Ejercicios Persistencia de Objetos

Descargar estos ejercicios

Índice

Ejercicio 1

> Usando el ejercicio de **Microprocesadores** de la entrega anterior. Vamos a realizar la serialización de forma 'automática' a través de la clase `BinaryFormatter` y el atributo `[Serializable]` como describe en los apuntes.

Ejercicio 2

> Reconstruye el ejemplo de **Alumno con Dirección** que hay en el tema y prueba su funcionamiento. A partir de este ejemplo vamos a realizar las siguientes modificaciones: Añadiremos la clase **Nota**:
>
> - Dentro definirás el tipo enumerado Modulo con los valores ED, PROG, SI, FOL, LM y GBD
> - Los campos modulo de tipo Modulo, nota de tipo ushort y trimestre de tipo ushort con valores posibles 1, 2 y 3.
> - Definiremos accesores o getters para los 3 campos.
> - Un constructor que reciba todos los campos y un constructor copia.
>
> Cambios en la clase Alumno:
>
> - Añadiremos una nuevo campo que será un array de notas (clase Nota) que inicializaremos a null en el constructor.
> - Añadiremos un método para añadir notas al array (dimensionándolo si procede) haciendo una copia de la nota al guardarla en el array.
> - Modificaremos el método ToString para que muestre los datos del alumno de forma similar a esta.

```
Nombre: Marcos
Apellido: Jiménez
Edad: 20
Dirección: Cerámica 24 Alicante 03010
       1º 2º 3º
   ED  4  4  3
 PROG  6  6 10
   SI  0  1  2
  FOL  6  3  6
   LM 10  4  0
  GBD  3  1  3
```

- Por último comprueba la serialización y deserialización del Alumno y modifica el programa principal para encajar las nuevas definiciones propuestas. Para introducir las notas por módulo y trimestre, puedes hacerlo de forma aleatoria a través del siguiente método void AñadeNotasAleatorias(Alumno a).

💡 **Pista:** como las notas del array no tienen porqué estár organizadas por módulos o trimestres, a la hora de redefinir el ToString, se puede crear una matriz auxiliar en la que organicemos las notas por filas y columnas según a que módulo o trimestre pertenezcan (estó será fácil de hacer si utilizamos el valor entero que se asigna en las enumeraciones por defecto).

✅Ejercicio3

Modifica el ejercicio anterior de forma que en este caso las notas se guardarán en otro fichero denominado `notas.csv` de tal manera que se guardará junto a cada nota el nia del alumno al que pertenece (a modo de clave ajena en DB). Crea un método que a partir de un nia busque un alumno y sus notas.
Plantéate qué habría que hacer para añadir una opción de modificación de datos de un alumno.
Nota: en este ejercicio el alumno no tendrá ningún array de notas, sino que cada una de las notas estará guardada en el fichero. Para conseguir todas las notas del alumno, podremos crear un método que nos devuelva un array de **Nota**, este método se encargará de leer objeto a objeto el fichero, y cada vez que coincida el dni con el buscado se añadirá al array resultante, obteniendo al final un array con todas las notas que coincidan con el dni.