Universidad Tecnológica Nacional
Facultad Regional Avellaneda

Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Programación II

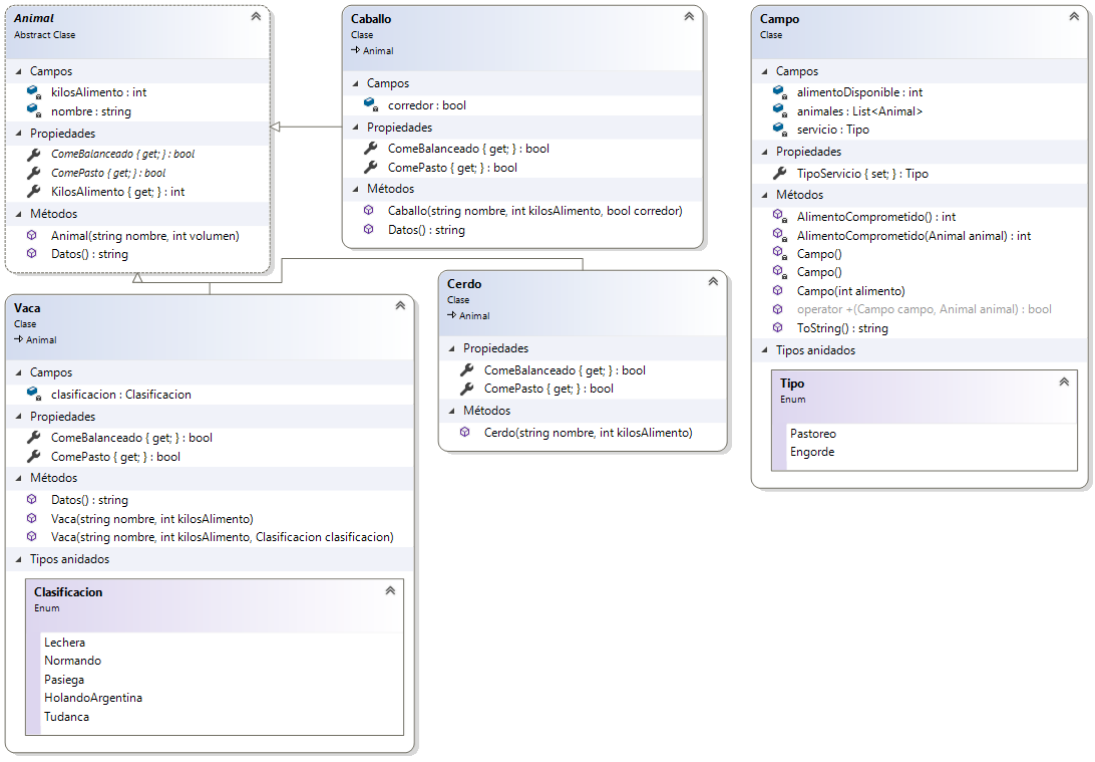
Apellido:					Fecha:		07-05-2020			
Nombre:					Docente(2):		Dávila-Cerizza-Rodríguez			
División:	2°C				Nota(2):					
Legajo:					Firma(2):					
Instancia(1):	**PP**	X	**RPP**		**SP**		**RSP**		**FIN**	

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**). Marque con una cruz.
(2) Campos a ser completados por el docente.

IMPORTANTE:

- **2 (dos) errores en el mismo tema anulan su puntaje.**
- La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
- Colocar sus datos personales en el nombre de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2D. No sé corregirán proyectos que no sea identificable su autor.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.
- Aplicar los principios de los 4 pilares de la POO.

1. Crear un proyecto del tipo Biblioteca de Clases y colocar el siguiente esquema de clases:



2. Clase **Animal**:
    a. Los métodos y propiedades marcados en cursiva son abstractos.
    b. Datos retornará los datos de la Animal utilizando StringBuilder y String.Format. Completar las palabras en mayúscula con los datos de cada Animal según corresponda:
        NOMBRE come TAMANIOkg
        Consume balanceado SI/NO
        Consume pasto SI/NO
3. Clase **Vaca**:
    a. Su Datos incorporará el concepto "`Clasificada como: XXXXX`".
    b. Come pasto y come balanceado.
4. Clase **Caballo**:
    a. Su Datos incorporará el concepto "`Es de carreras: SI/NO`".
    b. Come pasto y no come balanceado.
5. Clase **Cerdo**:
    a. Sólo come balanceado.
6. Clase **Campo**:
    a. La lista se inicializará en el constructor privado.
    b. El Tipo de campo será estático.
    c. Servicio se inicializará en el constructor estático con el valor Engorde.
    d. El operador + agregará siempre y cuando *disponga de alimento* suficiente para agregar un nuevo Animal (alimento disponible + kilos de alimento del Animal).
    e. AlimentoComprometido(Animal) retornará el total de alimento que consumen sus animales + el que necesita el animal recibido como parámetro.
    f. AlimentoComprometido retornará el espacio total de alimento que consumen sus animales. Utilizar el concepto de Polimorfismo.
    g. Sobreescribir el método ToString para que retorne los datos del Campo:
        Servicio del campo: PASTOREO/ENGORDE
        Alimento comprometido XXX de XXX
        LISTA DE ANIMALES

7. Clase **FrmPrincipal**:
    a. El formulario debe iniciar centrado en la pantalla con el siguiente formato:

a. Colocar el BackColor en un tono de rosa suave.
b. Hacer que no tenga bordes con el estilo de Form Border.
c. El nombre del formulario debe ser Campo.
d. Agregar un atributo del tipo Campo al formulario.
e. Agregar el siguiente código al evento *Load* del formulario. Reemplazar el Console.WriteLine por un Message Box con ícono de exclamación y sólo el botón OK:

```
this.campo = new Campo(1500);
Campo.TipoServicio = Campo.Tipo.Pastoreo;
bool pudo = this.campo + new Cerdo("Pegy", 300);
pudo = this.campo + new Cerdo("Babe", 250);
pudo = this.campo + new Vaca("Rosalinda", 450, Vaca.Clasificacion.Lechera);
pudo = this.campo + new Vaca("Lola", 325);
pudo = this.campo + new Caballo("Secretariat", 175, true);
if (!(this.campo + new Caballo("BoJack", 1, false)))
{
   Console.WriteLine("ERROR!");
}
```

f. Al presionar el botón *Ver datos* se mostrará en el RichTextBox *rtbSalidaDeTest* los datos del campo.