EJERCICIOS MÓDULO 10 (Herencia)

Problema 1

Crea un proyecto de nombre construcciones, que contenga las siguientes clases:

a) Diseña una clase base llamada Construcción que almacene los siguientes datos:

- el número de plantas que tiene un edificio,
- el número de habitaciones
- su superficie total.
- Debe tener un constructor por defecto y un constructor con 3 parámetros.
- Debe Sobreescribir la función ToString de la clase Object

```
class Construccion{
    private int Plantas;
    private int Habitaciones;
    private int Superficie;
    public Construccion() {



    }
    public Construccion(int pPla, int pHab, int pSup) {



    }

    public int get          {                }
    public int get              {                    }
    public int get            {                  }

    public override string ToString() {




    }
}
```

b) Crea una clase derivada llamada Casa que herede de Construcción y que almacene:

- el número de dormitorios
- el número de baños.
- Debe tener un constructor con 5 parámetros, de manera que con los tres primeros inicialice los datos heredados de Construccion.
- Debe Sobreescribir la función ToString de la clase Object

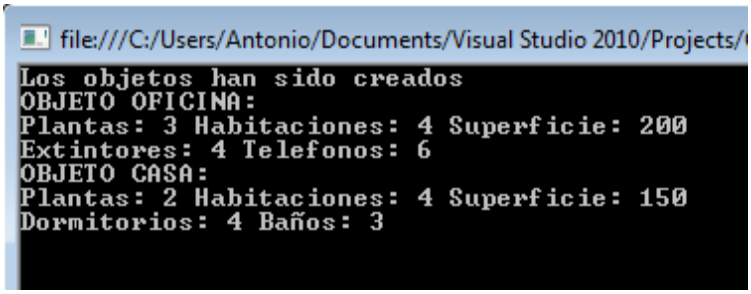
```
class Casa : Construccion
{
    int Dormitorios;
    int Banos;
    public Casa(int nPla, int nHab, int Sup, int nDorm, int nBano)

    {


    }

    public int get             { }
    public int get        {          }

    public override string ToString() {




    }
}
```

Crea otra clase derivada llamada Oficina que herede de Construcción y que almacene además:

- el número de extintores
- el número de teléfonos
- Debe tener un constructor con 5 parámetros, de manera que con los tres primeros inicialice los datos heredados de Construccion.
- Debe Sobreescribir la función ToString de la clase Object

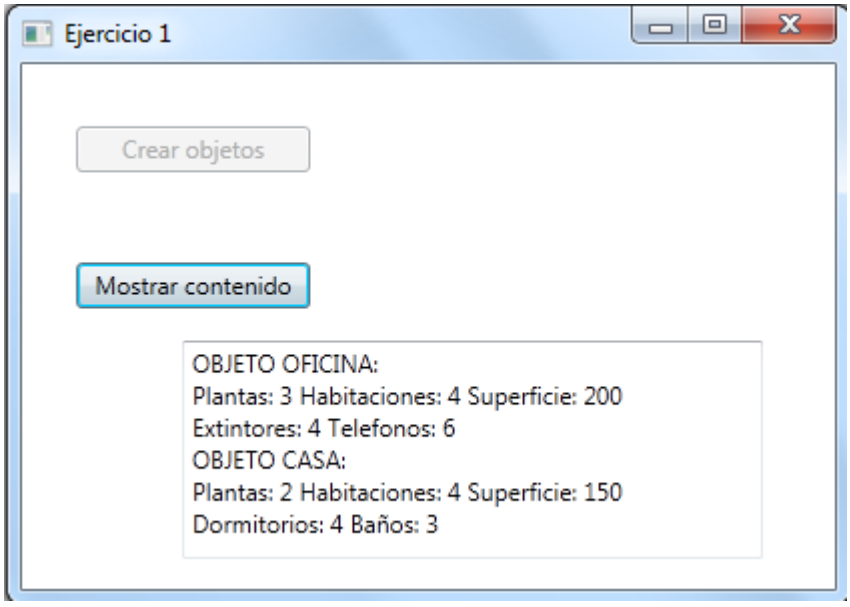
```
class Oficina:Construccion
{
    int Extintores;
    int Telefonos;
    public Oficina (int pPlan, int pHab, int pSup, int pExt, int pTel)

    {


    }
    public string get            {                          }
    public string get           {                         }

    public override string ToString() {




    }
}
```

Construye un pequeño programa para crear un par de objetos de cada tipo (Oficina y Casa) y mostrar sus valores. El resultado debe ser similar al siguiente:



Problema 2

Aprovechando las clases del ejercicio anterior, crea un proyecto en WPF de manera que muestre la siguiente interficie:



Problema 3

a) Crea una clase *Animal* que contenga los atributos

- *nombre científico*
- *alimentación* (herbívoro, omnívoro o carnívoro)
- *velocidad*

Implementa los siguientes métodos:

- Constructor de 3 parámetros
- *Correr:* en función del atributo *velocidad*, indica si corre mucho o poco
- *Saltar:* en función del atributo *velocidad*, indica si corre mucho o poco
- *ToString:* que devuelva un string con el contenido de los atributos de la clase

b) Crea las clases *Perro* y *Gato* que hereden de ella y establece un atributo propio para cada una de ellas: por ejemplo para el Perro el atributo Color, y para el Gato el atributo Raza. Sobreescribe la

función ToString de forma que imprima los elementos de la clase padre junto con los nuevos de la clase derivada.

c) Crea un programa con la siguiente declaración de variables.

```
static void Main(string[] args)
{
    Perro oPerro = new Perro("Can", "omnivoro", 150, "negro");
    Gato oGato = new Gato("Cat", "omnivoro", 40, "persa");
    Animal oAnim1 = oPerro;
    Animal oAnim2 = oGato;
```

A continuación se debe de llamar a los métodos ToString, Correr, Saltar, getColor y GetRaza con las variables oAnim1 y oAnim2 creadas anteriormente. ¿Qué problemas ocurren?

Problema 4

a) Crea una estructura de clases que incluya *Persona* como clase base y *Profesor, Director* y *Alumno* como clases derivadas. Define lo necesario para que podamos visualizar la información de cada tipo de objeto a través de este método común (definido en la clase *Persona*):

public String getInfo()

El método se deberá comportar de manera diferente según de qué tipo sea el objeto al que pertenece, es decir:

- si invocamos a *getInfo()* en un objeto de tipo *Alumno*, debe mostrar:
  - nombre
  - apellidos
  - notas matemáticas
  - notas lengua
  - notas inglés
- si invocamos *getInfo()* en un objeto de tipo *Profesor*:
  - nombre
  - apellidos
  - aula_asignada
  - esTutor (Si / No)
- si invocamos *getInfo()* en un objeto de tipo *Director*:
  - nombre
  - apellidos
  - colegio
  - antigüedad

b) Crea un programa que cree las siguientes personas:

```
static void Main(string[] args)
{
    Profesor oProfe = new Profesor("Pepe", "Ruiz Ruiz", 3, true);
    Persona oPer1 = oProfe;
    Alumno oAlum = new Alumno("Luis", "Lopez Lopez", 4.5, 6, 7);
    Persona oPer2 = oAlum;
    Director oDire = new Director("Jose", "Perez Perez", "ESPAI", 10);
    Persona oPer3 = oDire;
```

Se debe llamar a la función getInfo() de cada una de ellas.

Problema 5

Vamos a crear un par de métodos que extiendan a la clase *String*. El primero de ellos recibirá una frase y dos palabras y deberá sustituir todas las ocurrencias de la primera palabra en la frase por la segunda palabra. El otro método nos contará el número de vocales que tiene una palabra o frase. Genera unas variables de prueba para testear estas nuevas utilidades.