# 0485.- Programación.



## U5: Tarea 1 - Herencia

**Actividad 1:** Diseñar la clase **Hora** que representa un instante de tiempo compuesto por una hora (de 0 a 23) y los minutos.

Dispone del constructor:

- Hora (hora,minuto), que construye un objeto con los datos pasados como parámetro

y de los métodos:

- inc(), que incrementa la hora en un minuto
- setMinutos(valor), que da valor a los minutos, siempre y cuando sea un valor con sentido
- setHora(valor),que da valor a la hora, siempre y cuando sea un valor con sentido
- toString(), que devuelve un String con la representación del reloj.

**Actividad 2:** Escribir la clase **Hora12**, que funciona de forma similar a la clase Hora, con la diferencia de que las horas solo pueden tomar un valor entre 1 y 12; y se distingue la mañana de la tarde mediante "am" y "pm".

**Actividad 3:** A partir de la clase **Hora**, implementar la clase **HoraExacta**, que incluye en la hora los segundos. Además de los métodos visibles de *Hora*, dispondrá de:

- HoraExacta(hora, minuto, segundo)
- setSegundo(valor), que da valor a los segundos, siempre y cuando sea un valor con sentido
- inc(), que incrementa la hora en un segundo

**Actividad 4:** Añadir a la clase **HoraExacta** un método que compare si dos horas (la invocante y la otra pasada como parámetro de entrada al método) son iguales o distintas.

**Actividad 5:** Crear la clase **Instrumento**, que es una clase abstracta que almacena un máximo de 100 notas musicales. Mientras haya sitio es posible añadir nuevas notas musicales, al final, con el método **add()**. La clase también dispone del método abstracto **interpretar()** que en cada subclase que herede de **Instrumento**, mostrará por consola las notas musicales según las interprete. Utilizar enumerados para definir las notas musicales.

**Actividad 6:** Crear la clase **Piano** y la clase **Campana** que heredan de **Instrumento.**

**Actividad 7:** Las empresas de transportes, para evitar daños en los paquetes, embalan todas sus mercancías en cajas con el tamaño adecuado. Una caja se crea expresamente con un ancho, un alto y un fondo y, una vez creada, se mantiene inmutable. Cada caja lleva pegada una etiqueta con información útil como el nombre del destinatario, dirección, etc. Se pide implementar la clase **Caja** con constructor:

- Caja(double ancho, double alto, double fondo, Unidades u): que construye una caja con las dimensiones especificadas, que pueden encontrarse en "cm" (centímetros) o en "m" (metros)

y métodos:

- double getVolumen(): que devuelve el volumen de la caja en metros cúbicos
- String toString(): que devuelva una cadena con la representación de la caja.

**Actividad 8:** La empresa de mensajería BiciExpress que reparte en bicicleta, para disminuir el peso a transportar por sus empleados, solo utiliza cajas de cartón. Por motivos de privacidad, las etiquetas (con texto) que se pegan en las cajas normales, se sustituyen por una etiqueta con un número (que determina el cliente, la dirección del envío, etc). Además, las cajas de cartón se caracterizan porque su volumen se calcula como el 80% del volumen real, ya que si las cajas se llenan mucho, se deforman y se rompen. Otra característica de las cajas de cartón es que sus medidas siempre están en centímetros. Por último, la empresa, para controlar costes, necesita saber cuál es la superficie total de cartón utilizado para construir todas las cajas enviadas. Se pide implementar a partir de la clase **Caja** la clase **CajaCarton.**

**Actividad 9:** Crea una supeclase llamada **Electrodomestico** con las siguientes características:

- Los electrodomésticos tienen un **precio base**, un **color**, un **consumo energético** (letras entre A y F) y un **peso**. Estos elementos podrán ser vistos solo por sus clases hijas y por clases vecinas.
- Por defecto, el color será blanco, el consumo energético será F, el precioBase es de 100 € y el peso de 5 kg.
- Los colores disponibles son blanco, negro, rojo, azul y gris.

- Los constructores que se implementarán serán:
  - Un constructor con todos los valores por defecto.
  - Un constructor con el precio y peso. El resto por defecto.
  - Un constructor con todos los atributos.

- Los métodos que implementará serán:
  - métodos get de todos los atributos.
  - **getPrecioFinal()**: según el consumo energético, aumentara su precio, y según su tamaño, también. Esta es la lista de precios:

| Letra | Precio |
|---|---|
| A | 100 € |
| B | 80 € |
| C | 60 € |
| D | 50 € |
| E | 30 € |
| F | 10 € |

| Tamaño | Precio |
|---|---|
| Entre 0 y 29 kg | 10 € |
| Entre 30 y 49 kg | 60 € |
| Entre 50 y 79 kg | 80 € |
| Mayor o igual que 80 kg | 100 € |

  - **toString()**: con el valor de todos los atributos, así como su precio final.

Crearemos una subclase llamada **Lavadora** con las siguientes características:

- Un atributo **carga**, además de los atributos heredados.
- Por defecto, la carga es de 5 kg.
- Los constructores serán:
  - Un constructor por defecto.
  - Un constructor con el precio y peso. El resto por defecto.
  - Un constructor con la carga y el resto de atributos heredados.

- Los métodos que se implementarán serán:
  - Método get de carga.
  - **getPrecioFinal()**: además del incremento por consumo y tamaño, si tiene una carga mayor de 30 kg, aumentará el precio 50 €.
  - **toString()**: con el valor de todos los atributos, así como su precio final.
- Por defecto, la forma de ordenación de objetos Lavadora es la ordenación por carga. Es decir, una lavadora se considera menor que otra si su carga es menor.
- Queremos tener también la opción de poder comparar objetos Lavadora según su precioFinal.

Crearemos una subclase llamada **Television** con las siguientes características:

- Sus atributos son **resolución** (en pulgadas) y **sintonizador TDT** (booleano), además de los atributos heredados.
- Por defecto, la resolución será de 20 pulgadas y el sintonizador será false.
- Los constructores que se implementaran serán:
  - Un constructor por defecto.
  - Un constructor con el precio y peso. El resto por defecto.
  - Un constructor con la resolución, sintonizador TDT y el resto de atributos heredados.

- Los métodos que se implementarán serán:
  - Método get de resolución y sintonizador TDT.
  - **getPrecioFinal()**: si tiene una resolución mayor de 40 pulgadas, se incrementará el precio un 30% y si tiene un sintonizador TDT incorporado, aumentara 50 €. Recuerda que las condiciones que hemos visto en la clase Electrodomestico también deben afectar al precio.
  - **toString()**: con el valor de todos los atributos, así como su precio final.

Ahora crea una clase ejecutable que realice lo siguiente:

- Crea un array de Electrodomesticos de 10 posiciones.

- Asigna a cada posición un objeto de las clases anteriores con los valores que desees.
- Deberás mostrar el contenido del array. (Fíjate sobre todo en que el precio que muestra es el correspondiente a la clase correcta)
- Crea después un array de objetos Lavadora, y ordena el array con la ordenación por defecto de objetos Lavadora.
- Crea otro array y ordénalo por precioFinal.

## Estado de la entrega

| | |
|---|---|
| **Estado de la entrega** | No entregado |
| **Estado de la calificación** | Sin calificar |
| **Última modificación** | - |
| **Comentarios de la entrega** | ▸ Comentarios (0) |

Agregar entrega

Todavía no has realizado una entrega

◄ U5: Doc 1 - Herencia

Ir a...

Lectura Interfaces ►