Ejercicios Introducción a C#

1. Clases I y Repaso de String y Lists

**INSTRUCCIONES DE ENTREGA**

Para cada ejercicio se requiere:

1. La carpeta entera de todos los proyectos con todos los archivos generados.
2. **PROHIBIDO** el uso de Excepciones y Constructores.
3. Todos los proyectos van a poder ejecutarse desde el primero, es decir, ***1MayoresDeEdad.***
   1. Desarrollar un programa que se llamará ***1MayoresDeEdad***. Creará X instancias de la clase persona y pedirá sus nombres, apellidos y edad, validándolos todos. En el caso de la edad para considerarse una edad válida debe estar comprendida entre 0 y 120. Si es <0 ó >120, volverá a pedirse.

Todos las personas se irán guardando en una lista List (siempre y cuando la persona introducida no exista ya en la lista) y a dicha lista, además, se añadirá otra lista inventada con 5 ó 6 personas mediante el método AddRange.

**AYUDA**: Aquí te muestro como se añadirían 2 personas:

personas.AddRange(new[] {

new Persona

{

Nombre="JUAN",

Apellidos="LOPEZ CRUZ",

Edad=34,

Email="X"

},

new Persona

{

Nombre="JAIME",

Apellidos="LUQUE DE LOS RIOS",

Edad=16,

Email="X"

}

});

Después de realizar la carga por teclado de todos los datos, se ha de imprimir un listado con todos los miembros de la lista y otro con todas aquellas personas que sean mayores de edad (mayores o iguales a 18 años). Para ello, se tendrá que crear una clase Persona que contendrá las propiedades *Nombre, Apellidos, Email y Edad*.

Además, has de tener en cuenta lo siguiente a la hora de guardar los valores en cada propiedad:

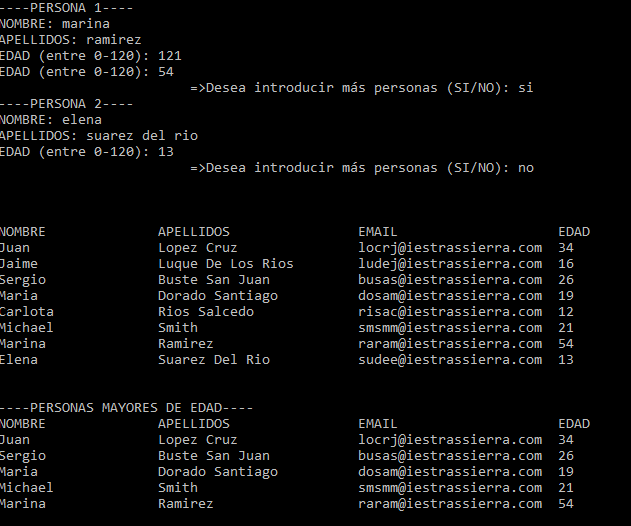
* *Nombre*: se guardará con la primera letra en mayúscula y el resto en minúscula, tanto para nombres simples como compuestos.
* *Apellidos*: exactamente igual que el nombre.
* *Email*: se generará mediante un método privado en la clase Persona, por eso no habrá que pedirlo por teclado Estará formado por las 2 primeras letras del primer apellido, las 2 primeras letras del segundo apellido y la primera letra del nombre. Todo ello seguido de @iestrassierra.com.

Por ejemplo, mi email sería: [fecoe@iestrassierra.com](mailto:fecoe@iestrassierra.com)

Si una persona sólo tuviera un apellido, se duplicaría la primera parte del email. Es decir, si alguien se llama John Smith, su email sería:

[smsmj@iestrassierra.com](mailto:smsmj@iestrassierra.com)

A continuación muestro un ejemplo de ejecución:



**REQUISITOS DE PROGRAMACIÓN:**

* Utiliza los métodos ToLower, Trim, Split, Substring o Replace, y PadRight de la clase String.
* Implementa al menos 2 métodos privados en la clase Persona, uno para generar el email y otro para dar formato a los nombres y apellidos antes de guardarlos. Además, implementa 1 método público que te devuelva todos los valores de las propiedades de la persona que se instancie en un string.
* Implementa al menos 4 métodos (a mí me han salido 7) en el programa principal para reducir y mejorar tu código.
* Trata de usar todos los conceptos vistos en clase pues se valorarán positivamente.
* Pon comentarios.
  1. Crea un programa que se llamará ***2HerenciaSimpleIES***. Hará lo mismo que el programa anterior (sin añadir la lista inventada), es decir, sólo se añadirán las personas introducidas por teclado. Además ahora se pedirá que se especifique si la persona es Alumno o Profesor. En caso de que sea alumno, se pedirá su número de expediente *(>0 y suele ser un entero largo*) y en caso contrario, la materia que imparte. Se creará una lista para alumnos y otra para profesores y al finalizar nos mostrará las dos listas.

Para implementar esto debes crearte 2 nuevas clases:

* ***Alumno***: hereda de Persona. Va a tener la propiedad *Numero de Expediente* y un método público que devolverá todos los valores de las propiedades del alumno instanciado en un string.
* ***Profesor***: hereda de Persona. Va a tener la propiedad *Materia* y un método público que devolverá todos los valores de las propiedades del profesor instanciado en un string.

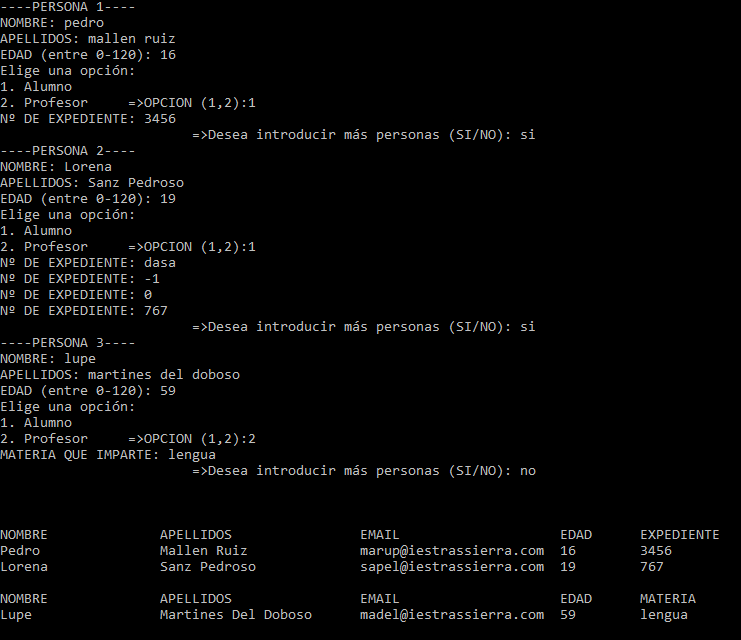
Para poder heredar de estas clases, debes incluir arriba:

using \_1MayoresDeEdad;

Por otro lado, en la clase Program puede que quieras reutilizar algún método de los implementados en el ejercicio 1. En dicho caso, al heredar debes poner lo siguiente:

class Program: \_1MayoresDeEdad.Program

Aquí se muestra un ejemplo de ejecución:



Aunque en el ejemplo no se aprecie, hay que validarlo todo. Es decir, lo mismo que se validó en el ejercicio 1 y además, la opción introducida para indicar si es alumno o profesor, el expediente para el alumno y la materia para el profesor.

**REQUISITOS DE PROGRAMACIÓN:**

* En las clases *Alumno y Profesor*, llama al método público creado en el ejercicio 1 cuando implementes el método que devuelva las propiedades de las instancias de dichas clases.
* En la clase Program, llama al menos a 2 métodos implementados en la clase Program del ejercicio 1.
* Implementa al menos 3 métodos (a mí me han salido 5) en el programa principal para reducir y mejorar tu código.
* Trata de usar todos los conceptos vistos en clase pues se valorarán positivamente.
* Pon comentarios.