Interfaces

Ejercicio 1

Para entender mejor las Interfaces, te pongo un ejemplo sencillo.
Fíjate, sobre todo, en el programa principal para comprender mejor su utilidad.

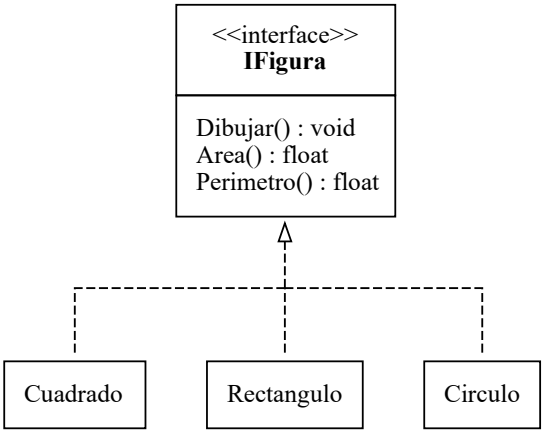
```
interface IVisualiza {
    void Visualiza();
}
class Triangulo : IVisualiza {
    private double @base;
    private double altura;
    public Triangulo(double base_, double altura) {
        @base = base_;
        this.altura = altura;
    }
    private double area  {
        get { return @base * altura / 2; }
    }
    public void Visualiza()    {
        Console.WriteLine($"Base del triángulo: {@base}");
        Console.WriteLine($"Altura del triángulo: {altura}");
        Console.WriteLine($"Área del triángulo: {area}");
    }
}
class Proveedor : IVisualiza {
    private string nombre;
    private string apellidos;
    public Proveedor(string nombre, string apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }
    public void Visualiza() {
        Console.WriteLine($"Nombre: {nombre}");
        Console.WriteLine($"Apellidos: {apellidos}");
    }
}
class EjemploInterfacesApp {
    static void VerDatos(IVisualiza oVisualizable) {
        oVisualizable.Visualiza();
    }
    static void Main() {
        Triangulo t = new Triangulo(10, 5);
        VerDatos(t);
        Proveedor p = new Proveedor("Erik", "Erik otra vez");
        VerDatos(p);
    }
}
```

Crea una librería de interfaces llamada MisInterfaces a la que le irás añadiendo todas las interfaces que implementes en los ejercicios.

Para poder usarla deberás incluirla en tus proyectos.

Ejercicio 2

A partir del siguiente UML crea las clases e Interfaces necesarias para implementar un editor de figuras geométricas. Crea además, un programa principal que te permita probarlo correctamente.



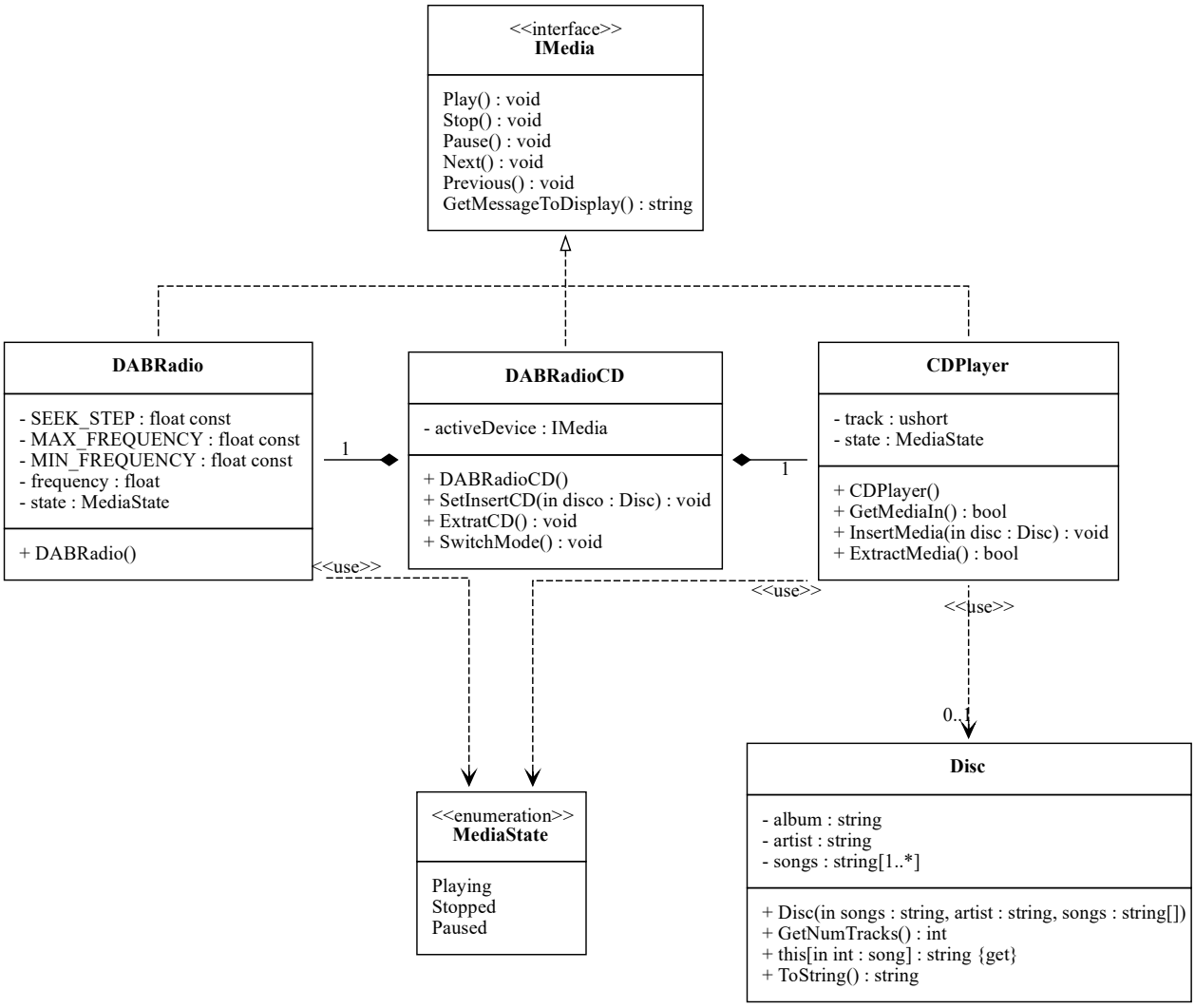
Ejercicio 3

- Tendremos una clase Estudios que implementará los interfaces IEstudios e IVisualiza.
- La interfaz **IEstudios** deberá asegurarse de que las clases implementen un método que diga por pantalla la edad mínima para empezar esos estudios.
- La interfaz **IVisualiza** deberá asegurarse de que las clases implementen un método,que presente por pantalla información.
  (Nota: En nuestro caso relativa a los estudios).
- La clase Estudios no podrá ser instanciada y derivarán de ella las clases Superior, Medio y Elemental.
  - Los estudios superiores, medios y elementales, tendrán como edades de acceso 18, 16 y 12 años respectivamente.
  - Para todos los estudios nos interesará ver en pantalla su nombre y duración.
  - Para los estudios superiores nos interesará el lugar donde se realizan.
  - Para los Medios nos interesa ver por pantalla el nombre de un estudio superior al que den acceso.
- Crear al menos un objeto de cada una de las clases y comprobar su funcionamiento.
- Comprueba si a las clases derivadas de Estudios son a su vez derivadas de **IEstudios** e **IVisualiza**.

Ejercicio 4

Vamos a diseñar las clases para un posible sistema operativo de una radio con DAB (Digital Audio Broadcasting) y un reproductor de CD.

Para ello, seguiremos el modelado propuesto en diagrama de clases del ejemplo teniendo en cuenta las siguientes especificaciones funcionales.



Nota: Recuerda que en C# los Get (Accesores), Set (Mutadores) son Propiedades y que los campos se pueden implementar a través de Porpiedades Autoimplementadas.

Tendremos una clase DABRadioCD que estará compuesta por dos dispositivos un reproductor de CD y un sintonizador DAB.

En el reproductor de CD además podremos tener un Compat Disc ® representado por la clase Disc.

La clase disc tendrá un indizador que me permitirá acceder al título de cada canción y una sobreescritura de ToString que me permitirá ver el nombre del álbum y el artista de la canción.

El reprouctor de CD implementa el interfaz IMedia con la funcionalidad:

- **GetMessageToDisplay:** Propiedad que devuelve un mensaje para el Display del DABRadioCD con el estado del reproductor.
  Devolviendo NO DISC si no hay un disco en un interior. Además, en este caso el resto de opciones de reproducción deberían devolver el mismo mensaje no teniendo efecto.
- **Play**: Que reproducirá el disco desde la pista 1 si el reproductor está parado o desde la pista correspondiente si está pausado.
  Devolviendo MessageToDisplay el estado, la información del CD y la pista que está sonando…
  PLAYING... Album: Thriller Artist: Michael Jackson Track 1 - Wanna Be Startin' Somethin
- **Stop**: Parará el la reproducción. Devolviendo MessageToDisplay...
  STOPPED... Album: Thriller Artist: Michael Jackson
- **Pause**: Pausará el la reproducción si está sonando y la reanudará si está pausada. Si pasa a pausado MessageToDisplay devolverá…
  PAUSED... Album: Thriller Artist: Michael Jackson. Track 1 - Wanna Be Startin' Somethin
- **Next**/**Previous**: Si esta sonando buscará la anterior o siguiente pista a reproducir de forma cíclica. Esto es, si llega al final irá al principio y viceversa. Además, si está pausado empezará a reproducir la nueva pista.

El sintonizador de DAB implementa el interfaz IMedia con la funcionalidad:

- Empezará parada.
- **GetMessageToDisplay:** Propiedad que devuelve un mensaje para el Display del DABRadioCD con el estado de la radio.
- **Play**: Que sintonizará la primera frecuencia de la banda de FM (MIN_FREQUENCY) si estaba apagada (OFF) o continuará con el streaming almacenado en el buffer si estaba pausada.
  Devolviendo MessageToDisplay… HEARING... FM – 87,5 MHz
- **Stop**: Parará el streamig. Devolviendo MessageToDisplay… RADIO OFF
- **Pause**: Pausará el la reproducción si entá sonando la radio y la reanudará si está pausada. Si pasa a pausado se almacenará todo el streaming en un buffer para poder reanudar la emisión donde se quedó y MessageToDisplay devolverá…
  PAUSED - BUFFERING... FM – 87,5 MHz
- **Next**/**Previous**: Si esta sonando moverá el dial la anteior o siguiente frecuencia, con saltos de 0,5 MHz cada vez que se pulse. Si llega al final de la banda (MAX_FREQUENCY) irá al principio de la misma y viceversa. Además, si está pausada empezará a reproducir desde la nueva frecuencia.

Nuestro DABRadioCD implementa el interfaz IMedia con la funcionalidad:

- Para los métodos de IMedia, llamará a los respectivos del dispositivo activo en ese momento.
- **GetMessageToDisplay:** Devolverá una cadena con el dispositivo activo, el estado devuelto por el correspondiente método del dispositivo activo y el menú de opciones para manejar nuestro DABRadioCD.
  MODO: CD
  STATE: PLAYING... Album: Thriller Artist: Michael Jackson. Track 1 - Wanna Be Startin' Somethin
  [1]Play [2]Pause [3]Stop [4]Prev [5]Next [6]Switch [7]Insert CD [8]Extract CD, [ESC]Turn off

- **Insertar un CD**: Devolverá una excepción si ya hay un CD dentro del reproductor. Si no lo hay, pasaremos a modo CD y empezará la reproducción automáticamente.
- **Extraer un CD**: Retirará el CD del reproductor y pasará a modo DAB.
- **Intercambiar modo**: Pasará de CD a DAB o viceversa. Teniendo en cuenta que si pasamos a CD este empezará a reproducir sonde se quedara.

Otras funcionalidades u operaciones sobre los objetos puedes deducirlas del siguiente programa principal de ejemplo y de la propuesta de diagrama de clases UML del ejercicio.

```
public static void Main()
{
    string[] canciones = {
        "Wanna Be Startin' Somethin", "Baby Be Mine", "The Girl Is Mine",
        "Thriller", "Beat It", "Billie Jean", "Human Nature",
        "P.Y.T. (Pretty Young Thing)", "The Lady in My Life"
    };
    Disc thriller = new Disc("Thriller", "Michael Jackson", canciones);
    DABRadioCD radioCD = new DABRadioCD();
    ConsoleKeyInfo tecla = new ConsoleKeyInfo();
    do {
        try  {
            Console.WriteLine(radioCD.MessageToDisplay);
            tecla = Console.ReadKey(true);
            Console.Clear();
            switch (tecla.KeyChar) {
                ...
                case '7':
                    radioCD.InsertCD = thriller;
                    break;
                ...
            }
        }
        catch (Exception e) {
            Console.WriteLine(e.Message);
        }
    } while (tecla.Key != ConsoleKey.Escape);
}
```

Ejercicio 5 Ampliacion

- Crea una Interfaz **IHora** con un indizador que reciba un objeto Hora y devuelva un string.
- La clase Hora se creará a partir de una cadena del tipo "HH:MM" que solo permitirá almacenar desde las 8:00 a las 13:00 horas e internamente almacenará los atributos hora y minutos.
- Esta clase deberá derivar de **IComparable** por lo que tendrás que implementar el método que nos permita comparar las horas y que **usaremos para buscarla en el indizador.**

  Nota: Elimina la redefinición de los operadores == y != para evitar problemas al comparar con null.

- Por otro lado tendremos la clase **Horario** que implementará el Interfaz **IHora**. Haciendo que el indizador devuelva una cadena formateada, con todas las clases que se dan durante la semana para esa hora.

- Para finalizar crea la clase programa que te permita inicializar un horario y mostrarlo, además de usar el indizador implementado por el interfaz.