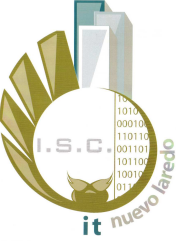


	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 2	
NOMBRE DE LA PRÁCTICA: Ejercicios con interfaces				
MAESTRO: M.C. Bruno López Takeyas		EMAIL: takeyas@itnuevolaredo.edu.mx		


OBJETIVO: El estudiante elaborará diagramas en UML y programas con interfaces
MATERIAL Y EQUIPO NECESARIO: <ul style="list-style-type: none"> • Papel y lápiz • Se recomienda la utilización de software para elaborar diagramas de clases de UML como NClass, el cual puede descargarse de manera gratuita del sitio web http://nclass.sourceforge.net/index.html • Elaborar programas de los ejercicios en C#

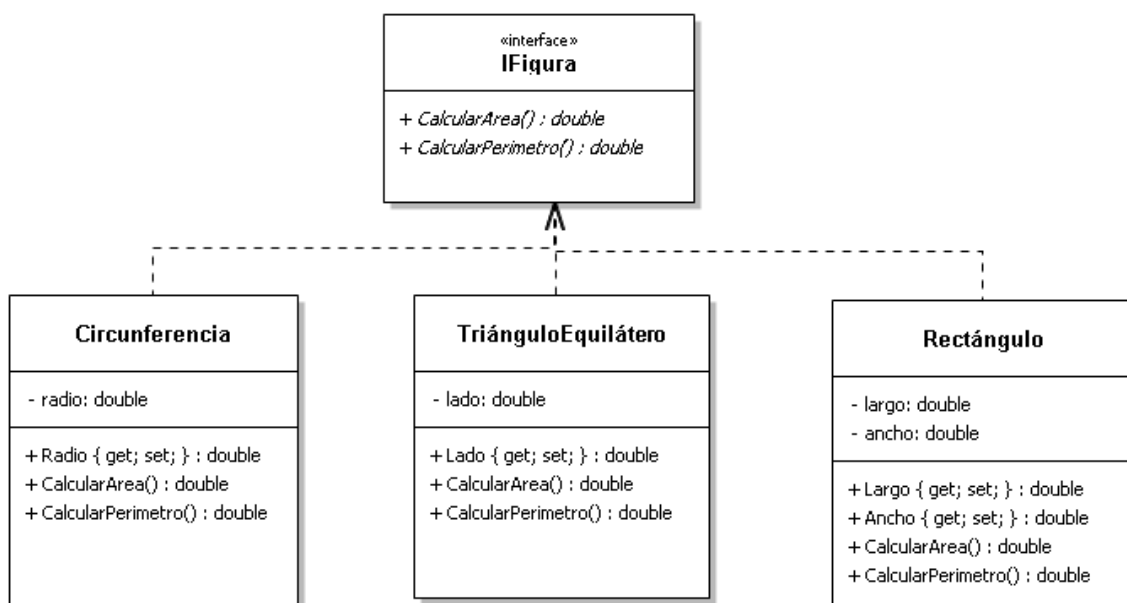
Conteste las siguientes preguntas:

1. ¿Cuáles son las similitudes y diferencias entre una clase abstracta y una interfase?
2. ¿Bajo qué circunstancias recomienda utilizar una interfase? ¿y una clase abstracta?
3. Cuando una clase implementa varias interfaces, ¿se considera herencia múltiple?, ¿Por qué?

Elabore el diagrama de clases en UML y la codificación de un programa para resolver los siguientes problemas:

1. Diseñe un sistema para calcular el área y el perímetro de diversas figuras geométricas utilizando polimorfismo y guiado por el siguiente diagrama en UML:

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 2	
NOMBRE DE LA PRÁCTICA: Ejercicios con interfaces				
MAESTRO: M.C. Bruno López Takeyas			EMAIL: takeyas@itnuevolaredo.edu.mx	

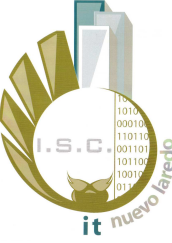


Diseñe una forma como la que se muestra a continuación, de tal manera que se active el o los cuadros de texto de los datos correspondientes a la figura seleccionada con los `radioButtons` y que, al oprimir un botón, se muestre el resultado mediante un `MessageBox`. Cree un objeto según la figura seleccionada, insértele sus datos e invoque su método `CalcularArea()` ó `CalcularPerimetro()` para hacer el cálculo correspondiente y mostrar el resultado.

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 2	
NOMBRE DE LA PRÁCTICA: Ejercicios con interfaces				
MAESTRO: M.C. Bruno López Takeyas		EMAIL: takeyas@itnuevolaredo.edu.mx		



- Realice las modificaciones al diagrama de clases del ejercicio 1 (figuras geométricas) para implementar el método `Equals()` de la interfase `IEquatable` para determinar si dos figuras son iguales.
- Realice las modificaciones al diagrama de clases del ejercicio 1 (figuras geométricas) para implementar el método `CompareTo()` de la interfase `IComparable` para determinar si una figura es mayor que otra.
- Detecte un problema que pueda resolverse mediante el uso de la interfase `IComparable`. Haga el diagrama de clases UML y su codificación.


	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 2	
NOMBRE DE LA PRÁCTICA: Ejercicios con interfaces				
MAESTRO: M.C. Bruno López Takeyas		EMAIL: takeyas@itnuevolaredo.edu.mx		

5. Detecte un problema que pueda resolverse mediante el uso de la interfase `IEquatable`. Haga el diagrama en UML y su codificación.

6. Realizar un diagrama de clases (UML) para la siguiente situación. Asegúrese de expresar herencia, por lo menos una interfaz y una conducta polimórfica:

En una agencia aduanal se va a desarrollar un software para el cobro de honorarios a los diferentes tipos de clientes. Para la agencia es importante que cada cliente tenga registrados: clave, nombre, rfc, dirección fiscal, teléfono y correo electrónico. Los clientes que pagan al contado deben registrar también si pagan en efectivo, con cheque o con transferencia electrónica. Los clientes que son de crédito registran su límite de crédito y el plazo máximo que se les autoriza para pagar. Se sabe que aunque cada cliente puede ser de contado o de crédito, todos deben proporcionar un medio para registrar su pago.

7. Modifique un ejercicio de composición (realizado en la práctica 3.2) para que elimine un objeto de tipo “parte” de la lista incluida en la clase del “todo”. Para lograrlo, la clase del “todo” debe contener un método llamado `Eliminar()` que reciba como parámetro el objeto de tipo “parte” que desea borrar de la lista el cual invoca al método `Remove()` de la colección genérica `List`. Para identificar el objeto que desea eliminar de la lista, el método `Remove()` requiere que se implemente el método `Equals()` de la interfase `IEquatable` en la clase de la “parte”. Haga el diagrama de clases UML y su codificación. Para mayor referencia del uso de `List.Remove()` consulte el sitio web: [https://msdn.microsoft.com/en-us/library/cd666k3e\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/cd666k3e(v=vs.110).aspx)

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES				
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 2		
NOMBRE DE LA PRÁCTICA: Ejercicios con interfaces					
MAESTRO: M.C. Bruno López Takeyas			EMAIL: takeyas@itnuevolaredo.edu.mx		

8. Modifique un ejercicio de composición (realizado en la práctica 3.2) para que ordene de manera ascendente los objetos de tipo “parte” de la lista incluida en la clase del “todo”. Para lograrlo, la clase del “todo” debe contener un método llamado `Ordenar()` el cual invoca al método `Sort()` de la colección genérica `List`. Para el reacomodo de las “partes” de la lista, el método `Sort()` compara los objetos y para ello requiere que se implemente el método `CompareTo()` de la interfase `IComparable` en la clase de la “parte”. Haga el diagrama de clases UML y su codificación.
- Para mayor referencia del uso de `List.Sort()` consulte el sitio web: [https://msdn.microsoft.com/en-us/library/b0zbh7b6\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/b0zbh7b6(v=vs.110).aspx)