Índice

Ejercicios Unidad 23 - Persistencia Simple de Objetos

Descargar estos ejercicios

Antes de empezar

Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_poo. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente.

Warning

Para que la codificación quede más clara debes crear cada tipo de datos en un fichero distinto.

Ejercicio 1. Registro de Usuarios (JSON Línea a Línea)

Vamos a crear un sistema de registro donde guardaremos múltiples usuarios en un mismo fichero, añadiendo una línea JSON por cada usuario.

```
Ejercicio 1. Serialización Básica de un Objeto

--- MENÚ USUARIOS ---
1. Añadir usuario
2. Leer todos los usuarios
3. Salir
Elige una opción: 1

Creando nuevo usuario...
Nombre: Juan
Edad: 23
¿Está activo? (s/n): s
Usuario añadido correctamente al fichero.

--- MENÚ USUARIOS ---
1. Añadir usuario
2. Leer todos los usuarios
3. Salir
Elige una opción: 1

Creando nuevo usuario...
Nombre: Maria
Edad: 24
¿Está activo? (s/n): s
Usuario añadido correctamente al fichero.

--- MENÚ USUARIOS ---
1. Añadir usuario
2. Leer todos los usuarios
3. Salir
Elige una opción: 2

--- LISTA DE USUARIOS REGISTRADOS ---
Nombre: Juan, Edad: 23, Activo: True
Nombre: Maria, Edad: 24, Activo: True
```

Requisitos

1. **Crea** una record `Usuario` con tres propiedades:

- `Nombre` (string)
- `Edad` (int)
- `Activo` (bool)
- **Importante:** Usa el atributo `[JsonPropertyName("nombre_propiedad")]` para que en el JSON las claves aparezcan en minúsculas ("nombre", "edad", "activo").

2. Completa el programa principal con las siguientes opciones:
   - **1. Añadir usuario:**
     - Pide los datos por consola.
     - Crea el objeto `Usuario`.
     - Serialízalo a una cadena JSON (`JsonSerializer.Serialize`).
     - Añade esa cadena como una nueva línea en el fichero `usuario.json` (usa `StreamWriter` con `append: true`).
   - **2. Leer todos los usuarios:**
     - Comprueba si el fichero existe.
     - Lee el fichero línea a línea (`StreamReader`).
     - Deserializa cada línea individualmente (`JsonSerializer.Deserialize`) y muestra los datos por pantalla.
   - **3. Salir.**

Ejercicio 2. Persistencia de Listas y Composición

En este ejercicio vamos a trabajar con estructuras más complejas, listas y relaciones entre objetos.

Requisitos

1. **Utiliza** la clase `Usuario` del ejercicio anterior.
2. **Crea** una nueva clase `Grupo` que tenga:
   - `NombreGrupo` (string)
   - `Miembros` (Lista de objetos `Usuario`)
   - **Importante:** Usa el atributo `[JsonPropertyName("nombre_propiedad")]` para que en el JSON las claves aparezcan en camelCasing.
3. **Desarrolla** los siguientes métodos estáticos en una clase de utilidad `GestorDeGrupos` :
   - `void GuardaGrupo(Grupo g, string ruta)` : Este método debe serializar el objeto `Grupo` completo (incluyendo su lista de usuarios) y escribirlo en disco.

- `List<>Grupo CargaGrupo(string ruta)` : Este método debe leer el fichero, deserializarlo y devolver todas las instancias de `Grupo` en una colección reconstruida.

Ejercicio 3. Opciones de Serialización y Propiedades Opcionales

Vamos a refinar el comportamiento del serializador para manejar casos especiales y mejorar la legibilidad del fichero generado.

```
Ejercicio 3. Opciones de Serialización

Lista guardada en usuarios_avanzados.json

Contenido del fichero generado (Verificación visual):
------------------------------------------------
[
  {
    "nombre": "Usuario Con Apodo",
    "edad": 22,
    "activo": true,
    "apodo": "TheUser",
    "fechaRegistro": "2026-01-09T21:09:24.2727021+01:00"
  },
  {
    "nombre": "Usuario Sin Apodo",
    "edad": 45,
    "activo": true,
    "fechaRegistro": "2025-12-30T21:09:24.2869067+01:00"
  }
]

Leídos 2 usuarios ok.

Pulse una tecla para seguir...
```

Requisitos

1. **Modifica** la clase `Usuario` añadiendo dos nuevas propiedades:
   - `Apodo` (string, puede ser nulo)
   - `FechaRegistro` (DateTime)
2. **Configura** las opciones de serialización (`JsonSerializerOptions`) para cumplir con lo siguiente:

- **Formateo**: El JSON resultante debe estar indentado (bonito/legible), no todo en una línea.
- **Nulos**: Si la propiedad `Apodo` es nula, **no debe aparecer** en el JSON generado (ignorar nulos).
- **Insensibilidad**: Permite que al deserializar no importen las mayúsculas/minúsculas de las propiedades (Case Insensitive).

3. **Crea** dos usuarios de prueba:
   - Uno con `Apodo` asignado.
   - Otro con `Apodo` nulo.
4. **Serializa** una lista con estos dos usuarios aplicando las opciones configuradas y **guárdala** en `usuarios_avanzados.json`. **Abre** el fichero generado y comprueba visualmente que el usuario sin apodo no tiene ese campo en el JSON y que el texto está indentado.