Índice

Ejercicios Unidad 10

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.

Ejercicio 1. Uso de TADs de la BCL y métodos estáticos de Console

Programa que demuestre el uso de TADs (Tipos Abstractos de Datos) predefinidos en la BCL de C# como Random y Point, y métodos estáticos de una clase contenedor como los que nos podemos encontrar en la clase Console. Este ejercicio a diferencia de los demás, está programado en casi su totalidad, solo deberás sustituir las líneas comentadas por el código que se pide.

```
Ejercicio 1: Uso de TADs de la BCL y métodos estáticos de Console

Introduce coordenada X (2-140): 75
Introduce coordenada Y (2-30): 15
          ☺

Presiona 'Escape' para salir o cualquier otra tecla para cambiar el tamaño de la ventan

Introduce coordenada X (2-140): 25
Introduce coordenada Y (2-30): 8
   ☺

Presiona 'Escape' para salir o cualquier otra tecla para cambiar el tamaño de la ventan
Programa terminado.
```

Requisitos:

- Método **GeneraColor()** que use el TAD de la BCL `Random` para seleccionar un `ConsoleColor` aleatorio. Devolveré el valor de color generado. `ConsoleColor` es una enumeración de la clase Console, por lo que puedes establecer el número de elementos que tiene con `Enum.GetValues()` para establecer el tope del valor aleatorio.
- **Uso de TAD Point** (estructura de la BCL de C#):
  - Usar la estructura **Point** de `System.Drawing` para almacenar coordenadas X e Y.
  - Deberás crear un objeto de tipo Point en el método **EstableceCoordenadas** y devolverlo
  - Fíjate cómo Point encapsula dos valores relacionados (X, Y) en un solo objeto.

Ideas

- Uso de **métodos estáticos de Console** (clase contenedor). Fíjate en los distintos métodos estáticos de la clase Console que se usan a lo largo del programa.
- **Console.SetCursorPosition(int left, int top)**: Posicionar el cursor en coordenadas específicas.
- **Console.BackgroundColor/ForegroundColor**: Cambiar colores de fondo y texto.
- **Console.CursorVisible**: Ocultar el cursor durante la ejecución.
- **Console.Clear()**: Limpiar la pantalla con el color de fondo actual.
- **Console.SetWindowSize()/SetBufferSize()**: Configurar dimensiones de ventana y buffer.
- **Console.WindowHeight/WindowWidth**: Obtener dimensiones actuales de la ventana.
- **Console.LargestWindowWidth/LargestWindowHeight**: Obtener dimensiones máximas disponibles.
- **Console.OutputEncoding**: Configurar codificación para caracteres especiales (UTF-8).
- **Console.ResetColor()**: Restaurar colores originales de la consola.

Ejercicio 2. Calculadora matemática (Clase contenedor vs TAD)

Este ejercicio permitirá comparar dos enfoques diferentes: una clase contenedor con métodos estáticos y una clase TAD para realizar operaciones matemáticas básicas. La clase contenedor con sus métodos, deberá ser creada por tí, mientras que la clase TAD ya está codificada.

```
Ejercicio 2: Calculadora matemática (Clase contenedor vs TAD)

--- CLASE CONTENEDOR (Métodos estáticos) ---
Introduce primer número para TAD:
Por favor, introduce un número válido.
Introduce primer número para TAD: 5
Introduce segundo número para TAD: 4

Resultados usando CalculadoraEstatica:
Suma: 9
Resta: 1
Multiplicación: 20
División: 1,25

--- CLASE TAD (Objetos) ---
Introduce primer número para TAD: 5
Introduce segundo número para TAD: +
Por favor, introduce un número válido.
Introduce segundo número para TAD: 4
Creando calculadora TAD con valores iniciales...

Calculadora 1: Valor1=5, Valor2=4
Suma: 9
Resta: 1
Multiplicación: 20
División: 1,25

Creando segunda calculadora...
Calculadora 2: Valor1=10, Valor2=5

Modificando valores de calculadora 1...
Calculadora 1 modificada: Valor1=20, Valor2=3
Nueva suma: 23

¿Las calculadoras son el mismo objeto? False
¿Las calculadoras tienen los mismos valores? False

Estado final:
Calculadora 1: Valor1=20, Valor2=3
Calculadora 2: Valor1=10, Valor2=5
¡Hasta la próxima!
```

Requisitos:

- **Clase CalculadoraEstatica** (contenedor con métodos estáticos):
  - Método **Suma(double a, double b)** que devuelva la suma.
  - Método **Resta(double a, double b)** que devuelva la resta.
  - Método **Multiplica(double a, double b)** que devuelva la multiplicación.

    - Método **Divide(double a, double b)** que devuelva la división.
- **Clase CalculadoraTAD** (permite crear objetos):
    - Revisa el código que ya está creado en la clase CalculadoraTAD. Fíjate como son los métodos que están dentro de la clase, ¿tienen el modificador static?¿en la clase contenedora llevan el modificador static?
- **Clase Program** clase contenedora que a parte del método Main que ya viene codificado, se le deberá añadir el código de los siguientes métodos:
    - **Método PideNumero(string mensaje)**:
        - Tipo de retorno: `double`
        - Parámetros: `string mensaje` (texto a mostrar al usuario)
        - Función: Solicita un número al usuario y valida que sea correcto usando `double.TryParse()`
        - Usa un bucle `while` que se repite hasta que el usuario introduzca un número válido
        - Si la entrada no es válida, muestra "Por favor, introduce un número válido."
        - Devuelve el número válido introducido por el usuario
    - **Método DemuestraCalculadoraEstatica(double valor1, double valor2)**:
        - Tipo de retorno: `void`
        - Parámetros: `double valor1, double valor2` (los dos números para las operaciones)
        - Función: Demuestra el uso de la clase contenedor con métodos estáticos
        - Llama a todos los métodos estáticos de `CalculadoraEstatica`: Suma, Resta, Multiplica y Divide
        - Muestra los resultados de todas las operaciones matemáticas
        - **Nota**: Observa que para llamar a los métodos se usa `CalculadoraEstatica.NombreMetodo()` (nombre de clase + método)
    - **Método DemuestraCalculadoraTAD(double valor1, double valor2)**:
        - Tipo de retorno: `void`
        - Parámetros: `double valor1, double valor2` (valores iniciales para crear objetos)
        - Función: Demuestra el uso de la clase TAD creando múltiples objetos
        - Crea dos objetos `CalculadoraTAD` diferentes con valores distintos
        - Compara objetos usando `ReferenceEquals()` (compara si son el mismo objeto en memoria)
        - Compara valores usando comparación directa de campos (`Valor1` y `Valor2`)
        - **Nota**: Observa que para llamar a los métodos se usa `nombreObjeto.NombreMetodo()` (instancia + método)

Ideas

Algunas claves a observar en este ejercicio:

- **Métodos estáticos**: Se llaman desde la clase, no necesitan crear objetos
- **Métodos de instancia**: Se llaman desde objetos específicos, cada objeto tiene su propio estado
- **Estado**: Los objetos TAD mantienen datos (Valor1, Valor2), las clases estáticas no
- **Múltiples objetos**: Se pueden crear varios objetos TAD independientes
- **ReferenceEquals()**: Compara si dos variables apuntan al mismo objeto en memoria

Ejercicio 3. Sistema de URIs usando TAD Uri de la BCL

Programa que demuestre el uso del TAD Uri de la BCL de C# (System) para trabajar con direcciones web y validar URLs. Para de esta forma poder entender mejor el concepto de TAD.

```
Ejercicio 3: Sistema de gestión de URIs (TAD de la BCL)
Usando la clase Uri de System

=== CREACIÓN DE URIs Y ANÁLISIS ===
Introduce URI 1: https://www.google.com/search?q=csharp
¿Es una URI válida? True

URI completa: https://www.google.com/search?q=csharp
Esquema: https
Host: www.google.com
Puerto: 443 (puerto por defecto HTTPS)
Ruta: /search
Query: ?q=csharp
¿Es archivo?: False
¿Es UNC?: False
¿Es absoluta?: True

Introduce URI 2: ftp://files.example.com:21/documents/file.txt
¿Es una URI válida? True

URI completa: ftp://files.example.com:21/documents/file.txt
Esquema: ftp
Host: files.example.com
Puerto: 21
Ruta: /documents/file.txt
Query:
¿Es archivo?: False
¿Es UNC?: False
¿Es absoluta?: True

Introduce URI 3: mailto:info@example.com
¿Es una URI válida? True

URI completa: mailto:info@example.com
Esquema: mailto
Host:
Puerto: -1
Ruta: info@example.com
Query:
¿Es archivo?: False
¿Es UNC?: False
¿Es absoluta?: True

Introduce URI 4: invalid-url-format
¿Es una URI válida? False
URI inválida detectada, se creará URI por defecto.

=== COMPARACIONES ===
¿URI 1 y URI 2 tienen el mismo host? False
¿URI 1 y URI 2 tienen el mismo esquema? False
```

```
¿URI 1 y URI por defecto son iguales? False
```

Requisitos:

- Método ***CreaUris*** que crea un array de 4 URIs (3 válidas + 1 inválida para demostrar validación) y lo devuelve.
- Método **MostrarInformación** que muestre las propiedades de cada URI válida (Scheme, Host, Port, Path, Query, etc.).
  - **Scheme**: Protocolo de la URI (http, https, ftp, mailto, etc.).
  - **Host**: Nombre del servidor o dominio.
  - **Port**: Puerto de conexión.
  - **AbsolutePath**: Ruta absoluta dentro del servidor.
  - **Query**: Parámetros de consulta.
  - **IsFile**: Si la URI apunta a un archivo local.
  - **IsUnc**: Si la URI es una ruta UNC.
  - **IsAbsoluteUri**: Si la URI es absoluta o relativa.
- Método **Compara** al que le llegan dos URIs y las compara sí (mismo host, mismo esquema, igualdad).
- Se pasa el código de la Main.