Relación de Ejercicios de Clases y Objetos (5): Dinero y Moneda

En esta relación, implementaremos la clase **Dinero** que nos permitirá guardar información sobre una cantidad de dinero expresada en una divisa concreta (p.ej.: 23,45 dólares). Para que todo funcione como es debido, mantendremos una lista con los cambios de moneda (p.ej.: cuántos dólares corresponden a un euro) que se mantendrá de manera estática para todos los objetos de la clase dinero, de manera que todos usaran la misma tabla para las conversiones.

Para que todo esto funcione, necesitaremos los siguientes elementos:

1) Un **enum** que se llamará **TipoMoneda** en el que tendremos una lista de las monedas con las que trabajaremos. El hecho de que esté en un *enum* le facilita al usuario el poder escoger la moneda fácilmente.

2) Una clase **Moneda** que nos servirá para mantener la información sobre cada moneda que usemos. Tendrá los siguientes componentes:
   - Atributos:
     - **tmoneda**, de tipo TipoMoneda
     - **decimales**, de tipo entero, en la que guardaremos cuántos decimales se usan para esa moneda (p.ej.: las pesetas no usaban decimales, los euros usan 2, etc.).
     - **simbolo**, de tipo cadena, en la que almacenaremos el símbolo que se usa para representar la moneda (p.ej.: $, €, etc.).
     - **cambioEuro**, de tipo decimal, en la se guardará el cambio con respecto al euro de cada moneda. Más exactamente, guardaremos a qué cantidad de esta moneda corresponde un euro (p.ej.: 1 € = 136.049717 Yen, ese valor será el que guardaremos).
   - Constructores:
     - Tendrá un único constructor que leerá los cuatro valores y los pondrá en los cuatro atributos. Haremos la comprobaciones pertinentes: que el número de decimales sea sensato (entre 0 y 4 es lo normal), que la cadena del símbolo no esté vacía y que el cambio con el euro no sea negativo.
   - Propiedades:
     - Tendremos propiedades de sólo lectura para acceder a los tres atributos que no van a cambiar: **tmoneda**, **decimales** y **simbolo**.
     - Para el atributo **cambioEuro**, tendremos una propiedad de que nos permitirá leerlo y modificarlo, teniendo en cuenta de que el valor no puede ser negativo.

3) Una clase **Dinero** que contendrá la lista de monedas y los datos correspondientes a cada cantidad de dinero concreta:
    a. Monedas (parte estática):
        - Tendremos un atributo estático que será una lista de monedas. Será estático para que una vez rellena con datos la lista, nos sirva para todas las cantidades de dinero (es independiente de cada objeto).
        - Tendremos un constructor estático (y yo que pensaba que no los íbamos a usar nunca) que rellenará la lista de monedas con los datos necesarios.
          Por cada valor que tengamos en **TipoMoneda**, tendremos que añadir una entrada a esa lista de monedas con los datos de esa moneda concreta.
          Este constructor se ejecuta automáticamente una sola vez cuando vayamos usar el primer objeto de la clase **Dinero**, así que tendremos la lista rellena cuando nos haga falta.
        - Tendremos un método estático **ActualizaCambio** al que le pasaremos un TipoMoneda y un nuevo valor para el cambio en euros de esa moneda y lo actualizará en la lista.
        - A medida que vayáis escribiendo el resto de componentes de esta clase, os harán haciendo falta varios métodos para ir accediendo fácilmente a la información que hay en esta lista. Todos esos métodos hacedlos privados para que no se vean desde fuera.
    b. Dinero:
        - Atributos:
            - **cantidad**, será un decimal en el que se guardará la cantidad exacta de dinero, independientemente del número de decimales que use la moneda concreta.
            - **moneda**, será del **TipoMoneda** y nos guardará en que moneda está representada la cantidad.
        - Constructores:
            - Un constructor con dos parámetros, cantidad (decimal) y moneda (TipoMoneda), que inicializará los atributos.
            - Otros dos constructores para que podamos pasarle un entero y un *double* en lugar de un decimal para la cantidad.
        - Propiedades:
            - Las necesarias para acceder a los atributos (serán de lectura y escritura).
        - Métodos:
            - **ToString()**, habrá que sobrescribirlo para que muestre el valor contenido en cantidad con el número de decimales correspondientes a nuestra moneda y con el símbolo de la moneda detrás. Para redondear podéis usar **Math.Round()**.
            - **ValorEn(TipoMoneda)**, nos devolverá un decimal que corresponderá al valor de nuestro dinero en la moneda que nos pasen por parámetro
            - **ConvierteEn(TipoMoneda)**, nos devolverá un objeto de tipo Dinero que corresponderá al valor de nuestro dinero expresado en la moneda que nos pasen por parámetro.

        - **ToString(TipoMoneda)**, será como el ToString() de más arriba pero nos escribirá la cantidad representada en la moneda que nosotros le digamos (con sus decimales correspondientes y su simbolito).
    - Más propiedades:
        - **ADouble**: nos devuelve la cantidad pero como *double* en lugar de decimal.
        - **AEntero**: nos devuelve la cantidad pero como entero en lugar de decimal.
    - Operadores (aquí llega la diversión):
        - Hay que definir, básicamente todos los operadores. O sea, los cuatro aritméticos y los seis lógicos (+,-,*,/,==,!=,<,>,<=,>=).
        - Cuando los dineros que nos pasan sean de la misma moneda, no hay ningún problema en operar con ellos, así que lo haremos directamente.
        - Cuando los dineros que nos pasan sean de distinta moneda, convertiremos el segundo operando a la moneda del primero y luego haremos las operaciones pertinentes.
        - En el caso de los operadores aritméticos, en los que tendremos que devolver una variable de tipo Dinero, la moneda usada será la moneda del primer operando.
          Ej.: 2 euros + 3 dólares = 2 euros + 2.6544 euros = 4.6544 euros.
        - Los operadores * y / reciben un Dinero y un valor decimal, ya que multiplicar 1 euro por 1 dólar es un poco absurdo.

Ejercicios adicionales (completamente opcional)

Estaría muy chulo que la lista de cambio de monedas se actualizara automáticamente de internet. Para ello, usaremos la clase **WebClient** que nos permite descargar páginas HTTP.

Necesitamos poner un **using System.Net**, crear un objeto **WebClient** y usar el método **DownloadString** al que le pasamos una página web y nos la pone en una cadena para poder trabajar con ella.

Por ejemplo, para descargar el cambio de Euros a Dólares, podemos usar esta página:

http://www.x-rates.com/calculator/?from=EUR&to=USD

Una vez descargada, tendremos que buscar dentro de la página (con esas funciones tan maravillosas de cadenas) el valor de la conversión y actualizarlo en la lista.

Se recomienda añadir lo siguiente a las clases ya creadas:

- Clase Moneda
    - Añadir un atributo **codigo** en cada moneda con el código de la moneda (ej.: euro = EUR, libra = GBP, dólar = USD, etc.).
    - Modificar el constructor para que haya que pasarle el código de la moneda y lo guarde en el atributo.
- Clase Dinero

- En el constructor estático, añadir los códigos de todas las monedas a la lista.
- Añadir un método **ActualizaListaInternet** que automáticamente vaya actualizando todas las monedas que tenemos en la lista bajándose las páginas de los cambios de internet.