

# Examen de Programación – Curso 2014/15

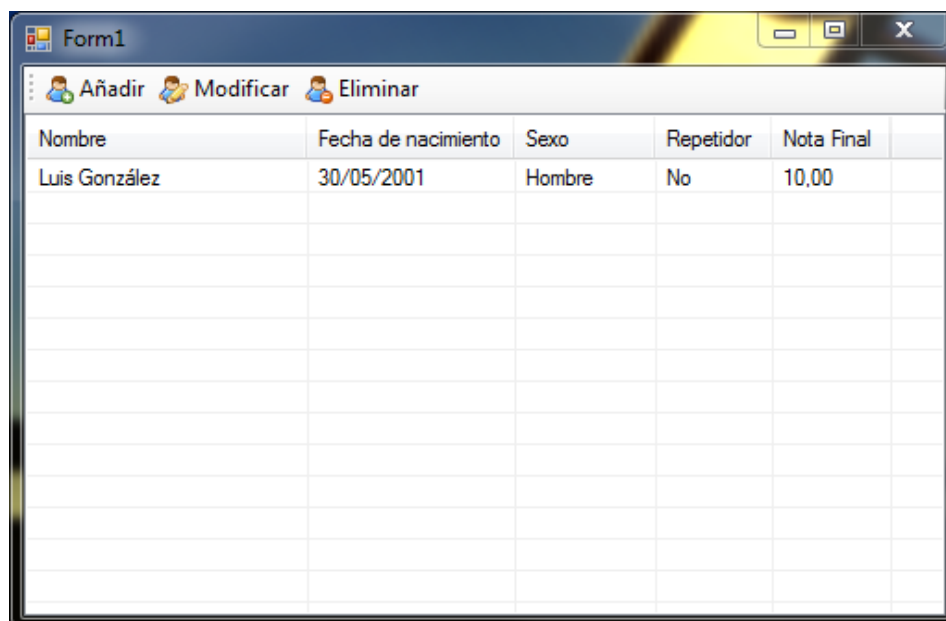
## 3<sup>er</sup> Trimestre

---

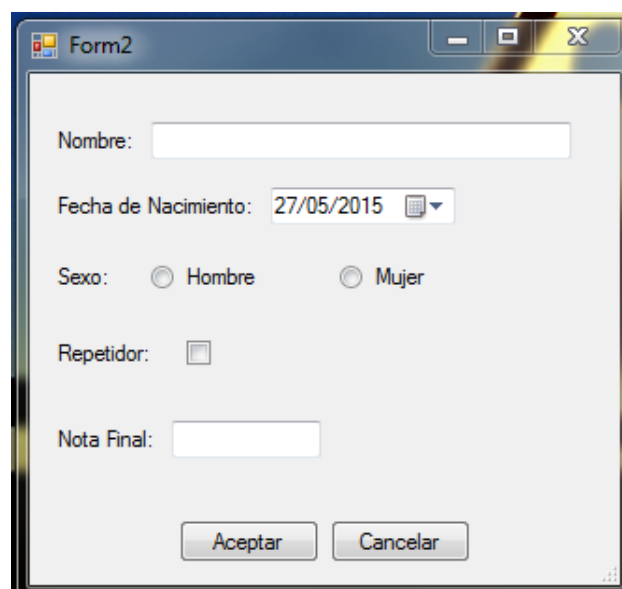
### INSTRUCCIONES:

- Cada ejercicio en un proyecto que se llamará “Ejercicio 1”, “Ejercicio 2”, etc.
- Para entregarlo, comprimido todo en un .zip o un .rar con vuestro nombre completo y llamadme para que lo recoja.

1. Escribe una aplicación visual “Alumnos”, que constará de un ListView y un segundo formulario para añadir datos al ListView. Aquí os pongo un par de capturas de pantalla:



- El menú de arriba está hecho con un ToolStrip, pero podéis poner botones normales si no sabéis usarlo.
- Estaría bien que los botones de “Modificar” y “Eliminar” sólo se activaran cuando haya algo seleccionado en el ListView.



- Lo del nombre y la nota es un TextBox.
- Para la fecha de nacimiento he usado un DateTimePicker. Es muy fácil de usar, en "Format" elegís el formato y en "Value" hay un DateTime que es la fecha elegida. Si no sabéis usarlo, poned un TextBox y ya está.
- Dos RadioButton para el sexo y un CheckBox para si es repetidor o no.

No hace falta guardar los datos en ninguna parte ni usar clases para mantener la información. El objetivo de este ejercicio es únicamente implementar la interfaz.

2. Escribe la clases "ApacheLogRegister" y "ApacheLog" que nos que nos permitirá leer un fichero de log de acceso del servidor Apache y realizar varias operaciones sobre los datos.

El formato del fichero es el siguiente:

- En cada línea va una entrada en la que se registra un acceso al servidor.
- Cada línea consta de varios campos, separados por espacios.
- Si un campo contiene espacios entre su interior, el campo se pone entre comillas para distinguir esos espacios de los separadores de campo.
- Para representar un campo que no tiene datos, se usa un guión.
- Los campos son los siguientes:
  - o IP del cliente que accede al servidor.
  - o Identificador del usuario identd (siempre va a venir vacío).
  - o Identificador del usuario HTTP (sólo viene relleno si se usa autenticación HTTP).
  - o Fecha de acceso (va entre corchetes). Incluye la zona horaria, que no tendremos en cuenta.
  - o Petición que envía el cliente (normalmente entre comillas porque va a contener espacios).
  - o Código de respuesta del servidor (código de tres números).
  - o Longitud de la respuesta en bytes.
  - o Referer de la petición de la página (puede estar en blanco o ser una dirección web).
  - o User Agent del cliente.

La clase "ApacheLogRegister" representará cada línea del fichero de log, y os la doy hecha:

```
class ApacheLogRegister
{
    public string ip;
    public string identd;
    public string httpAuth;
    public DateTime accessDate;
    public string httpRequest;
    public string httpResponseCode;
    public long dataSize;
    public string referer;
    public string userAgent;
}
```

Podéis añadir los constructores que creáis necesarios.

La clase “ApacheLog” constará de una lista de registros “listLog” como atributo y los siguientes métodos:

- ReadFile(string fichero): nos leerá un fichero de log cuyo nombre le pasamos por parámetro, lo procesará y lo volcará los datos a la lista de registros.
- ToString(): nos devolverá el contenido de la lista como un string para ponerlo por pantalla (bien con un Console.WriteLine o con algún elemento visual). El contenido de la cadena será un listado con un ancho de 80 caracteres y con cuatro columnas con información: IP de acceso, fecha y hora de acceso, petición del cliente (truncada si ocupa mucho) y respuesta del servidor.
- ToString(DateTime inicio, DateTime fin): nos devolverá un listado como el anterior pero sólo con los registros que tenga una fecha y hora de acceso entre “inicio” y “fin”.
- ToStringErrors(): nos devolverá un listado como los anteriores, pero en el que sólo se incluyen aquellas líneas en las que el código de respuesta de tres caracteres empiece por un 4 o por un 5 (que son los errores en HTTP).

3. Escribe un programa “Examen de inglés” que nos haga un examen de inglés del tipo “completa la frase con la palabra correcta”. Para ello, escribiremos la clase “EnglishTest” que tendrá las siguientes características:

- Dos atributos: una lista de cadenas “listaFrases” donde guardaremos las frases y otra lista de cadenas “listaPalabras” donde guardaremos las palabras sueltas.
- Un constructor al que le pasaremos un nombre de fichero de texto y que leerá los datos desde este fichero. Guardará todas las líneas en listaFrases y, además, separará todas las palabras y las guardará en listaPalabras.
- Un método “Pregunta()” que nos realizará una pregunta de la siguiente forma:
  - o Se elegirá una frase al azar de la lista de frases.
  - o Se elegirá una palabra de la frase, que será la que se va a esconder para que la intente acertar el examinado.
  - o Se cogerán al azar otras cuatro palabras de la lista de palabras, que junto con la palabra que hemos escondido, se darán como opciones.

Nuestro método nos devolverá un array de 7 cadenas, que serán:

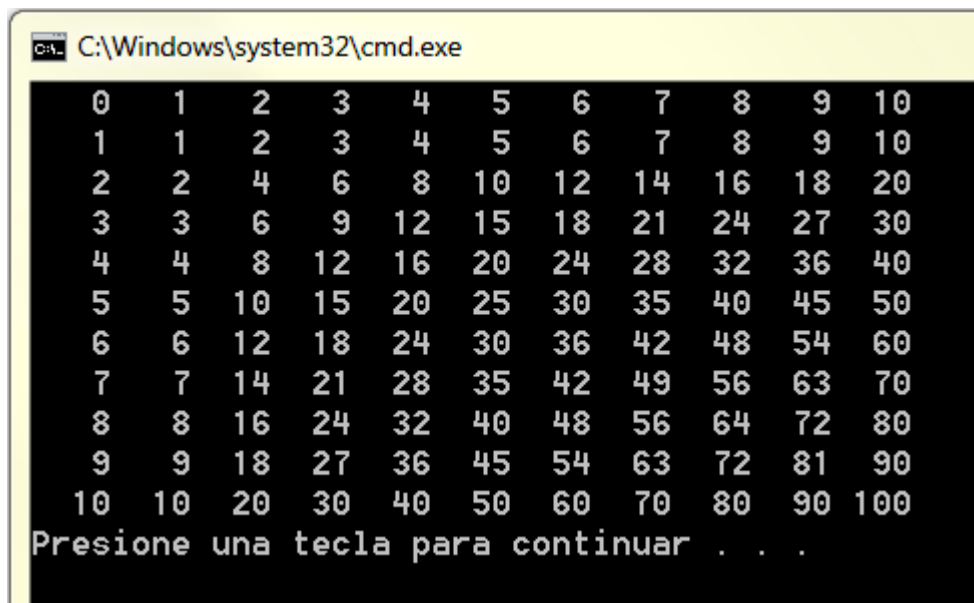
- 1) La frase con la palabra escondida sustituida por guiones bajos.
- 2) La palabra correcta que hay que acertar.
- 3) Y después las cinco palabras que se dan como opción, desordenadas (entre ellas se encontrará la respuesta correcta).

Ej.: Dead men \_\_\_\_ no lies (death / price / man / tell / himself)

El resto del programa consistirá en crear la clase, ejecutar el método Pregunta(), presentar la pregunta en pantalla y ver si la respuesta es correcta o no (hacer 5 preguntas, por ejemplo). Se recomienda hacerlo en consola, por ser mucho más sencillo, pero si a alguno le sobra tiempo y lo quiere hacer en visual... él sabrá.

- Opcional: que no se repitan las preguntas.
- Opcional: que si la palabra que se esconde es la primera, todas las opciones aparezcan con la primera letra en mayúsculas (que si no canta mucho), y viceversa.

4. Escribe un programa que nos presente la tabla de multiplicar de todos los números del 1 al 10 por pantalla. El programa constará de dos funciones:
- `static int[,] TablaMultiplicar()`: nos devolverá un array bidimensional de enteros con el contenido de la tabla de multiplicar con las siguientes características:
    - El array será de 11x11.
    - En la primera fila y la primera columna, estarán los números del 1 al 10.
    - A partir de la segunda fila y columna en adelante, estará el resultado de multiplicar los dos valores correspondientes a esa fila y esa columna. Esto no se entiende muy bien, así que mirad el dibujo de abajo y ya está.
  - `static void EscribeBonito(int[,] array)`: le pasamos un array bidimensional y nos lo escribe por pantalla para que se vea bonito (como el de la captura).
    - La función deberá funcionar para cualquier tamaño de array, no sólo uno de 11x11.
    - Para que quede bonito, la función deberá ir revisando primero todo el array para buscar cuál es el número que más espacio ocupa (en este caso, el 100 = 3 caracteres), para en una segunda vuelta, ir dejando el espacio necesario.



```
C:\Windows\system32\cmd.exe

0  1  2  3  4  5  6  7  8  9 10
1  1  2  3  4  5  6  7  8  9 10
2  2  4  6  8 10 12 14 16 18 20
3  3  6  9 12 15 18 21 24 27 30
4  4  8 12 16 20 24 28 32 36 40
5  5 10 15 20 25 30 35 40 45 50
6  6 12 18 24 30 36 42 48 54 60
7  7 14 21 28 35 42 49 56 63 70
8  8 16 24 32 40 48 56 64 72 80
9  9 18 27 36 45 54 63 72 81 90
10 10 20 30 40 50 60 70 80 90 100
Presione una tecla para continuar . . .
```

Este es un poco de regalo, no digáis que no. Pero por si acaso alguno acaba el examen muy pronto y se quiere calentar la cabeza:

- Extremadamente opcional: Escribir la función `TablaMultiplicar` usando un solo bucle.
- Absurdamente opcional: Escribir la función `TablaMultiplicar` sin usar bucles.