Índice

Unidad 12 - Tipos Valor vs Tipos Referencia en POO

Descargar estos ejercicios

> **Antes de empezar**
>
> Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_tipos_valor. 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.

Ejercicio 1. Comparación entre tipos valor y tipos referencia

Programa que demuestre las diferencias fundamentales entre tipos valor (DateTime, TimeSpan) y tipos referencia (StringBuilder, arrays) en cuanto a asignación, paso por parámetros y comportamiento en memoria.

```
Ejercicio 1: Comparación entre tipos valor y tipos referencia

--- DateTime (tipo valor) ---
Fecha original: 25/12/2025 10:30:00
Fecha copiada: 25/12/2025 10:30:00

Modificando fecha copiada (añadiendo 5 días)...
Fecha original: 25/12/2025 10:30:00
Fecha copiada: 30/12/2025 10:30:00

--- StringBuilder (tipo referencia) ---
Texto original: "Hola mundo"
Texto copiado: "Hola mundo"

Modificando texto copiado (añadiendo "!!!")...
Texto original: "Hola mundo!!!"
Texto copiado: "Hola mundo!!!"


--- Tipos valor como parámetros ---
DateTime antes de llamar a método: 01/01/2025 12:00:00
DateTime después de llamar a método: 01/01/2025 12:00:00
(El método no puede modificar el original)

--- Tipos referencia como parámetros ---
StringBuilder antes de llamar a método: "Inicial"
StringBuilder después de llamar a método: "Inicial - Modificado en método"
(El método SÍ puede modificar el original)

Pulsa una tecla para acabar...
```

Requisitos:

- **Métodos para demostrar tipos valor**:
  - Método **DemuestraTipoValor()** que cree dos variables DateTime, a la primera se le da un valor y después asignas a la segunda la primera, muestra sus valores.

Posteriormente modificas una y vuelve a mostrar los valores. Usar **ReferenceEquals()** para demostrar que son objetos diferentes en memoria. Usar comparación directa **==** para mostrar si tienen el mismo valor.

- **Métodos para demostrar tipos referencia**:
  - Método **DemuestraTipoReferencia()** que cree dos variables StringBuilder y realiza los mismos pasos que para el método anterior. Demostrar que modificar una variable afecta a la otra porque apuntan al mismo objeto.
- **Métodos para paso por parámetros**:
  - Método **ModificaTipoValor(DateTime fecha)** que intente modificar el parámetro y demuestre que no afecta al original.
  - Método **ModificaTipoReferencia(StringBuilder texto)** que modifique el parámetro y demuestre que SÍ afecta al original.

Ejercicio 2. Calculadora de tiempo y fechas con tipos valor

Programa que demuestre el uso práctico de tipos valor DateTime, TimeSpan, DateOnly y TimeOnly para realizar cálculos temporales.

```
Ejercicio 2: Calculadora de tiempo y fechas

=== INFORMACIÓN FECHA ACTUAL ===

Fecha actual: jueves, 31 de julio de 2025
Hora actual: 20:41
Fecha solo: 31/07/2025
Hora solo: 20:41

Introduce tu fecha de nacimiento (dd/MM/yyyy): 15/02/2001

=== CÁLCULOS CON DATETIME ===
Fecha de nacimiento: 15/02/2001
Cálculos desde tu nacimiento:
- Días vividos: 8.932 días
- Años completos: 24 años y 166 días
- Próximo cumpleaños: 15/02/2026 (en 198 días)
- Día de la semana que naciste: jueves

Vamos a calcular la jornada laboral...
Introduce la hora de inicio: 8:00
Introduce la hora de fin: 15:30
Introduce la hora de inicio del descanso: 10:30
Introduce la duración del descanso (en minutos): 30

=== CÁLCULOS CON TimeOnly ===
Hora de inicio trabajo: 8:00
Hora de fin trabajo: 15:30
Duración jornada laboral: 07:30:00

Hora de inicio descanso: 10:30
Duración del descanso: 00:30:00
Hora de fin descanso: 11:00

Tiempo trabajado efectivo: 07:00:00
Horas trabajadas en la semana (5 días): 1.11:00:00

Vamos a mostrar comparaciones temporales...

=== COMPARACIONES TEMPORALES ===
¿El 15 de marzo es antes que el 31 de julio? True
¿Las 08:30 es antes que las 17:45? True
¿Quedan más de 100 días para fin de año? True

Pulsa una tecla para acabar...
```

Requisitos:

- Método **CalculaEdad(DateTime fechaNacimiento)** que calcule y muestre como en salida: días vividos, años completos y próximo cumpleaños y día de la semana.

- Método **CalculaJornadaLaboral()** que pida los datos necesarios en el método y calcule duración jornada laboral, hora de fin de descanso y tiempo efectivo, usando `TimeOnly` y `TimeSpan`. Por último tendrá que calcular horas semanales.
- Método **MuestraComparaciones()** que con las fechas 15/03/2025 08:30:00 y 31/07/2025 17:45:00 use los operadores de comparación con fechas y horas para conseguir la salida de pantalla.

Ejercicio 3. Conversor de formatos de fecha y zona horaria

Programa que demuestre el formateo avanzado de fechas, conversión entre zonas horarias y parsing de diferentes formatos de entrada, usando los tipos valor DateTime y TimeSpan.

```
Ejercicio 3: Conversor de formatos de fecha y zona horaria

=== FORMATOS PERSONALIZADOS ===
Formato corto: 31/07/2025
Formato largo: jueves, 31 de julio de 2025
Formato ISO 8601: 2025-07-31T20:51:57
Formato americano: 07/31/2025 8:51:57 PM
Formato europeo: 31.07.2025 20:51:57
Formato personalizado: "El día 31 de julio del año 2025 a las 20 horas y 51 minutos"

=== CONVERSIONES DE ZONA HORARIA ===
Hora local (Madrid): 04/02/2025 14:30:25
En Londres (UTC+0): 04/02/2025 13:30:25
En Nueva York (UTC-5): 04/02/2025 08:30:25
En Tokio (UTC+9): 04/02/2025 22:30:25

=== PARSING DE FECHAS ===
Introduce fecha en formato dd/MM/yyyy: 12/05/2002
Fecha parseada: 12/05/2002 00:00:00
¿Es válida? True

Introduce fecha en formato MM-dd-yyyy: 05-02-2009
Fecha parseada: 02/05/2009 00:00:00
¿Es válida? True

Introduce fecha en formato inválido: 5-14-2000
¿Es válida? False
Error: El formato de fecha no es válido.

Pulsa una tecla para acabar...
```

Requisitos:

- Método **MuestraFormatosPersonalizados()** que muestre la fecha y hora actual en múltiples formatos. Usar los formatos necesarios para conseguir la salida esperada.
- Método **ConvierteAZonasHorarias(DateTime fechaLocal)** que muestre la hora en diferentes zonas. Crear array de tuplas (nombre de ciudad, offset UTC) para las

conversiones. Usa **AddHours()** para ajustar las horas según el offset UTC. *Puedes mirar los apuntes para resolver esto*.

- Método **PruebaParsing()** que solicite fechas en diferentes formatos y las valide. Usa `DateTime.TryParseExact()` para formatos específicos. Manejar errores de parsing y mostrar mensajes informativos.

Ejercicio 4. Sistema de identificadores únicos con GUID

Crea un programa que simule un sistema de gestión de identificadores únicos para diferentes tipos de elementos usando el tipo valor `Guid`.

```
Ejercicio 4: Sistema de identificadores únicos con GUID

Generando identificadores únicos ...

--- Usuarios del sistema ---
Usuario 1: 22233f5d-c06a-4f3f-95ee-00240fef9658
Usuario 2: 8b65ee81-38d5-4bc1-916c-555593d2d026
Usuario 3: 697d9bed-aee2-4aed-be21-f8fb07f54e6f

--- Información del primer GUID ---
GUID analizado: 22233f5d-c06a-4f3f-95ee-00240fef9658
Versión: 4
Variante: 9
Formato sin guiones: 22233f5dc06a4f3f95ee00240fef9658
Formato con llaves: {22233f5d-c06a-4f3f-95ee-00240fef9658}
Formato con paréntesis: (22233f5d-c06a-4f3f-95ee-00240fef9658)

--- Verificación de duplicados ---
¿Usuario 1 y Usuario 2 tienen el mismo ID? False
¿Todos los IDs son únicos? True

--- Conversión desde texto ---
ID desde texto válido: 12345678-9abc-def0-1234-56789abcdef0 -> Convertido correctamente
ID desde texto inválido: texto-no-valido -> Error en conversión

Pulsa una tecla para acabar...
```

Requisitos:

- Implementar método `UsoGuid` que tendrá las siguientes funcionalidades:
  - Arrays para los guid de los usuarios(3 elementos)
  - Generar GUIDs únicos usando `Guid.NewGuid()`.
  - Mostrar propiedades `Version` y `Variant`
  - Convertir a diferentes formatos: `ToString("N")`, `ToString("D")`, `ToString("B")`, `ToString("P")`

- Comparar GUIDs para verificar unicidad entre los que componen el array.
- Intentar conversión desde string válido con `Guid.Parse()`
- Manejar conversión desde string inválido con `Guid.TryParse()`