**ABSTRACT CLASS: Etiqueta**

Etiqueta es una clase abstracta, pues nunca se van a instanciar objetos de dicha clase, pero sí de sus subclases: **EtiquetaSimple, EtiquetaPersona, EtiquetaEspecial**.

Esta clase tampoco tiene atributos, pues estas tres clases solo tienen en común que son etiquetas. La razón por la cual es necesario trabajar esta clase de esta forma es para que en la clase **Imagen** se pueda crear una sola lista con todas las etiquetas asignadas a cada objeto de esa clase **(polimorfismo).** Además, no sabemos cómo se va a comportar esta clase cuando agreguemos la interfaz de usuario y es posible que tengamos que agregarle métodos a dicha clase. Por comodidad, creamos esta clase.

**CLASS: EtiquetaSimple**

Esta clase se usa para agregar las Etiquetas simples en una Imagen, y **hereda de Etiqueta**. Tal como dice el enunciado, esta etiqueta únicamente debe tener una frase. De esta forma, tiene un atributo que sirve para almacenar dicha frase y un valor DEFAULT para su longitud. Además, tiene constructor y getter/setter para dicho atributo.

Atributos:

* **- frase : string** => Atributo donde se almacena la frase de la etiqueta
* **- DEFAULT\_LONGITUD\_MAX\_FRASE:int** => Atributo para la longitud default de la frase (por fijar con el equipo)

Metodos:

* **+ EtiquetaSimple()** => Constructor parameterless
* **+ EtiquetaSimple(string frase)** => Sobrecarga constructor
* **+ Frase(string frase):bool** => Setter de frase. En el se debe controlar que la etiqueta no supere una cierta longitud máxima soportada por todas las etiquetas simples. Retorna true si el cambio se realizó con éxito y false si no es así.
* **+ Frase() : string** => Getter de frase

**CLASS: EtiquetaPersona**

Esta clase sirve para etiquetar personas en una Imagen, y hereda de Etiqueta. Tal como dice el enunciado, en esta etiqueta se deben poder agregar múltiples atributos sobre la persona que se está etiquetando. Además, para crear la etiqueta se debe seleccionar el rostro de la persona en la imagen. Así, uno de los atributos de esta clase es un array de 4 doublés, donde el primero y el segundo representan el largo y el ancho del rectángulo del rostro y el tercero y el cuarto la distancia del borde izquierdo y superior de la imagen hasta dicho rectángulo (todas medidas en pixeles). Se asume que la interacción para seleccionar el rectángulo lo realiza la interfaz de usuario, cosa que aun no se desarrolla en esta entrega. Además, los colores posibles se guardan en un archivo EColores, las dos opciones de sexo en un ESexo y las nacionalidades en ENacionalidades. Para poder crear una etiqueta de persona, el mínimo parámetro necesario es el nombre y las coordenadas del rectángulo de la cara de la persona.

Atributos:

* **- nombre : string** => Atributo obligatorio que toda etiqueta de persona debe tener
* **- DEFAULT\_APELLIDO : string** => Atributo de apellido por default. Deberá ser null.
* **- apellido : string** => Atributo para almacenar el apellido de la persona
* **- DEFAULT\_NACIONALIDAD : ENacionalidades** => Atributo de nacionalidad por default. Debera ser none (none pertenece a ENacionalidades)
* **- nacionalidad : ENacionalidades** => Atributo de nacionalidad de las etiquetas, cuyo tipo debe ser el enum ENacionalidades
* **- DEFAULT\_COLOR : EColores** => Atributo de color por default para colorOjos y colorCabello. Debera ser none (none pertenece a EColores)
* **- colorOjos : EColores** => Atributo de color de ojos de las etiquetas, cuyo tipo debe ser el enum EColores
* **- colorCabello : EColores** => Atributo de color de cabello de las etiquetas, cuyo tipo debe ser el enum EColores
* **- DEFAULT\_SEXO : ESexo** => Atributo de sexo por default. Deberá ser none (none pertenece a ESexo)
* **- sexo : ESexo** => Atributo de sexo de las etiquetas, cuyo tipo debe ser el enum ESexo
* **- DEFAULT\_FECHANACIMIENTO : DateTime** => Atributo de fecha de nacimiento por default. Debera ser 01-01-1900.
* **- fechaNacimiento : DateTime** => Atributo de fecha de nacimiento de las etiquetas, cuyo tipo debe ser DateTime
* **- coordenadas : double[4]** => Atributo obligatorio que toda etiqueta de persona debe tener

Métodos:

* **+ EtiquetaPersona(string nombre, double[4] coordenadas)** => Constructor con menos parámetros posibles
* **+ EtiquetaPersona(string nombre, string apellido, double[4] coordenadas)** => Sobrecarga del constructor
* **+ EtiquetaPersona(string nombre, string apellido, ENacionalidades nacionalidad, double[4] coordenadas)** => Sobrecarga del constructor
* **… Resto de sobrecargas de constructores encadenados entre sí, llegando al último constructor que debe asignar valores para todos los atributos de la clase. El encadenamiento se debe realizar con los DEFAULT values.**
* **+ Nombre(string nombre) : bool** => Setter del atributo nombre. Retorna true si se logra hacer el cambio y false si no
* **+ Nombre() : string** => Getter del atributo nombre
* **+ Apellido(string apellido) : bool** => Setter del atributo apellido. Retorna true si se logra hacer el cambio y false si no
* **+ Apellido() : string** => Getter del atributo apellido
* **+ Nacionalidad(ENacionalidades nacionalidad) : bool** => Setter del atributo nacionalidad. Retorna true si se logra hacer el cambio y false si no
* **+ Nacionalidad() : ENacionalidades** => Getter del atributo nacionalidad
* **+ ColorOjos(EColores color\_ojos) : bool** => Setter del atributo colorOjos. Retorna true si se logra hacer el cambio y false si no
* **+ ColorOjos() : EColores** => Getter del atributo colorOjos
* **+ ColorCabello(EColores color\_cabello) : bool** => Setter del atributo colorCabello. Retorna true si se logra hacer el cambio y false si no
* **+ ColorCabello() : EColores** => Getter del atributo colorCabello
* **+ Sexo(ESexo sexo ) : bool** => Setter del atributo sexo. Retorna true si se logra hacer el cambio y false si no
* **+ Sexo() : ESexo** => Getter del atributo sexo
* **+ FechaNacimiento(DateTime fecha\_nacimiento) : bool** => Setter del atributo fechaNacimiento. Retorna true si se logra hacer el cambio y false si no
* **+ FechaNacimiento() : DateTime** => Getter del atributo fechaNacimiento
* **+ Coordenadas(double[4] coordenadas) : bool** => Setter del atributo coordenadas. Retorna true si se logra hacer el cambio y false si no
* **+ Coordenadas() : double[4]** => Getter del atributo coordenadas

**CLASS: EtiquetaEspecial**

Esta clase sirve