Profesores:

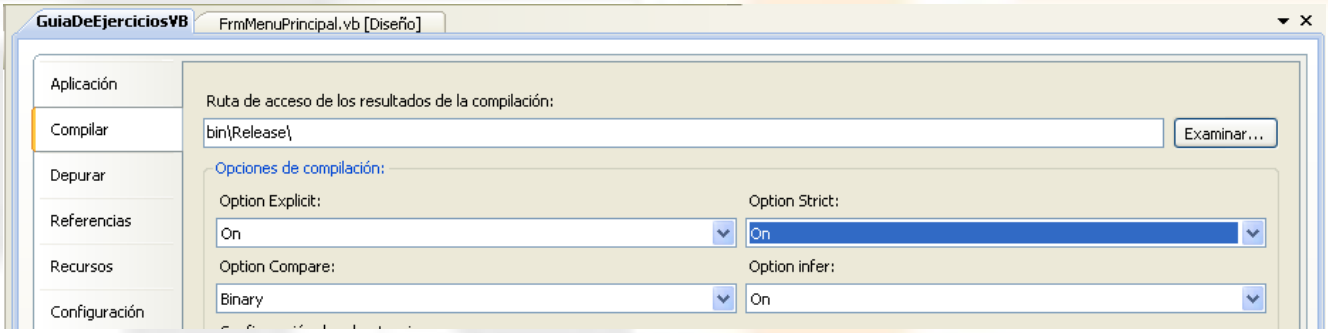
Maximiliano Neiner
Octavio Villegas

Nota:
Esta guía forma parte del trabajo práctico número uno (TP Nro. 1), que estará separado en dos entregas (una antes del primer parcial y la otra antes del segundo parcial), las fechas de entrega serán publicadas por el profesor de Laboratorio III a su debido tiempo.

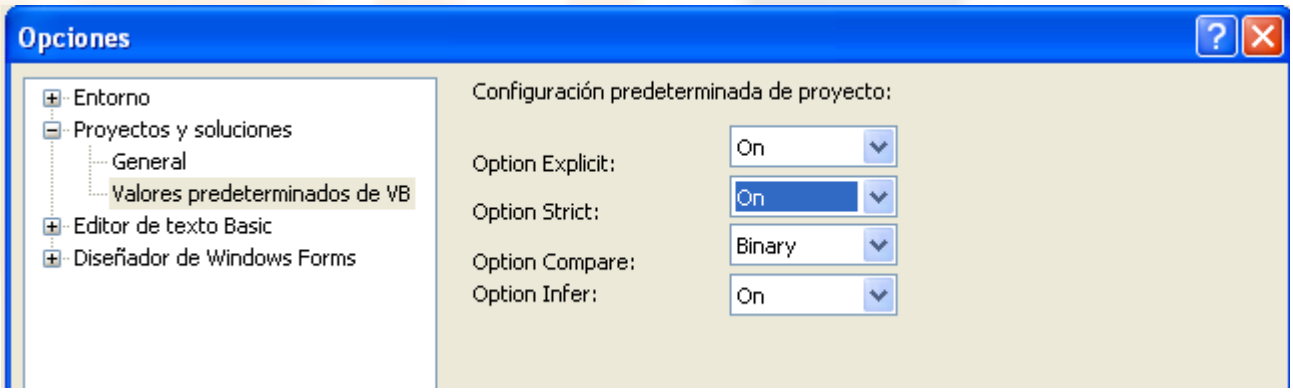
Antes de empezar tener en cuenta los siguientes puntos:

1. Todos los proyectos tienen que tener la opción *Option Strict* en **ON**, para esto tenemos dos formas de hacerlo:

   I. Ir al menú Proyecto -> Propiedades y en la solapa "Compilar" colocar la opción en ON.



   II. La segunda opción es ir al menú Herramientas -> Opciones y en el árbol de vista seleccionar Proyectos y soluciones -> Valores predeterminados de VB



Nota:
Tener en cuenta que de esta forma todos nuestros proyectos quedarán con estas opciones, si se elige la primera forma sólo se cambiarán los valores para ese proyecto, debiendo repetir el mismo procedimiento cada vez que se cree un nuevo proyecto.

2. Esta guía será entregada en un CD con una carpeta nombrada con su nombre punto (.) su apellido punto (.) su división. El siguiente ejemplo corresponde a un alumno de 3 C llamado Juan Pérez, la carpeta dentro del CD quedará:
   Juan.Perez.3C.
   Dentro de esta carpeta estarán las soluciones/proyectos correspondientes a cada ejercicio de la guía. Las soluciones/proyectos se nombrarán de la siguiente forma: Aplicación punto (.) ##, dónde ## será el número del ejercicio. El siguiente ejemplo corresponde al nombre de la solución/proyecto del ejercicio número 3:
   Aplicación.03.

3. Se deben utilizar los prefijos para cada elemento dentro de las aplicaciones de tipo Windows Form como las convenciones al nombrar clases, métodos, atributos, etc., ya que se tendrán en cuenta al momento de evaluar este trabajo práctico.

4. Todas las clases intervinientes en los ejercicios de tipo WindowsForm, deben ser creadas en proyectos de tipo Biblioteca de clases.

5. El punto de entrada de todas nuestras aplicaciones debe ser un "Sub Main", que se encontrara en un modulo público. Ej.:

```
Try
      Dim frmInicio As New FrmPrincipal

      Application.Run(frmInicio)
  Catch ex As Exception
      MessageBox.Show("Error: " & ex.Message, "Error  desconiciodo")
  Finally
End Try
```

6. Es obligatorio el uso del bloque `Try Catch Finally` en todas las instruciones que puedan generar excepciones.

Parte 1 - Ejercicios con Consola

Aplicación Nº 1

Se requiere crear la clase abstracta SerHumano que posea los siguientes atributos privados:

- _nombre (String)
- _peso (Single)
- _altura (Single)
- _sexo (String)

Además tendrá los siguientes métodos:

- Comer(String)
- Dormir (método abstracto)

A partir de la clase SerHumano, se pide:
Crear dos clases (que hereden de la anterior) llamadas Gerente y Empleado.
Cada una de dichas clases poseerá atributos y métodos característicos (por ejemplo, la clase Empleado tendrá como atributo sueldo y como método Trabajar).
Generar, en el método Main, las sentencias necesarias para probar los miembros de ambas clases.

Aplicación Nº 2

Realizar la clase Mascota que posea como atributos **protegidos**:

- _nombre (String)
- _edad (Integer)
- _raza (enum eRaza)

Y los siguientes métodos Públicos:

- Sub Jugar (conQueJuego As String) (virtual)
- Sub Mostrar () (Abstracto)

Contructores:

- Sub New (Nombre As String, edad As Integer, queRaza As Raza)
- Sub New (Nombre As String, queRaza As Raza)

Crear dos clases **(Perro y Gato)** que hereden de **Mascota** y que posean *Ladra* (Booleano) y *Maulla* (Booleano) como atributos, cada uno en la clase correspondiente. Cada una de estas clases deberá implementar el método Mostrar para poder visualizar desde la Consola todos sus atributos.
De la clase perro no se podrá generar herencia.
Ambas clases deberán tener un método **de Clase** que reciba un Perro o un Gato, según la clase, y retorne un enumerado con la raza.
Se desea construir la clase **Guardería** que tendrá un atributo **Privado** de tipo lista genérica de Mascotas y además dos atributos **Privados** más: _precioPerro (Double) y _precioGato (Double), estos dos últimos se inicializarán desde el constructor.
Realizar propiedades de sólo lectura para estos atributos.
Se sobrecargará al operador "**+**" para permitir agregar una mascota a la guardería, retornando una nueva guardería ej. :

Miguarderia = Miguarderia + UnaMascota

Los métodos públicos que tendrá la Guardería son:

- **MostrarTotalFacturado**: devolverá la ganancia de la guardería (Single), dicho método tendrá una sobrecarga que reciba como parámetro la enumeración eMascota (con Perro y Gato como enumerados) y retornará la ganancia de la Guardería por tipo de Mascota.

- **IngresarMascota**: recibirá como único parámetro una Mascota y la agregará a la lista genérica de dicho objeto.
- **ToString**: deberá devolver en un String los datos de todas las mascotas que tiene la Guardería, este método utilizará un objeto de tipo **StringBuilder**.

Crear una interface que se llame **"IGuardar"** que contenga los siguientes métodos:

Este método guardará, sin sobrescribir, los datos de las mascotas que tengo en mi guardería.
```
Function ImprimirDatosListaMascotas() As Boolean
```

Serializa la lista de mascotas.
```
Function SerializarListaMascota(ByVal ruta As String) As Boolean
```

Crear una interface que se llame **"ICargar"** que contenga los siguientes métodos:

Este método, leerá de un archivo de texto, los datos de las mascotas que están en él para luego guardarlos en un StringBuilder y mostrarlos por Consola.
```
Function TraerDatosMascota() As Boolean
```

Deserializa la lista de mascotas.
```
Function DeserealizarListaMascota() As Boolean
```

Nota: Las Interfaces se implementarán sólo en la clase Guardería.

```
Module Module1

    Sub Main()

        //Crear tres gatos.
        //Crear tres perros.
        //Crear una Guardería.
        //Ingresar las mascotas a la guardería.
        // Mostrar el total Facturado.
        // Mostrar el total Facturado por Gato.
        // Mostrar el total Facturado por Perro.
        // Guarderia.ImprimirDatosListaMascotas
        // Guarderia.SerializarListaMascotas
        // Guarderia.DeserializarListaMascotas
        // Guarderia.TraerDatosListaMascotas

        //MOSTRAR POR CONSOLA TODOS LOS ATRIBUTOS DE LAS MASCOTAS INGRESADAS EN LA
        GUARDERIA

    End Sub

End Module
```

DIAGRAMA DE CLASES