
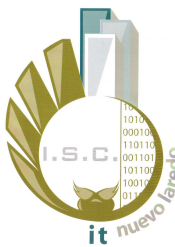

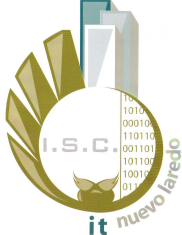


| | | | | | |
|---|---|--|-----------------------|---|--|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  | |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | | |
| MAESTRO: 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 entre clases |
| MATERIAL Y EQUIPO NECESARIO: <ul style="list-style-type: none"> 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:

1. Una compañía editorial produce tanto libros impresos como audio-libros en discos compactos. Diseñe una clase denominada `Publicación` que almacene el título (cadena) y el precio (numérico real) de una publicación. A partir de esta clase, derive dos clases: `Libro` a la cual le agregue el número de páginas (entero) y `CD`, a la cual le agregue el tiempo de reproducción en minutos (numérico real). Cada una de las clases debe tener propiedades para acceder a sus respectivos datos. Elabore un diagrama de clases UML indicando las relaciones de herencia y codifique un sistema mediante el cual se generen instancias de las clases `Libro` y `CD`, donde el usuario capture sus datos y se inserten en los respectivos objetos. Diseñe la forma que se muestra a continuación.

| | | | | |
|---|---|--|-----------------------|---|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | |
| MAESTRO: Ing. Bruno López Takeyas, M.C. | | EMAIL: takeyas@itnuevolaredo.edu.mx | | |


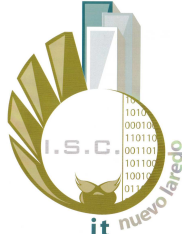


Cuando el usuario seleccione el libro impreso, entonces debe activarse el cuadro de texto para capturar el número de páginas y desactivar el cuadro de texto del tiempo en minutos; mientras que si el usuario selecciona el audio-libro, entonces debe activarse el cuadro de texto del tiempo en minutos y desactivar el cuadro de texto del número de páginas.

Para limpiar los cuadros de texto, se recomienda el siguiente código:

```
foreach(Control x in groupBox2.Controls)
    if(x is TextBox)
        x.Text=" " ;
```

2. Modifique el ejercicio 1 e implemente la sobrescritura del método `ToString()` para mostrar los datos.
3. Modifique el ejercicio 1 para evitar crear instancias de la clase base (utilice una clase abstracta).


| | | | | | |
|---|---|--|-----------------------|---|--|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  | |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | | |
| MAESTRO: Ing. Bruno López Takeyas, M.C. | | EMAIL: takeyas@itnuevolaredo.edu.mx | | | |


4. Modifique lo que considere necesario en el ejercicio 1 y agregue un arreglo de celdas de tipo numérico real para almacenar las ventas en dinero de cada publicación durante los 3 últimos meses.
5. Una agencia vendedora de autos desea un sistema computacional para administrar los datos de sus vehículos y clasificarlos por tipo. Todos los autos tienen los siguientes datos:
 - Número de serie del motor.
 - Marca.
 - Año.
 - Precio.

Los vehículos se clasifican en autos compactos, autos de lujo, camionetas y vagonetas. Para los autos y vagonetas, también es importante almacenar la cantidad de pasajeros; mientras que para las camionetas se debe controlar la capacidad de carga en kgs. y la cantidad de ejes y de rodadas. Modele este sistema con diagramas de clases en UML e instancie cada una de las clases, asignándole datos mediante sus respectivas propiedades. Agregue un constructor con parámetros a cada clase para inicializar sus datos e invoque el constructor de la clase base desde el constructor de cada clase derivada (no utilice constructores *default*). Implemente la sobrescritura del método `ToString()` para mostrar los datos de cada tipo de auto.

NOTA: No se permiten componentes duplicados en las clases ni clases vacías (sin elementos).

6. Un equipo de futbol debe controlar las estadísticas de sus jugadores y para ello requiere un sistema computacional orientado a objetos con relaciones de herencia. Los datos de cada jugador son el número de uniforme, nombre y su posición (portero, defensa, medio y delantero). Además, el equipo desea almacenar la cantidad de goles anotados por cada jugador (excepto el portero) y los minutos jugados. Defina las clases con sus relaciones de herencia de la manera más apropiada para modelar este sistema e impleméntelas a través de un proyecto de formas como la siguiente:

| | | | | |
|---|---|--|-----------------------|---|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | |
| MAESTRO: Ing. Bruno López Takeyas, M.C. | | EMAIL: takeyas@itnuevolaredo.edu.mx | | |



Cuando el usuario seleccione un portero, entonces debe desactivarse el cuadro de texto de los goles anotados. Instancie cada una de las clases, capture sus datos para mostrarlos posteriormente. Agregue un constructor con parámetros a cada clase para inicializar sus datos e invoque el constructor de la clase base desde el constructor de cada clase derivada (no utilice constructores *default*). Implemente la sobrescritura del método `ToString()` para mostrar los datos de cada tipo de jugador.

NOTA: No se permiten componentes duplicados en las clases ni clases vacías (sin elementos).

- Un equipo de beisbol se compone de tres tipos de jugadores: *pitchers*, jugadores de posición y bateadores designados. Los datos de cada jugador son: el número de uniforme, nombre y su posición. En esta liga, los *pitchers* lanzan, participan a la defensiva pero no batean. Los bateadores designados no lanzan y no participan a la defensiva (sólo batean). Los jugadores de posición no lanzan, pero participan a la defensiva y batean. Modele un sistema orientado a objetos con clases heredadas con el que controle los ponches recetados por los *pitchers*, los hits bateados por los otros jugadores y los errores cometidos por quienes participan a la defensiva.

Diseñe la siguiente forma, de tal manera que cuando el usuario seleccione un *pitcher*, solamente se activen los cuadros de texto de ponches y errores. El jugador de posición

| | | | | |
|---|---|--|-----------------------|---|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | |
| MAESTRO: Ing. Bruno López Takeyas, M.C. | | EMAIL: takeyas@itnuevolaredo.edu.mx | | |

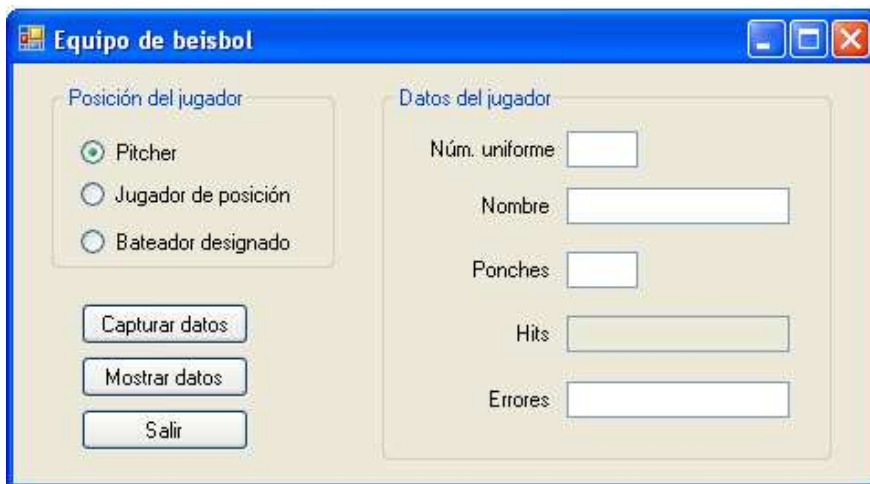
solamente debe activar los cuadros de texto de los hits y errores; mientras que el bateador designado debe activar solamente el de los hits.

Agregue un constructor con parámetros (no utilice el constructor *default*) a cada clase para inicializar sus datos e invoque el constructor de la clase base desde el constructor de cada clase derivada.


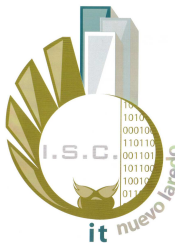
Instancie un objeto de cada posición, capture sus datos y muéstrellos en pantalla. Implemente la sobrescritura del método `ToString()` para mostrar los datos de cada tipo de jugador.

NOTAS:

- No se permiten componentes duplicados en las clases ni clases vacías (sin elementos).
- No utilice herencia múltiple (recuerde que no puede implementarse en C# .NET)



- Para una empresa es importante controlar los datos de las personas relacionadas a ella, como lo son sus clientes y sus empleados. Los datos de cada persona son el nombre y su domicilio; sin embargo, también es importante administrar el límite de crédito de sus clientes y el salario de sus empleados. Modele este sistema diseñando el diagrama de clases en UML y estableciendo las relaciones de herencia pertinentes. Agregue un constructor con parámetros a cada clase para inicializar sus datos e invoque el constructor de la clase base desde el constructor de cada clase derivada (no utilice

| | | | | |
|---|---|--|-----------------------|---|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | |
| MAESTRO: Ing. Bruno López Takeyas, M.C. | | EMAIL: takeyas@itnuevolaredo.edu.mx | | |

constructores *default*). Instancie cada una de las clases y asígnele sus respectivos datos mediante sus propiedades. Posteriormente despliegue los datos suministrados.

Implemente la sobrescritura del método `ToString()` para mostrar los datos de cada tipo de objeto.

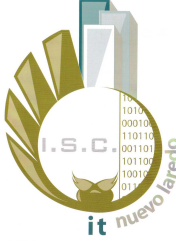
NOTA: No se permiten componentes duplicados en las clases ni clases vacías (sin elementos).

9. Realice la modificación que considere apropiada en el ejercicio anterior para evitar generar instancias de la clase base.
10. Diseñe una jerarquía para las clases Cuadrilátero, Trapecio, Rectángulo y Cuadrado. Use la clase Cuadrilátero como la clase base de la jerarquía. Los datos privados de la clase base deben ser las coordenadas x-y de los cuatro vértices de la figura y debe contener un método abstracto para calcular el área. Agregue un constructor a cada clase para inicializar sus datos e invoque el constructor de la clase base desde el constructor de cada clase derivada. Escriba un programa que instancie objetos de cada una de las clases y calcule el área correspondiente. Investigue las fórmulas.
11. Elabore el diseño de una jerarquía de clases para modelar los alimentos de un restaurante, en el que se sirven:

- Platillo fuerte
- Ensalada
- Sopa
- Postre
- Bebida

El sistema debe tomar en cuenta las siguientes consideraciones:

- Todos los alimentos tienen como datos el nombre y su precio.
- El pollo puede ser un ingrediente del platillo fuerte, la ensalada y la sopa.
- La carne puede ser un ingrediente solamente del platillo fuerte.
- Tanto el platillo fuerte, la ensalada y la sopa pueden contener sal.
- El azúcar puede ser un ingrediente del postre y la bebida.
- Para comer el platillo fuerte y la ensalada se requiere uso de tenedor y/o cuchillo.
- Para degustar la sopa, se requiere cuchara.
- El postre se puede ingerir mediante cuchara o tenedor.

| | | | | |
|---|---|--|-----------------------|---|
|  | INSTITUTO TECNOLÓGICO DE NUEVO LAREDO ING. EN SISTEMAS COMPUTACIONALES | | |  |
| | MATERIA: Programación Orientada a Objetos (C#) | UNIDAD: 3 | PRÁCTICA: 1 | |
| NOMBRE DE LA PRÁCTICA: Ejercicios aplicando herencia | | | | |
| MAESTRO: Ing. Bruno López Takeyas, M.C. | | EMAIL: takeyas@itnuevolaredo.edu.mx | | |

- La bebida se sirve en un vaso y/o se puede utilizar un popote.

Utilice variables booleanas para identificar la presencia de un ingrediente y el uso de un determinado cubierto para cada alimento. Capture dichas variables a través de *checkboxes* activando solamente las necesarias de acuerdo al tipo de alimento.

Agregue un constructor con parámetros a cada clase para inicializar sus datos e invoque el constructor de la clase base desde el constructor de cada clase derivada (no utilice constructores *default*).

Implemente la sobrescritura del método `ToString()` para mostrar los datos de cada tipo de objeto.

NOTA: No se permiten componentes duplicados en las clases ni clases vacías (sin elementos).

12. Implemente un sistema tomando como referencia el siguiente diagrama de clases en UML:

