
	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.		EMAIL: takeyas@itnuevolaredo.edu.mx		

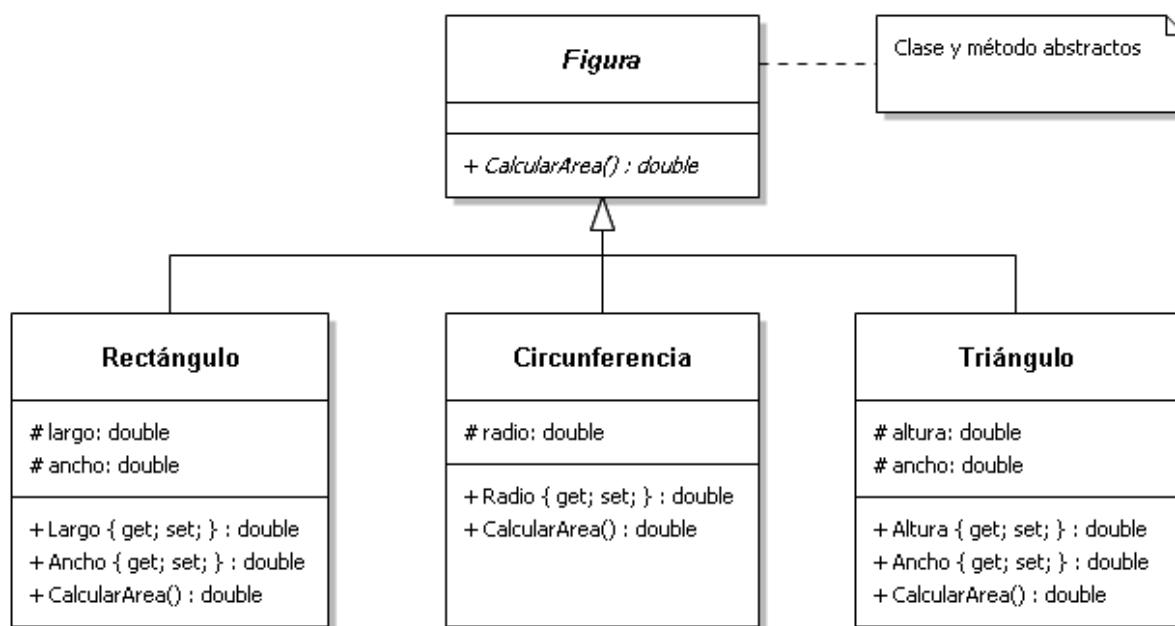
OBJETIVO: El estudiante elaborará diagramas de clases en UML que apliquen relaciones de herencia y polimorfismo



MATERIAL Y EQUIPO NECESARIO:

- 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#

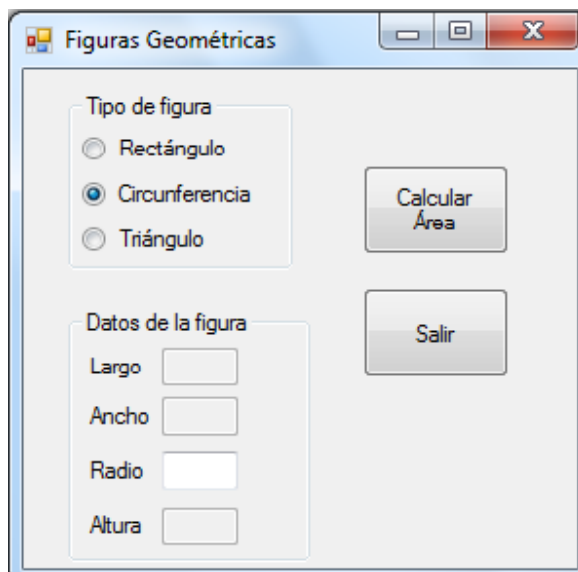
Elabore el diagrama de clases en UML y la codificación de un programa para resolver los siguientes problemas:

- Diseñe un sistema para calcular el área 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: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.		EMAIL: takeyas@itnuevolaredo.edu.mx		

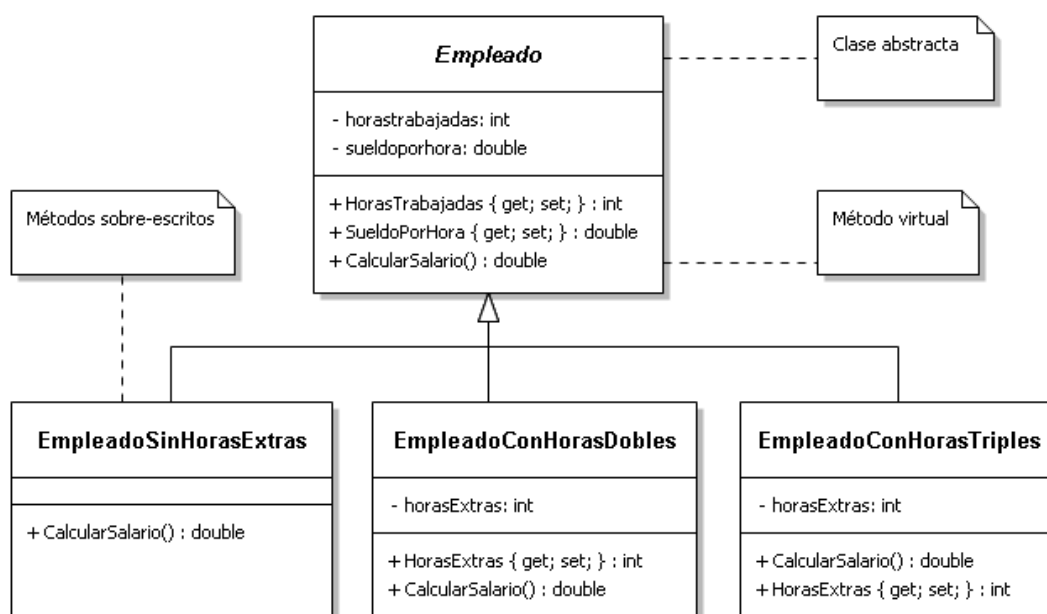
Diseñe una forma como la que se muestra a continuación, de tal manera que se active el o los cuadros de texto de los datos correspondientes a la figura seleccionada con los `radioButtons` y que, al oprimir el botón Calcular área, se muestre el resultado mediante un `messageBox`. Cree un objeto según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



- Una empresa desea un sistema capaz de calcular el salario semanal de sus empleados de acuerdo a la cantidad de horas trabajadas, el sueldo por hora y tomando en cuenta los siguientes criterios:
 - Si las horas trabajadas son más de 40, entonces el excedente se considera hora extra.
 - Si las horas trabajadas están entre el rango de 41 a 45, entonces cada hora extra se paga doble.
 - Si las horas trabajadas son más de 45, entonces cada hora extra se paga triple.

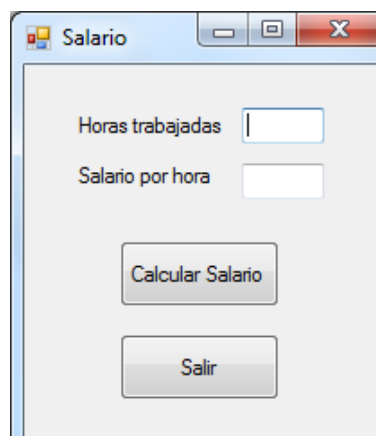
Implemente el sistema de acuerdo al siguiente diagrama de clases en UML:

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.		EMAIL: takeyas@itnuevolaredo.edu.mx		



- Es necesario crear un objeto de acuerdo a la cantidad de horas trabajadas.
- Para el cálculo del salario base (40 horas o menos) utilice el método `CalcularSalario()` de la clase base (`base.CalcularSalario()`).

Diseñe una forma como la que se muestra enseguida:





Salario

Horas trabajadas

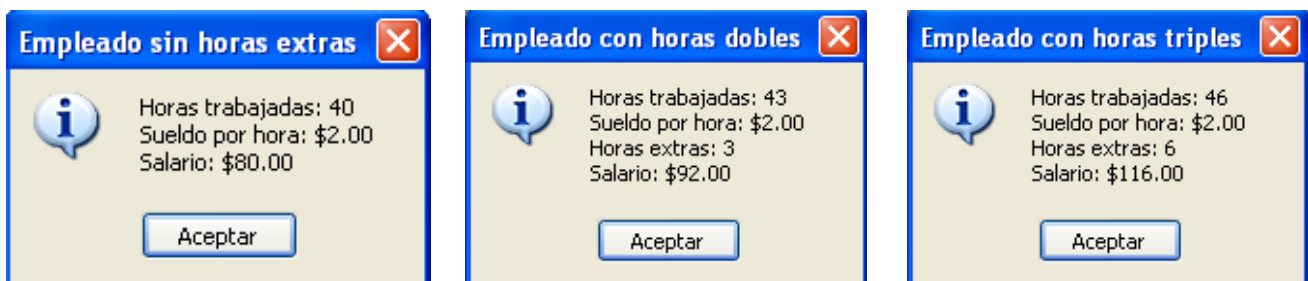
Salario por hora

Calcular Salario

Salir

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.		EMAIL: takeyas@itnuevolaredo.edu.mx		

Considere mostrar los resultados de salida mediante messageBoxes como los que se muestran a continuación:



- Una agencia de renta de vehículos dispone de autobuses y tractores (según el siguiente diagrama en UML).

La renta de autobuses se factura por kilómetros. Debido a esto, los datos de la clase `Autobús` son:

- El precio por kilómetro.
- La cantidad de kilómetros que tiene el autobús cuando se renta.
- La cantidad de kilómetros que tiene el autobús cuando se devuelve.



En cambio, la renta de tractores se factura por días. Debido a esto, los datos de la clase `Tractor` son:

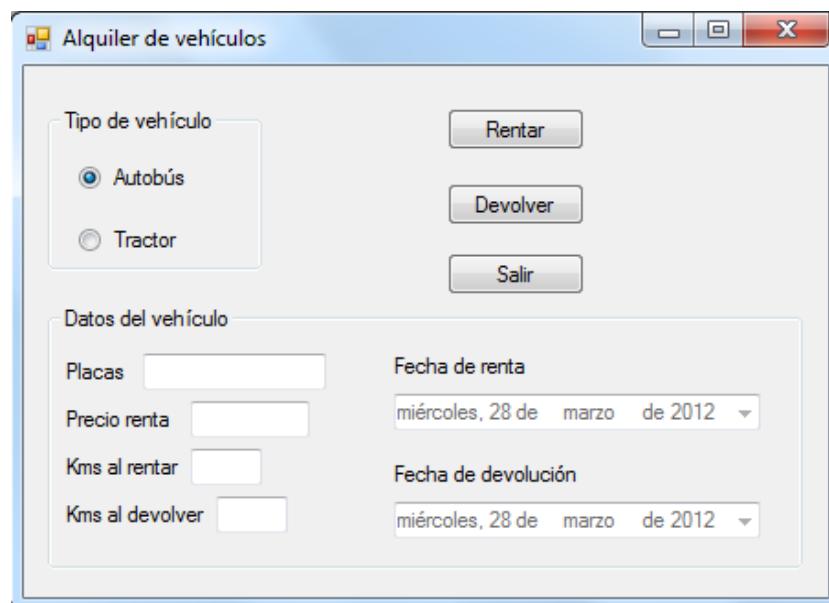
- El precio por día.
- La fecha de su renta (obtenido de la fecha y hora del sistema).
- La fecha de su devolución (obtenido de la fecha y hora del sistema).

Cuando se rente un vehículo, se deben capturar sus placas, sus datos correspondientes (de acuerdo al tipo de vehículo), marcarlo como alquilado (asignando el valor booleano `true` a la propiedad respectiva) y mostrar sus datos mediante un `messageBox`.

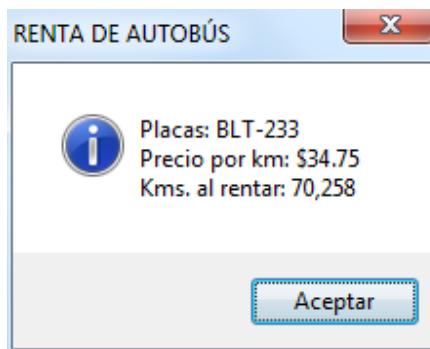
Cuando se devuelve un vehículo, se elimina la marca de alquilado (asignando el valor booleano `false` a la propiedad correspondiente), se calcula el importe a pagar por la renta y se muestran sus datos mediante un `messageBox`.



Diseñe una forma como la que se muestra a continuación, donde active y desactive los controles indicados dependiendo del tipo de vehículo seleccionado:

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.		EMAIL: takeyas@itnuevolaredo.edu.mx		



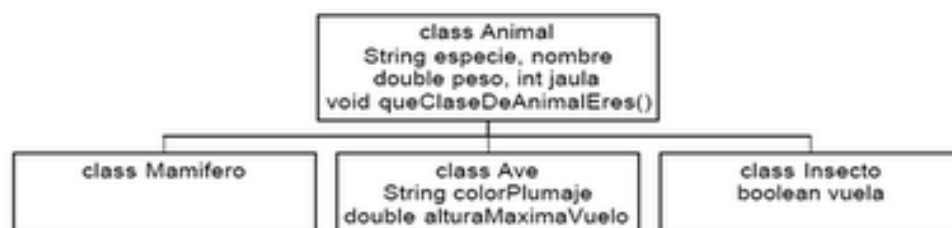
Al oprimir cualquiera de los botones se deben desplegar los datos del vehículo correspondiente según la operación indicada mediante `messageBoxes` como los que se muestran enseguida:




	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.		EMAIL: takeyas@itnuevolaredo.edu.mx		

4. Gestión de un Zoológico.

A continuación se presenta la jerarquía de clases que representa los animales de un posible zoológico:



La clase `Animal` es una clase abstracta con cuatro atributos miembros `protected`:

- Una cadena indicando la **especie** (león, águila, abeja),
- Una cadena indicando el **nombre** del animal concreto
- Un dato numérico real indicando el **peso** en kg.
- Un dato numérico entero indicando el número de **jaula** que se asigna al animal.

Además, la clase `Animal` declara un método virtual `queClaseDeAnimalEres()` que habrá que definir en las clases derivadas.



La clase `Mamifero` no añade nuevos atributos miembro, aunque deberá implementar el método `queClaseDeAnimalEres()`.

La clase `Ave` tiene dos nuevos atributos `protected`:

- Una cadena `colorPlumaje` indicando el color predominante y
- Un dato numérico real indicando la `alturaMaximaVuelo`.

La clase `Insecto` tiene un nuevo atributo miembro `protected` de tipo booleano llamado `vuela` que indica si el insecto vuela o no.

Para realizar este ejercicio se pide lo siguiente:

	INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES			
	MATERIA: Programación Orientada a Objetos (C#)	UNIDAD: 4	PRÁCTICA: 1	
NOMBRE DE LA PRÁCTICA: Ejercicios aplicando polimorfismo				
MAESTRO: Ing. Bruno López Takeyas, M.C.			EMAIL: takeyas@itnuevolaredo.edu.mx	

- I. **Crear las cuatro clases indicadas**, con los correspondientes constructores y sobrecarga de constructores, Como ayuda, se indica que el orden de los argumentos en el constructor parametrizado de la clase base es:

```
public Animal(String especie, String nombre, double peso, int jaula) {...}
```

- II. **Definir** los métodos llamados `queClaseDeAnimalEres()` en cada una de las clases derivadas de `Animal`. Este método no tiene valor de retorno (es *void*) ni argumentos. Debe ser capaz de mostrar por la pantalla la información correspondiente al animal de que se trate (ver el ejemplo), utilizando para ello la información almacenada en las variables miembro.

Ejemplo:

```
Soy un mamífero llamado: xxxxxxxx
de la especie: xxxxxxxx
Peso en Kg: xxx
Estoy en la jaula: xx
```

- III. **Crear una clase ejecutora llamada Zoologico**. Pruebe la jerarquía de clases que implemento, tome en consideración que los animales de carne y hueso se guardan en jaulas.