# Programación en C#

## Actividad

1. (Clases) Crear una clase llamada Vehículo con los siguientes atributos: matrícula, marca (Toyota, Audi, Alfaromeo), tipo de vehículo (Todoterreno, Sedan, Deportivo), número de puertas (3, 5), precio base, llantas de alineación (si tiene o no), techo solar (si tiene o no) y número de airbags (hasta 6). Implementa, además de los constructores y las propiedades, los siguientes métodos:
    - ✓ *PrecioTecho:* el precio se incrementará en un 2,3% si el vehículo dispone de techo solar.
    - ✓ *PrecioPuerta:* si es un vehículo de tres puertas, el precio se decrementará un 0,5%.
    - ✓ *PrecioMarca:* si el vehículo es de la marca Audi y, además, es un modelo deportivo, el precio se incrementará un 120%.
    - ✓ *PrecioLlantas:* se incrementará el precio en 2.500€ si al vehículo se le añaden unas llantas de aleación.
    - ✓ *PrecioAirbag:* el precio se incrementará un 1,5% por cada airbag que se añada.
    - ✓ *ImporteTotal:* devolverá el precio final del vehículo con todos los extras.

    Declara las constantes y enumerados que consideres necesarios para realizar el ejercicio. Además, haz en la clase los controles necesarios para que no se creen atributos con valores incorrectos (ejem: un coche con 2 puertas).
    Crea una clase ejecutable en la que se creen y se muestre el importe final para al menos tres vehículos.

2. (Clases). Crea una clase Empleado con los siguientes atributos: nombre, antigüedad, si es extranjero, el número de horas que ha trabajado a la semana y su nivel (junior, senior o analista). Implementa, además de los constructores y las propiedades, los siguientes métodos:
    - ✓ *SalarioBase:* será el resultado de multiplicar el número de horas que el empleado está contratado a la semana por el precio al que se paga la hora (27 euros). En el salario base sólo se pagarán 40 horas a la semana como máximo.
    - ✓ *HorasExtra*: los trabajadores pueden trabajar un número de horas flexibles. Hasta 40 horas se pagará el salario base. A partir de las 40 horas se pagará un 30% más.

- ✓ *Antigüedad*: cada vez que el empleado cumple un sexenio en la empresa recibe 90 euros más. Ejem: 4 años --> 0€. 6 años -> 90€. 12 años -> 180€. 14 años --> 180€.
- ✓ *Extranjero*: a los empleados que son extranjeros, se le aumenta su salario un 12%.
- ✓ *Nivel:* a los empleados de nivel senior se les aumentará su salario un 25% mientras que a los de nivel analista un 50%.
- ✓ *ImporteBruto:* será el resultado de la suma del salario base, horas extra, antigüedad, si es extranjero más su nivel.
- ✓ *ImporteNeto:* se calculará restando el impuesto al importe bruto.
  - ❖ 15% de impuesto si gana menos de 1400€.
  - ❖ 22% de impuesto si gana entre 1401€ y 1700€.
  - ❖ 27% de impuesto si gana más de 1700€.

Declara las constantes y enumerados que consideres necesarios para realizar el ejercicio. Crea una clase ejecutable en la que se muestre el importe neto de los siguientes empleados:

| Apellidos, Nombre | Antigüedad | Extranjero | Horas | Nivel |
|---|---|---|---|---|
| Adams, Jackson H. | 3 | Yes | 49 | Analista |
| Bailey, Whilemina D. | 11 | No | 42 | Junior |
| Burnett, Kyla I. | 7 | No | 46 | Analista |
| Chapman, Eugenia Y. | 8 | Yes | 37 | Senior |

3. (Clases). Haz una clase Persona que cumpla las siguientes condiciones.
   - ✓ Sus atributos serán: nombre, edad, DNI, sexo (H hombre, M mujer), peso y altura. Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo (0 números, cadena vacía para String, etc.). Sexo será hombre por defecto.
   - ✓ Se crearán varios constructores:
     - ❖ Un constructor por defecto.
     - ❖ Un constructor con el nombre, edad y sexo (el resto por defecto).
     - ❖ Un constructor con todos los atributos como parámetro.
   - ✓ Los métodos que se deben implementar son:
     - ❖ *CalcularIMC*: calculara si la persona está en su peso ideal (peso en kg/(altura^2 en m)). Si esta fórmula devuelve un valor menor que 20, la función devuelve un -1 que significa que la persona está por debajo de su peso ideal. Si devuelve un número entre 20 y 25 (incluidos), la función devuelve un 0 y si devuelve un valor mayor que 25 significa que tiene sobrepeso y la función devuelve un 1. Te recomiendo que uses constantes para devolver estos valores.

- ❖ *MayorDeEdad*: indica si es mayor de edad.
- ❖ *ComprobarSexo*: comprueba que el sexo introducido es correcto. Si no es correcto, será H. Este método no será visible al exterior.
- ❖ *ToString*: devuelve toda la información del objeto.
- ❖ *GeneraDNI*: genera un número aleatorio de 8 cifras y a partir de este su número, se genera su letra correspondiente. Este método será invocado cuando se construya el objeto. Puedes dividir el método para que te sea más fácil. No será visible al exterior.

✓ Ahora, crea una clase ejecutable que haga lo siguiente:

- ❖ Pide por teclado el nombre, la edad, sexo, peso y altura.
- ❖ Crea 3 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado, el segundo objeto obtendrá todos los anteriores menos el peso y la altura y el último por defecto, para este último utiliza los métodos para darle a los atributos un valor.
- ❖ Para cada objeto, deberá comprobar si está en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
- ❖ Indicar para cada objeto si es mayor de edad.
- ❖ Por último, mostrar la información de cada objeto.

4. (Arrays) Haz una clase llamada Password con las siguientes características:
   - ✓ Que tenga los atributos longitud y contraseña.
   - ✓ Un constructor por defecto. Por defecto, la longitud será de 8.
   - ✓ Un constructor con la longitud que nosotros le pasemos. Genera una contraseña aleatoria con esa longitud.
   - ✓ Crea las propiedades.
   - ✓ Los métodos que implementa serán:
     - ❖ EsFuerte(): devuelve un booleano si es fuerte o no, para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 5 números.
     - ❖ GenerarPassword(): genera la contraseña del objeto con la longitud que tenga.
   - ✓ Ahora, crea una clase ejecutable:
     - ❖ Crea un array de Passwords con el tamaño que le indiques por teclado
     - ❖ Crea un bucle que cree un objeto para cada posición del array.
     - ❖ Indica también por teclado la longitud de los Passwords (antes de bucle).

- ❖ Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).
- ❖ Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior). Usa este simple formato:
  - contraseña1 valor_booleano1
  - contraseña2 valor_bololeano2

5. (Herencia) Crearemos una clase llamada Electrodoméstico con las siguientes características:
   - ✓ Sus atributos son precio base, color, consumo energético (letras entre A y F) y peso.
     - ❖ Por defecto, el color será blanco, el consumo energético será F, el precioBase es de 100 € y el peso de 5 kg. Usa constantes para ello.
     - ❖ Los colores disponibles son blanco, negro, rojo, azul y gris. No importa si el nombre está en mayúsculas o en minúsculas.
   - ✓ Los constructores que se implementarán serán:
     - ❖ Un constructor por defecto.
     - ❖ Un constructor con el precio y peso. El resto por defecto.
     - ❖ Un constructor con todos los atributos.
   - ✓ Los métodos que implementara serán:
     - ❖ Propiedades
     - ❖ ComprobarConsumoEnergetico(char letra): comprueba que la letra es correcta, sino es correcta usara la letra por defecto. Se invocará al crear el objeto y no será visible.
     - ❖ ComprobarColor(String color): comprueba que el color es correcto, sino lo es usa el color por defecto. Se invocará al crear el objeto y no será visible.
     - ❖ PrecioFinal(): 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 |
|---|---|
| 0 – 19 Kg | 10€ |
| 20 – 49 Kg | 50€ |
| 50 – 79 Kg | 80€ |
| > 80 Kg | 100€ |

- ✓ Crearemos una subclase llamada Lavadora con las siguientes características:
  - ❖ Su atributo es carga, además de los atributos heredados.
  - ❖ Por defecto, la carga es de 5 kg. Usa una constante para ello.
  - ❖ Los constructores que se implementarán 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. Recuerda que debes llamar al constructor de la clase padre.
  - ❖ Los métodos que se implementara serán:
    - Propiedades.
    - PrecioFinal(): si tiene una carga mayor de 30 kg, aumentara el precio 50 €, sino es así no se incrementara el precio. Llama al método padre y añade el código necesario. Electrodoméstico también deben afectar al precio.
- ✓ Crearemos una subclase llamada Televisión 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 implementarán 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 implementara serán:

- Propiedades y sintonizador TDT
- PrecioFinal(): si tiene una resolución mayor de 40 pulgadas, se incrementara el precio un 30% y si tiene un sintonizador TDT incorporado, aumentara 50 €. Recuerda que las condiciones que hemos visto en la clase Electrodoméstico también deben afectar al precio.

✓ Ahora crea una clase ejecutable que realice lo siguiente:

- ❖ Crea un array de Electrodomésticos de 10 posiciones.
- ❖ Asigna a cada posición un objeto de las clases anteriores con los valores que desees.
- ❖ Ahora, recorre este array y ejecuta el método precioFinal().
- ❖ Deberás mostrar el precio de cada clase, es decir, el precio de todas las televisiones, por un lado, el de las lavadoras por otro y la suma de los Electrodomésticos (puedes crear objetos Electrodoméstico, pero recuerda que Televisión y Lavadora también son electrodomésticos).
- ❖ Por ejemplo, si tenemos un Electrodoméstico con un precio final de 300, una lavadora de 200 y una televisión de 500, el resultado final será de 1000 (300+200+500) para electrodomésticos, 200 para lavadora y 500 para televisión.

6. (Herencia/Polimorfismo/Interfaces) Desarrollar un programa para simular el funcionamiento de un juego siguiendo las especificaciones que se detallan a continuación:

✓ En el juego se dispondrá de varios tipos de personajes (sacerdote, cazador o curandero). Independientemente del tipo, todos los personajes tendrán la siguiente información:

- ❖ Una cadena de texto con el nombre del personaje, un entero que representará su energía y un entero con la vida que le queda.
  Por defecto, los distintos personajes tendrán los siguientes atributos:
  - Cazador.
    - Vida: 50.
    - Energía: 80.
    - Dispondrá de un atributo específico que nos dirá el arma que lleva. Se podrá elegir entre arco, escopeta o pistola.
  - Sacerdote.
    - Vida: 100.

        - Energía: 80.
    - Curandero.
        - Vida: 50.
        - Energía: 100.

❖ Los siguientes métodos:

- *Defender*. Nos devolverá si el personaje ha podido defenderse de un ataque realizado (se generará un número aleatorio entre 1 y 3 considerando que la defensa ha sido exitosa solo si sale un 1).
- *RecibirGolpe*. Se decrementa el golpe completo de la vida del personaje sólo si el personaje no ha podido defenderse. Si ha podido defenderse, sólo se decrementará la mitad. Se mostrará por pantalla el nombre del personaje que ha recibido el golpe y su vida restante después de recibirlo.

  Se muestra por pantalla el personaje que ha recibido el golpe y si ha conseguido defenderse. Si el personaje se muere, deberá aparecer un mensaje informando.
- *Atacar*. Aunque es de obligada implementación, dependerá de cada personaje.
    a. Cazador: tiene una probabilidad de 5/10 de acertar el golpe. Si acierta, generará de forma aleatoria un golpe de fuerza entre ¼ y ¾ de su energía disponible. Una vez efectuado, decrementará el golpe de la energía del personaje.
    b. Curandero: generará de forma aleatoria un golpe de fuerza entre 0 y ¼ de su energía disponible. Una vez efectuado, decrementará el golpe de la energía del personaje.
    c. Sacerdote: generará de forma aleatoria un golpe de fuerza entre ¼ y ¾ de su energía disponible. Una vez efectuado decrementará el golpe de la energía del personaje.

  Se muestra por pantalla el personaje que ha atacado y el golpe de fuerza que ha generado. Si se trata de atacar un personaje que está muerto, aparecerá un mensaje informando del error.
- *EstaVivo*. Devuelve true si el personaje está vivo. **Un personaje que está muerto no podrá realizar ninguna acción.**

- *Comer*. Cuando un personaje come, su energía le sube la mitad.
- *ToString*. Se mostrará por pantalla la clase del personaje y todos sus atributos.

✓ Todos los personajes de sanación (curandero y sacerdote) deberán implementar los siguientes métodos:

❖ *Curar.*

- Sacerdote. Se cura a sí mismo un cuarto de su energía disponible. Se debe decrementar la energía gastada en la sanación de la energía del personaje.
- Curandero. Se cura a sí mismo la mitad de su energía disponible. Se debe decrementar la energía gastada en la sanación de la energía del personaje.

Se mostrará el personaje que se ha curado y su vida después de la cura.

❖ *CurarOtros.*

- Sacerdote. Cura a otro personaje elegido un cuarto de su energía disponible. Se debe decrementar la energía gastada en la sanación de la energía del personaje.
- Curandero. Cura a otro personaje elegido la mitad de su energía disponible. Se debe decrementar la energía gastada en la sanación de la energía del personaje.

Se mostrará el personaje que se ha curado y su vida después de la cura. Si se trata de curar un personaje que está muerto se mostrará el mensaje de error por pantalla.

✓ La partida de juego permitirá gestionar distintos jugadores independientemente de su clase. Éstas serán sus funcionalidades:

❖ Crear un nuevo jugador.
❖ Atacar a otro personaje.
❖ Curar a otro personaje. Sólo los personajes de sanación podrán realizar esta acción.
❖ Curarse a sí mismo. Sólo los personajes de sanación podrán realizar esta acción.
❖ Cuando un personaje se muera, se deberá eliminar de la lista de personajes de la partida.
❖ Mostrar todos los personajes de sanación.
❖ Comer.
❖ Mostrar la vida de todos los personajes.

**Nota**: se utilizará el manejo de excepciones en el ejercicio.