Práctica : Creación y uso de métodos

Objetivos

Al final de esta práctica, usted será capaz de:

Crear y hacer llamadas a métodos con y sin parámetros.

Utilizar distintos mecanismos para pasar parámetros.

Requisitos previos

Antes de realizar la práctica debe estar familiarizado con los siguientes temas:

Creación y uso de variables

Instrucciones de C#

Ejercicio 1
Uso de parámetros en métodos que devuelven valores

En este ejercicio definirá y usará parámetros de entrada en un método que devuelve un valor. También escribirá un sistema de prueba que lea dos valores de la consola y muestre los resultados.

Creará una clase llamada **Utils**. En esa clase creará un método llamado **Greater** que aceptará como entrada dos parámetros enteros y devolverá el valor del que sea mayor.

Para probar la clase creará otra clase llamada **Test** que pedirá dos números al usuario, llamará a **Utils.Greater** para determinar cuál de los dos es mayor e imprimirá el resultado.

Cómo crear el método Mayor

1. Abra el proyecto Utils.sln en la carpeta Starter\Utility dentro del fichero lab05.zip.

   Encontrará un espacio de nombres llamado **Utils** que contiene una clase del mismo nombre en la que escribirá el método **Greater**.

2. Cree el método **Greater** como se indica a continuación:

   a. Abra la clase **Utils**.

   b. Añada a la clase **Utils** un método public static llamado **Greater**.

   c. El método aceptará dos parámetros **int** llamados *a* y *b* que se pasarán por valor, y devolverá un valor **int** que represente el mayor de los dos números.

   El código para la clase **Utils** debería ser como éste:

```
namespace Utils
{
    using System;

    class Utils
    {

       //
       // Devuelve el mayor de dos valores enteros
       //

       public static int Mayor(int a, int b)
       {
         if (a > b)
             return a;
         else
             return b;
       }
    }
}
```

Cómo probar el método Mayor

1. Abra la clase **Test**.
2. Escriba el siguiente código en el método **Main**.
   a. Defina dos variables enteras llamadas *x* e *y*.
   b. Añada instrucciones que lean dos enteros desde la entrada por teclado y los asignen a *x* e *y*. Use los métodos **Console.ReadLine** e **int.Parse** de módulos anteriores.
   c. Defina otro entero llamado *greater*.
   d. Haga una llamada al método **Greater** para probarlo, y asigne el valor devuelto a la variable *mayor*.
3. Escriba el código necesario para mostrar el mayor de los dos enteros empleando **Console.WriteLine**.

   El código para la clase **Test** debería ser como éste:

```
namespace Utils
{
    using System;

    /// <resumen>
    ///   Sistema de prueba
    /// </resumen>

    public class Test
    {
        public static void Main( )
        {
            int x;       // Valor de entrada 1
            int y;       // Valor de entrada 2
            int mayor;   // Resultado de Mayor( )

            // Obtener números de entrada
            Console.WriteLine("Escriba el primer número:");
            x = int.Parse(Console.ReadLine( ));
            Console.WriteLine("Escriba el segundo número:");
            y = int.Parse(Console.ReadLine( ));

            // Probar el método Mayor( )
            mayor = Utils.Mayor(x,y);
            Console.WriteLine("El valor mayor es "+
➥ mayor);

        }
    }
}
```

4. Guarde el trabajo realizado.
5. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.

Ejercicio 2
Uso de métodos con parámetros referencia

En este ejercicio escribirá un método llamado **Swap** que intercambiará los valores de sus parámetros. Utilizará parámetros que se pasan por referencia.

Cómo crear el método Intercambio

1. Abra el proyecto Utils.sln en la carpeta Utility del ejercicio anterior, si no está ya abierto.
2. Añada el método **Swap** a la clase **Utils** como se indica a continuación:
   a. Añada un método public static void llamado **Swap**.
   b. **Swap** aceptará dos parámetros **int** llamados *a* y *b*, que se pasarán por referencia.
   c. Escriba instrucciones en el cuerpo de **Swap** para intercambiar los valores de *a* y *b*. Tendrá que crear una variable **int** local en **Swap** para guardar temporalmente uno de los valores durante el intercambio. Llame a esta variable *temp*.

   El código para la clase **Utils** debería ser como éste:

```
namespace Utils
{
    using System;

    public class Utils
    {

    ... código anterior omitido para mayor claridad ...

    //
    // Intercambiar dos enteros, pasados por referencia
    //

    public static void Intercambio(ref int a, ref int b)
    {
        int temp = a;
        a = b;
        b = temp;
    }

    }
}
```

Cómo probar el método Intercambio

1. Edite el método **Main** en la clase **Test** ejecutando los siguientes pasos:
   a. Asigne valores a las variables enteras *x* e *y*.
   b. Haga una llamada al método **Swap**, pasando estos valores como parámetros.

   Visualice los nuevos valores de los dos enteros antes y después de intercambiarlos. El código para la clase **Test** debería ser como éste:

```
namespace Utils
{
    using System;

    public class Test
    {

    public static void Main( )
    {
        ... código anterior omitido para mayor claridad ...

        // Probar el método Intercambio
        Console.WriteLine("Antes: " + x + "," + y);
        Utils.Swap(ref x,ref y);
        Console.WriteLine("Después: " + x + "," + y);

    }

    }
}
```

2. Guarde el trabajo realizado.
3. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.

Consejo Si los parámetros no se intercambiaran como esperaba, compruebe que no los ha pasado como parámetros **ref**.

Ejercicio 3
Uso de métodos con parámetros de salida

En este ejercicio definirá y usará un método static con un parámetro de salida.

Escribirá un nuevo método llamado **Factorial** que recibirá un valor **int** y calculará su factorial. El factorial de un número es el producto de todos los números entre 1 y ese número; el factorial de cero es 1 por definición. A continuación se dan algunos ejemplos de factoriales:

Factorial(0) = 1

Factorial(1) = 1

Factorial(2) = 1 * 2 = 2

Factorial(3) = 1 * 2 * 3 = 6

Factorial(4) = 1 * 2 * 3 * 4 = 24

Cómo crear el método Factorial

1. Abra el proyecto Utils.sln en la carpeta Utility del ejercicio anterior, si no está ya abierto.
2. Añada el método **Factorial** a la clase **Utils** como se indica a continuación:
   a. Añada un método public static llamado **Factorial**.
   b. Este método recibirá dos parámetros llamados *n* y *respuesta*. El primero se pasa por valor y es el **int** cuyo factorial se va a calcular. El segundo es un parámetro **out int** que se utilizará para devolver el resultado.
   c. El método **Factorial** debe devolver un valor **bool** que indique si ha funcionado (podría desbordarse y producir una excepción).
3. Añada funcionalidad al método **Factorial**.

   La forma más sencilla de calcular un factorial e por medio de un bucle. Ejecute los siguientes pasos para añadir funcionalidad al método:
   a. Cree una variable **int** llamada *k* en el método **Factorial** para utilizarla como contador del bucle.
   b. Cree otra variable **int** llamada *f*, que se usará como valor de trabajo dentro del bucle. Inicialice la variable de trabajo *f* con el valor 1.
   c. Use un bucle **for** para realizar la iteración. Comience con un valor de 2 para *k*, y termine cuando *k* llegue al valor del parámetro *n*. Incremente *k* cada vez que se ejecute el bucle.
   d. En el cuerpo del bucle, multiplique *f* sucesivamente por cada valor de *k* y almacene el resultado en *f*.
   e. El resultado de un factorial puede ser muy grande aunque el valor de entrada sea pequeña. Asegúrese de que todos los cálculos de enteros están en un bloque de comprobación (checked) y de que se capturan excepciones como el desbordamiento aritmético.
   f. Asigne el valor del resultado en *f* al parámetro de salida *respuesta*.
   g. Devuelva **true** desde el método si el cálculo se realiza sin problemas, y **false** en caso contrario (es decir, si se produce una excepción).

El código para la clase **Utils** debería ser como éste:

```
namespace Utils
{
    using System;

    public class Utils
    {

    ... código anterior omitido para mayor claridad ...

    //
    // Calcular factorial
    // y devolver el resultado como un parámetro out
    //

    public static bool Factorial(int n, out int respuesta)
    {
        int k;        // Contador del bucle
        int f;        // Valor de trabajo
        bool ok=true; // True si está bien, false si no

        // Comprobar el valor de entrada

        if (n<0)
            ok = false;

        // Calcular el valor del factorial como el
        // producto de todos los números de 2 a n

        try
        {
            checked
            {
                f = 1;
                for (k=2; k<=n; ++k)
                {
                    f = f * k;
                }
            }
        }
        catch(Exception)
        {
            // Si hay algún problema en el cálculo, se
            // capturará aquí. Todas las excepciones se
            // tratan de la misma manera: poner el
            // resultado a cero y devolver false.
```

El código continúa en la página siguiente.

```
                f = 0;
                ok = false;
            }


            // Asignar el valor del resultado
            respuesta = f;
            // Devolver al llamador
            return ok;
        }


    }
}
```

Cómo probar el método Factorial

1. Edite la clase **Test** como se indica a continuación:
   a. Declare una variable **bool** llamada *ok* para el resultado **true** o **false**.
   b. Declare una variable **int** llamada *f* para el resultado del factorial.
   c. Pida un entero al usuario. Asigne el valor de entrada a la variable **int** *x*.
   d. Haga una llamada al método **Factorial**, pasando *x* como primer parámetro y *f* como segundo parámetro. Devuelva el resultado en *ok*.
   e. Si *ok* es **true**, muestre los valores de *x* y *f*; en caso contrario, muestre un mensaje para indicar que se ha producido un error.

El código para la clase **Test** debería ser como éste:

```
namespace Utils
{
    public class Test
    {

    static void Main( )
    {
        int f;       // Resultado del factorial
        bool ok;     // Factorial correcto o fallido

        ... código anterior omitido para mayor claridad ...

        // Obtener entrada para factorial

        Console.WriteLine("Número para factorial:");
        x = int.Parse(Console.ReadLine( ));

        // Probar la función factorial
        ok = Utils.Factorial(x, out f);
        // Enviar los resultados del factorial a la salida
        if (ok)
            Console.WriteLine("Factorial(" + x + ") = " +
f);
        else
            Console.WriteLine("No se puede calcular este
➥factorial");
    }
    }
}
```

2. Guarde el trabajo realizado.
3. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.

Si el tiempo lo permite
Método con recursión

En este ejercicio reescribirá el método **Factorial** creado en el Ejercicio 3 utilizando recursión en lugar de un bucle.

El factorial de un número se puede definir recursivamente de la siguiente manera: el factorial de cero es 1, y el factorial de cualquier otro entero más grande se puede calcular multiplicando ese entero por el factorial del número anterior. En resumen:

Si n = 0, entonces Factorial(n) = 1; en los demás casos, Factorial(n) = n * Factorial(n-1)

Cómo modificar el método Factorial anterior

1. Edite la clase **Utils** y modifique el método **Factorial** para que emplee recursión en lugar de iteración.

   Los parámetros y tipos de retorno serán los mismos, pero cambiará el funcionamiento interno del método. Si desea conservar la solución del Ejercicio 3, tendrá que utilizar otro nombre para este método.
2. Utilice el pseudocódigo mostrado más arriba para escribir el cuerpo del método **Factorial** (tendrá que convertirlo a la sintaxis de C#).
3. Añada código a la clase **Test** para probar el nuevo método.
4. Guarde el trabajo realizado.

5. Compile el proyecto y corrija los posibles errores. Ejecute y pruebe el programa.

   La versión recursiva del método **Factorial** (**RecursiveFactorial**) se muestra a continuación:

```
//
// Otra forma de resolver el problema del factorial,
// esta vez como una función recursiva
//

public static bool FactorialRecursivo(int n, out int f)
{
    bool ok=true;

    // Capturar entradas negativas
    if (n<0)
    {
            f=0;
            ok = false;
    }

    if (n<=1)
        f=1;
    else
    {
        try
        {
            int pf;
            checked
            {
                ok = FactorialRecursivo(n-1,out pf);
                f = n * pf;
            }
        }
        catch(Exception)
        {
            // Algo ha ido mal. Poner indicador
            // de error y devolver cero.
            f=0;
            ok=false;
        }

    }

    return ok;
}
```