

Ejercicios Enumeraciones

Índice

1. [Ejercicio 1](#)
2. [Ejercicio 2](#)
3. ☒ [Ejercicio 3](#)
4. [Ejercicio 4](#)
5. ☒ [Ejercicio 5](#)

Ejercicio 1

Crea una enumeración Color e inicializala con los siguientes valores en **hexadecimal**:

```
Rojo → 0xFF0000
Azul → 0x0000FF
Verde → 0x00FF00
Amarillo → 0xFFFF00
Violeta → 0xC71585
Blanco → 0xFFFFFFFF
Negro → 0x000000
Gris → 0xCDCDCD
Marron → 0xA52A2A
```

1. Recorre todos los elementos de la enumeración, mostrando el nombre del color en el enum y su valor correspondiente en hexadecimal.
2. Pide un color al usuario y muestra su valor en hexadecimal si existe en la enumeración, o un mensaje de error si no existe.

📖 **Recuerda:** existe una serie de métodos de la clase Enum, que nos permitirán trabajar con las enumeraciones. Algunos de utilidad para este ejercicios pueden ser Enum.IsDefined() y Enum.GetNames().

Ejercicio 2

Crea un programa que permita controlar el **coste del abono de transporte urbano** de una determinada ciudad.

- Existirán diferentes tipos de abonos: **QuinceDias, TreintaDias, FamiliasNumerosas, TerceraEdad, Discapacidad, Juvenil, Infantil, Turístico**.
- El coste del viaje de cada una de estas tarifas será, respectivamente, el siguiente: **0.70, 0.60, 0.50, 0.30, 0.20, 0.65, 0.35, 0.90**. Frente al precio de **1.20** del viaje sin tarifa.
- Los abonos se podrán comprar para un **mínimo de 7 días** (descartando los dos primeros que son para **15** y **30** días respectivamente), y un **máximo de 60 días**.

El programa deberá:

1. pedir al usuario **el tipo de abono que quiere comprar, y para cuantos días** (teniendo en cuenta que los dos primeros no necesitan este datos).
2. **Calcular el coste** del Abono, mostrando total a pagar por pantalla.
3. **Controlar que la entrada de datos es correcta**, mostrando mensajes de aviso en caso contrario.

Se deberán crear los métodos necesarios para que el código siga las normas de modularidad que ya vimos en temas anteriores.

💡 **Tip:** Para una mejor codificación deberás asociar los precios del viaje a la enumeración. Como a las enumeraciones solo se pueden asociar a valores enteros, podrás dividir entre 100 el valor asociado para que sea real.

✓ Ejercicio 3

Crea un método **genérico** llamado `LeerEnum`, que permitirá recoger un string del usuario hasta que coincida con un elemento de la enumeración X. El método tendrá que comprobar si la string introducida por el usuario pertenece a la enumeración, si pertenece devolverá el elemento de la enumeración que coincida, si no mostrará todos los elementos de la enumeración y volverá a pedir el string al usuario.

Este método tendrá la siguiente signature:

```
public static Object LeerEnum(Type tipo, string texto, string textoError);
```

Donde:

- **tipo**: El tipo de la enumeración obtenido con la función `typeof(IdentificadorEnumeracion)`.
- **texto**: El texto que se muestra para pedir al usuario que introduzca el valor de la enumeración.
- **textoError**: El texto que se mostrará si el valor que se introduce no pertenece a la enumeración.

Modifica el **Ejercicio 2** para usar el método `LeerEnum` a la hora de recoger el tipo de Abono a comprar.

👉 **Pista:** Este método utilizará `Enum.IsDefined` para comprobar si el valor pertenece a la enumeración, `Enum.Parse` para convertir el string a tipo `Enum` y `Enum.GetNames` para crear el array que se mostrará en caso de que el dato sea erróneo.

Ejercicio 4

Crea una enumeración para gestionar una respuesta, que puede ser **Si, No, Nose**.

- Para eso se necesitaran tres métodos `LeePregunta`, `VisualizaRespuesta` y `LeeEnum`.
- Al método `LeePregunta` se le pasará una cadena, que será la que se muestre para pedir la respuesta y asignar a la variable respuesta, siempre que sea válida (para ello utilizaremos el método `LeeEnum` del ejercicio anterior).
- A `VisualizarRespuesta` se le pasará la cadena que se mostrará para visualizar la respuesta y mostrará el valor de la respuesta en ese momento.

✓ Ejercicio 5

Crea una aplicación para gestionar la personalización de coches en un determinado taller.

- Tendrás que utilizar **Enumeraciones NO excluyentes**, debes definir la enumeración con un mínimo de 7 colores (incluido el `None`).
- La aplicación permitirá añadir un color o más a la elección, eliminar un color de los que ya se habían elegido y mostrar los colores elegidos.
- El programa comenzará mostrando un menú, con las tres opciones y la que nos permita salir.

Nota: Deberás usar el método `LeeEnum`, para introducir los datos que se piden al usuario y tendrás que crear, como mínimo, un método para cada una de las posibles opciones del menú.