()**: Calcula el salario como $(725 + (\text{años consolidados} \times 33.25))$

Define la subclase **ProfesorInterino** que hereda de **Profesor**:

- Tiene los atributos **fecha_findecontrato** y **fecha_iniciodecontrato** de tipo **Fecha**.
- Sobrescribirá el método **obtenerSalarioBase** que será igual al salario base + un porcentaje extra correspondiente al salarioBase de un mes / la cantidad de meses que estará en el centro.

Define la subclase **ProfesorTitular** que hereda de **Profesor**:

- Tiene los atributos **fecha_alta** de tipo **Fecha** y **número de años**.
- Sobrescribirá el método **obtenerSalarioBase** que será igual al salario base calculado por Profesor más la cantidad: 47,80 por años de titular.

Las tres clases deben contar con **Constructor** con y sin parámetros y métodos **get/set** para los atributos.

Los métodos **toString()** de ambas subclases deberán crearse:

- En Titular, muestra el nombre del profesor, la fecha de alta, el centro, la especialidad y su salario.
- En Interino, muestra el nombre del profesor, las fechas de inicio y fin de contrato, el centro, la especialidad y el salario.

Finalmente crea una clase main que pruebe las clases. Calcula los salarios de ambos objetos y muestra su información.

Ejercicio 2: **cajas**

Para hacer envíos por correo se debe entregar dentro de una **Caja** lo que queramos enviar. La caja puede tener medidas de hasta 50 cm en sus tres dimensiones y un peso máximo de 10 kg. Las cajas también llevan una **Etiqueta** que cuenta con la información del destinatario (su nombre, su dirección, la dirección del remitente, la empresa que lo envía, el peso (de nuevo), y una breve descripción sobre el contenido).

La clase **Caja** que crees en java debe contener un constructor al que se le pasen todos los parámetros excepto los relativos a la Etiqueta. Estos valores se introducirán en otro método llamado setEtiqueta. Además se debe crear otro método que calcule el volumen de la caja, otro que calcule cuanto costará el envío (volumen*peso) y otro que nos permita recuperar la información de la caja.

Algunas veces nos interesará almacenar nuestro envío en una **CajaDeCartón** dado que son más livianas que las cajas convencionales. Estas cajas además de contar con todo lo anterior, tendrán un método que calcule la cantidad de cartón utilizado para montarla y sobrescribirán el método que calcula el volumen para reducir su volumen a un 70% dado que cuentan con un descuento.

Ejercicio 3: **coches**

En un concesionario contamos con varios tipos de coches todos ellos tienen en común la velocidad que alcanzan, un precio, un color y se debe calcular su precio en base a sus características propias.

El tipo Truck tiene un peso máximo permitido y su precio variará en función de si peso > 2000, 10% de descuento. Si no , 20% descuento.

Ford tiene un año de fabricacion y un descuento impuesto por el fabricante que tendrá que aplicarse si el coche lleva más de tres años en el concesionario.

El tipo Sedan tiene el atributo longitud que afectará al precio en función de longitud > 4 , 5% descuento, si no, 10% descuento.

Crea todas las clases necesarias con sus constructores y métodos get/set para los campos.

Después crea 2 objetos de todas las clases creadas con valores apropiados en todos los campos de manera que **sucedan todos los casos de uso** y muestra los precios de todos los coches junto con el modelo que son.

Ejercicio 4: **informatica**

Una empresa informática necesita llevar un registro de todos sus empleados que se encuentran en la oficina central, para eso quieren almacenar los siguientes datos de todos sus empleados: su nombre y apellidos, su domicilio, su edad, si está casado o no, su salario base y su clasificación (Junior, Mid o Senior). Tan pronto como se almacenen los datos del empleado se debe establecer la clasificación del empleado según la edad de acuerdo al siguiente criterio: (Si edad es menor o igual a 25 => Junior, Si edad es >=26 y <=35 => Mid y Si edad es >35, => Senior).

Se debe permitir mostrar por pantalla toda la información de un empleado. En cualquier momento debe de poder modificar el salario base en un porcentaje indicado y, por otro lado, recuperar el salario final teniendo en cuenta los beneficios por clasificación: Junior+150 €, Mid+300 € y Senior+600 €

Algunos de los empleados de la empresa son programadores, de los cuales, también se almacenan las líneas de código que escribe por hora y el lenguaje de programación en el que se siente más cómodo. A la hora de calcular su salario final habrá que tener en cuenta un bonus por líneas de código escritas de manera: si es menor que 3000 => +100€, si está entre 3000 y 6000 => +120€ y si es mayor de 6000 => +150€.

Todas las clases creadas deben contar con constructor con parámetros y métodos get/set para sus variables, así como un método `toString`.

Una vez todo esté completo, crea cinco empleados, dos de ellos programadores y calcula sus salarios totales así como muestra su información. Asegúrate de que sus datos sean diferentes para probar todos los casos de uso.

Ejercicio 5: **empresa_alimentaria**

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 y productos congelados.

Todos los productos llevan esta información común: nombre, fecha de caducidad y número de lote. A su vez, cada tipo de producto lleva alguna información específica.

- Los productos frescos deben llevar la fecha de envasado y el país de origen.
- Los productos refrigerados deben llevar el código del organismo de supervisión alimentaria, la fecha de envasado, la temperatura de mantenimiento recomendada y el país de origen.
- Los productos congelados deben llevar la fecha de envasado, el país de origen y la temperatura de mantenimiento recomendada.

Hay tres tipos de productos congelados: congelados por aire, congelados por agua y 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.

Crea todas las clases necesarias para gestionar los alimentos.

Cada clase debe disponer de constructor con parámetros y tener los métodos get/set así como un método que permita mostrar toda la información del producto.

Crear un 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.