UNIVERSIDAD TECNOLÓGICA NACIONAL

Tecnicatura Superior en Programación

Programación II

Laboratorio de Computación II

Trabajo Práctico

Fecha de entrega: a definir por cada docente.

Formato de entrega: Se deberá entregar en formato impreso (tamaño A4 o Carta, y las hojas no deberán estar sueltas, utilizar folio o carpeta) y en formato digital (CD o DVD rotulado con los datos del alumno). En ambos casos la información que se deberá incluir será la siguiente: Apellido y Nombre, Legajo, Materia, dirección de correo electrónico.

Formato impreso: código fuente solicitado de los 2 proyectos.

Formato digital: En caso de utilizar la versión Express del Visual Studio, código fuente de los 2 proyectos.

En caso de no utilizar la versión Express, código fuente de los 2 proyectos, dentro de una única Solución.

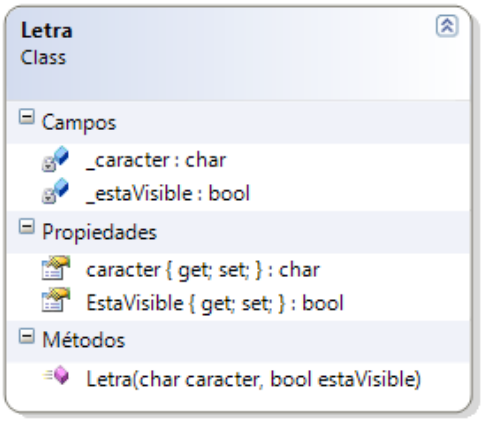
Se podrá entregar el código fuente en versiones **2008**, **2010** y **2012** del Visual Studio.

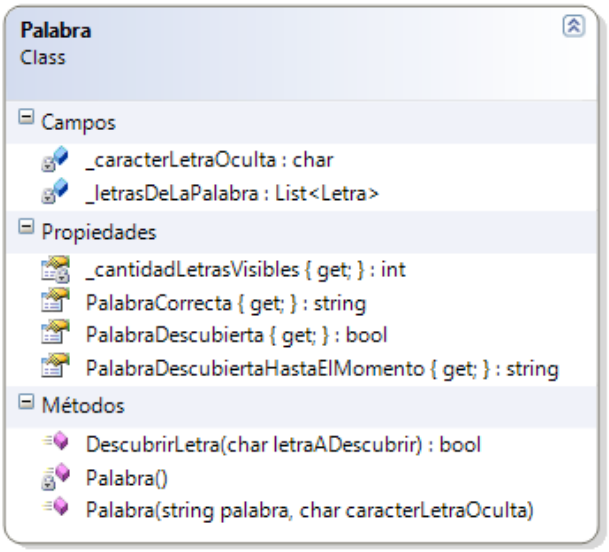
El incumplimiento de los ítems detallados anteriormente, implicará la no aceptación del trabajo práctico por parte de los docentes.

Los alumnos deberán crear:

1 Biblioteca de clases, en lenguaje C#

Las clases que contendrá serán las siguientes:






```
public class Palabra //forma parte del espacio de nombres
Ahorcado.Palabras

/// <summary>
/// Retorna el número total de letras visibles que tiene la Palabra
/// </summary>
private int _cantidadLetrasVisibles
{
      //...
}


/// <summary>
/// Retorna true si la cantidad de letras de la Palabra es igual
/// a la cantidad de letras visibles
/// </summary>
public bool PalabraDescubierta
{
      //...
}

/// <summary>
/// Dentro de este constructor NO SE DEBE inicializar el atributo
_letrasDeLaPalabra.
/// Deberá agregar las Letras de acuerdo a la palabra con la que se
jugará. Tener en cuenta que en un principio, todas las letras NO DEBEN
estar visibles, a excepción de los espacios, que sí deben poder
visualizarse por defecto
/// </summary>
/// <param name="palabra">Representa la Palabra que se utilizará para
jugar</param>
/// <param name="caracterLetraOculta">Representa el caracter que se
mostrará para las letras que no fueron descubiertas</param>
```

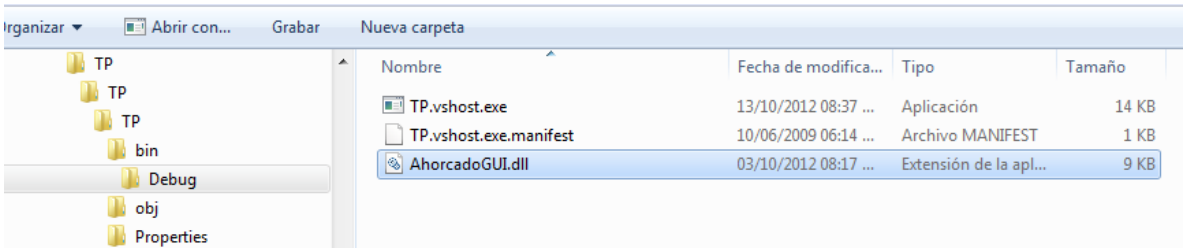
```
public Palabra(string palabra, char caracterLetraOculta)
{
      //...
}
/// <summary>
/// Este constructor es el único que inicializa el atributo
_letrasDeLaPalabra
/// </summary>
private Palabra()
{
      //...
}


/// <summary>
/// Retorna una cadena de caracteres con la palabra correcta
/// </summary>
public string PalabraCorrecta
{
      //...
}


/// <summary>
/// Retorna una cadena de caracteres con la palabra descubierta hasta el
momento, los caracteres que no fueron
/// descubiertos se mostrarán con el valor del atributo
_caracterLetraOculta, además deberá haber un espacio
/// entre cada letra.
/// Se deberá tener en cuenta si cada Letra está visible o no.
/// </summary>
public string PalabraDescubiertaHastaElMomento
{
      //...
}


/// <summary>
/// Recorre la lista de Letras, si existe el caracter que se recibe por
parámetros, deberá habilitar
/// la letra (ponerla visible), para TODAS las coincidencias.
/// Si se encuentra por lo menos 1 coincidencia, retornará verdadero.
/// </summary>
/// <param name="letraADescubrir"></param>
/// <returns></returns>
public bool DescubrirLetra(char letraADescubrir)
{
      //...
}
```

La clase Letra formará parte del mismo espacio de nombres que la clase Palabra.

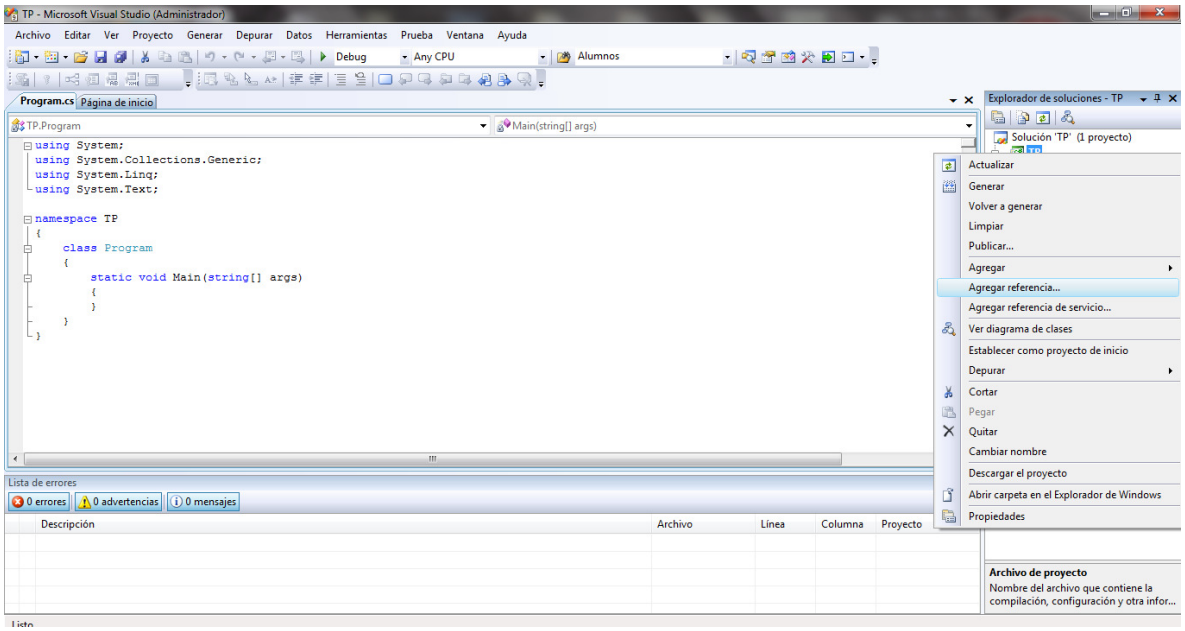
Además de esta Biblioteca de Clases, se deberá realizar una aplicación de consola, en la cual se utilizarán 2 dlls: la desarrollada por el alumno y otra para utilizar una interfaz “gráfica”.

Para utilizar la dll provista, se deberá agregar una referencia desde el proyecto que se quiera utilizar, de la siguiente manera:

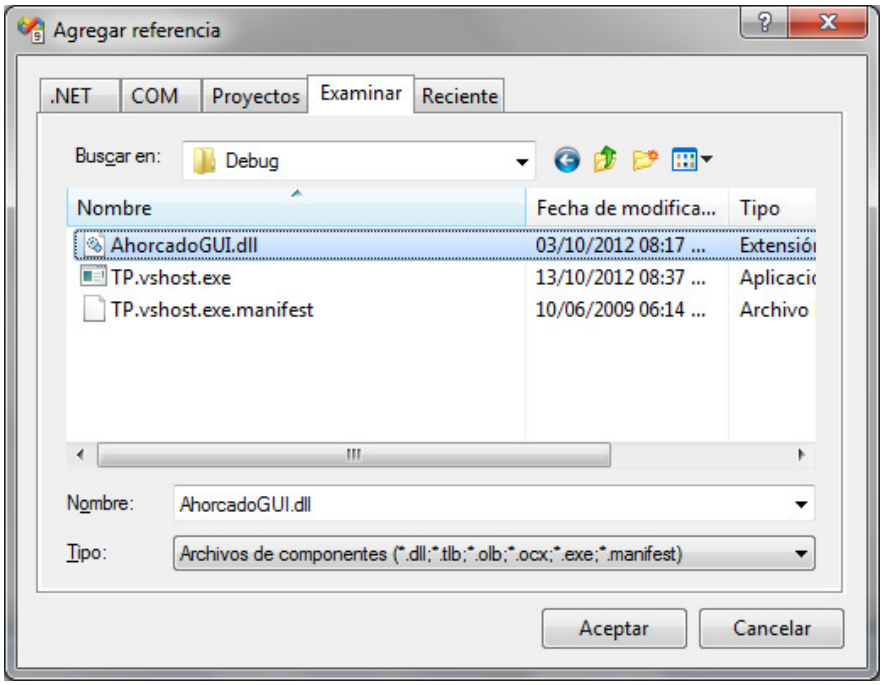
1. Copiar la dll dentro de la siguiente ubicación: ProyectoQueVaAUtilizarElAlumno\bin\Debug o bien ProyectoQueVaAUtilizarElAlumno\bin\Release



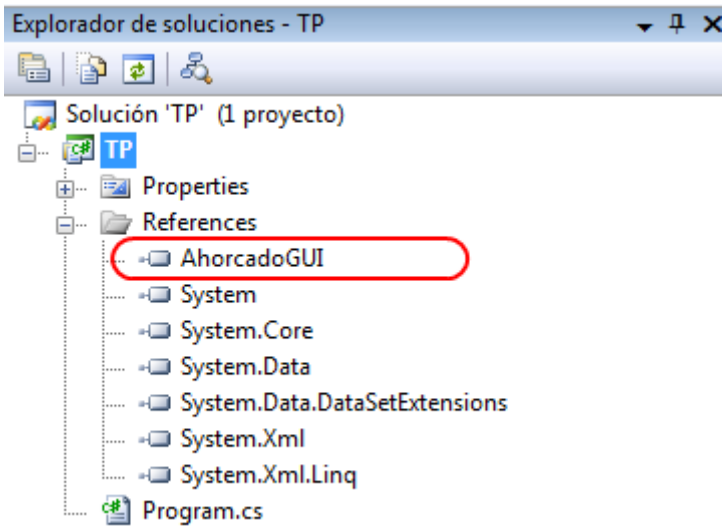
2. Click derecho sobre el proyecto, opción Agregar Referencia.



3. Seleccionar la solapa Examinar, luego ir hasta la ubicación del punto 1, click en Aceptar.



4. Dentro de la rama Referencias del proyecto, deberá observarse los siguiente:



5. Para utilizar las clases que forman parte de la dll, se deberá acceder a ellas a través del espacio de nombres Ahorcado.GUI

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ahorcado.GUI;

namespace TP
{
    class Program
    {
        static void Main(string[] args)
        {
            Pantalla pantalla = Pantalla.ObtenerInstancia("TP Prog 2 - Pepito Pérez", 0, 0);

            pantalla.DibujarFigura(eErrores.SeisErrores);
            Console.ReadKey();

        }
    }
}
```

6. Ejemplo de utilización de la “interfaz gráfica”:

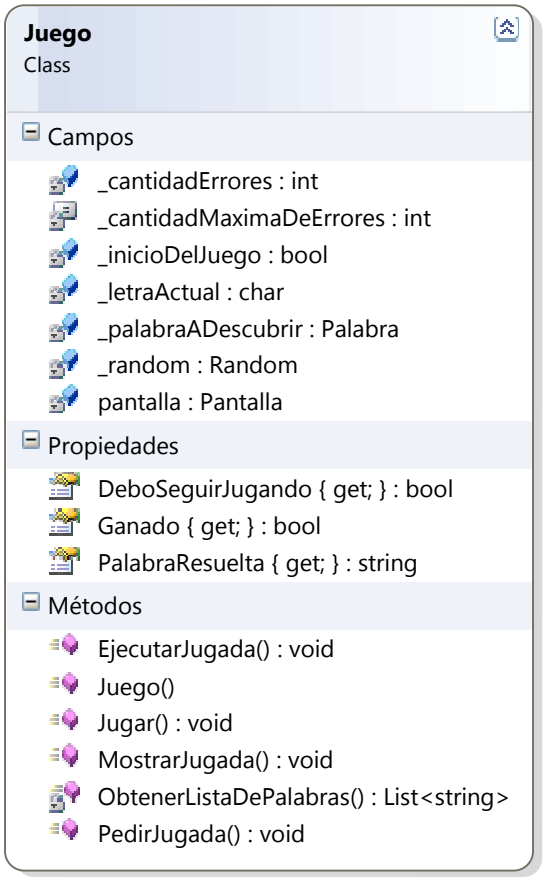
```
Pantalla pantalla = Pantalla.ObtenerInstancia("TP Prog 2 - Pepito Pérez", 0, 0);

pantalla.DibujarFigura(eErrores.SinErrores);
Console.ReadKey();
```

La aplicación de consola que cada alumno deberá desarrollar, tendrá el siguiente Main:

```
class Program
{
      private static void Main(string[] args)
      {
            Juego miJuego = new Juego();

            do
            {
                  miJuego.Jugar();
            } while (miJuego.DeboSeguirJugando);

            if (miJuego.Ganado)
            {
            Console.WriteLine("JUEGO GANADO :)");
            }
            else
            {
                  Console.WriteLine("JUEGO PERDIDO :( LA PALABRA CORRECTA ERA: {0}", miJuego.PalabraResuelta);
            }
            Console.WriteLine("Presione una tecla para continuar...");
            Console.ReadKey();
      }
```

```
}
```

Además, en esta aplicación de consola que deberán desarrollar, deberá contener la clase Juego (utilizada en el Main), y deberá tener las referencias correspondientes a las 2 dlls que se deben utilizar.




```
private const int _cantidadMaximaDeErrores = 6;

private Random _random = new Random();


/// <summary>
/// Retorna cuál era la palabra correcta
/// </summary>
public string PalabraResuelta
{
      get
      {
            //...
      }
```

```
}


/// <summary>
/// Retorna una lista de palabras, que luego se utilizarán para jugar con
una de ellas. En principio
/// deberá contener 10 elementos que quedan a criterio del alumno.
/// </summary>
/// <returns></returns>
private List<string> ObtenerListaDePalabras()
{
      //...
}


public Juego()
{
      List<string> _listaDePalabras;
      int indice;

      _listaDePalabras = ObtenerListaDePalabras();

       indice = _random.Next(0, _listaDePalabras.Count - 1);
      _palabraADescubrir = new Palabra(_listaDePalabras[indice], '_');
}

/// <summary>
/// Retornará true si la palabra fue descubierta
/// </summary>
public bool Ganado
{
      get
      {
            //...
      }
}

/// <summary>
/// Retornará verdadero si la cantidad de errores no supera la cantidad
máxima, y además la
/// palabra no fue descubierta.
/// </summary>
public bool DeboSeguirJugando
{
      get
      {
            //...
      }
}

public void Jugar()
{
      if (_inicioDelJuego)
      {
            this.MostrarJugada();
            _inicioDelJuego = false;
```

```
        }

        this.PedirJugada();
        this.EjecutarJugada();
        this.MostrarJugada();
}

/// <summary>
/// Pide el ingreso de una letra (por consola), almacena el valor leído
en _letraActual
/// </summary>
public void PedirJugada()
{
        //...
}

/// <summary>
/// Utiliza el atributo _letraActual (que previamente modificó el método
PedirJugada()), para
/// descubrir la letra de la _palabraADescubrir; tener en cuenta que en
el caso que no exista la
/// letra en la palabra a descubrir, la _cantidadErrores deberá ser
modificada.
/// </summary>
public void EjecutarJugada()
{
        //...
}

/// <summary>
/// De acuerdo a _cantidadErrores, invocará a pantalla.DibujarFigura(...)
con el parámetro que corresponda.
/// Luego, deberá mostrar por consola la palabra descubierta hasta el
momento
/// </summary>
public void MostrarJugada()
{
        //...
}
```