Laborator 4 – Metode și parametri

Obiective

După completarea acestui laborator veți dobândi următoarele cunoștințe:

- Crearea și apelarea metodelor cu și fără parametri
- Utilizarea diferitelor mecanisme de transmitere a parametrilor

Condiţii prealabile

Înainte de a realiza acest laborator trebuie să fiţi familiarizaţi cu următoarele concepte:

- Crearea unei clase şi a unei metode **Main()** în interiorul acesteia
- Crearea și utilizarea variabilelor
- Instrucțiuni C#

➢ Exerciţiul 1- Utilizarea parametrilor în metode ce returnează valori

În acest exerciţiu veţi defini și utiliza parametri de intrare, într-o metodă ce returnează o valoare. De asemenea, veți scrie un *framework* de testare, care citește două valori introduse de la consolă și care va afișa rezultatul.

Veți crea o clasă numită **Utils** în interiorul căreia veți avea o metodă **Greater**. Această metodă va primi doi parametri de intrare de tip **int** și va returna valoarea parametrului mai mare.

Pentru a testa metoda veți crea o altă clasă numită **Test** care va cere utilizatorului să introducă, de la tastatură, valorile celor două numere, după care va apela metoda **Utils.Greater** și va afișa rezultatul obținut în urma evaluării acesteia.

Crearea metodei Greater

- Creați o clasă **Utils**, la fel ca în laboratorul precedent, și adăugați o metoda publică și statică în interiorul acesteia numită **Greater**.
- Metoda va primi doi parametri de intrare, numiți **a** și **b,** care vor fi transmiși prin valoare. Metoda va returna o valoarea întreagă, care reprezintă cel mai mare număr dintre cele două primite ca parametri.

```
namespace Utils
{
      using System;
      class Utils
      {
            //
            // Return the greater of two integer values
            //
            public static int Greater(int a, int b)
            {
            }
      }
}
```

Testarea metodei Utils.Greater()

- Creați o clasă **Test** ce conține o metodă **Main()**.
- În interiorul metodei **Main(),** declarați două variabile întregi, numite **x** și **y**. Utilizați instrucțiunea de citire de la tastatură (**Console.ReadLine()**) pentru a atribui valori lui **x** și **y**. Nu uitați de metoda **int.Parse(),** prezentată în ședința precedentă.
- Declarați o altă variabilă întreagă numită **greater**. Păstrați, în aceasta, rezultatul obținut în urma evaluării metodei **Greater(),** apelată pentru cei doi întregi **x** și **y** citiți de la tastatură.
- Afișați valoarea variabilei **greater** utilizând funcția de *output* **Console.WriteLine()**.

```
namespace Utils
{
      using System;
      /// <summary>
      /// This the test harness
      /// </summary>

      public class Test
      {
            public static void Main( )
            {
            }
      }
}
```

➢ Exercițiul 2 – Utilizarea metodelor cu parametri referință

În acest exercițiu veți scrie o metodă numită Swap(), care va interschimba valorile parametrilor săi. Veți utiliza parametri transmiși prin referință.

Crearea metodei Swap

- Adăugați la clasa creată la exercițiul anterior – **Utils** – o metodă numită **Swap** care este statică, publică și nu întoarce nicio valoare (de tip **void**).
- Metoda va primi doi parametri **a** și **b**, de tip **int** care vor fi transmiși prin referință.
- Scrieți instrucțiunile necesare în interiorul metodei **Swap** care realizează interschimbarea celor două valori **a** și **b**. Pentru aceasta veți mai avea nevoie de o variabilă locală, **temp**, care va păstra una dintre cele două valori în timpul interschimbării.

```
namespace Utils
{
      using System;
      public class Utils
      {
            ... existing code omitted for clarity ...
            //
            // Exchange two integers, passed by reference
            //
            public static void Swap(ref int a, ref int b)
            {
            }
      }
}
```

Testarea metodei Swap

- Editați metoda **Main()** din clasa **Test** creată la Exercițiul 1 pentru a executa următorii pași:
  - Popularea cu valori a variabilelor **x** și **y**;
  - Apelarea metodei **Swap** transmițând aceste variabile ca parametrii referință.
- Afișați valorile lui **x** și **y** înaintea și după ce are loc interschimbarea.

- Salvați ce ați lucrat, compilați și rulați programele.

```
namespace Utils
{
      using System;
      public class Test
      {
            public static void Main( )
            {
                  ... existing code omitted for clarity ...
                  // Test the Swap method
            }
      }
}
```

➢ Exerciţiul 3 – Utilizarea metodelor cu parametri de ieșire

În acest exerciţiu veţi defini și utiliza o metodă statică ce va avea un parametru de ieșire.
Veți crea o nouă metodă, numită **Factorial,** care primește o valoarea întreagă și calculează factorialul acesteia. Factorialul unui număr este produsul dintre toate numerele întregi cuprinse între 1 și valoarea numărului respectiv. Factorialul lui 0 este prin definiție 1.
Următoarele sunt niște exemple de factoriale.

Factorial (x)	Valoare
x = 0	1
x = 1	1
x = 2	2
x = 3	6
x = 4	24

Crearea metodei Factorial

- Adăugați o nouă metodă statică și publică, numita **Factorial,** la clasa **Utils,** creată la Exercițiul 1.
- Această metodă va primi doi parametri întregi, **n** și **answer**. Primul parametru transmis prin valoare reprezintă numărul pentru care se va calcula factorialul. Al doilea parametru este de ieșire (**out int**) și va fi folosit pentru a păstra rezultatul.
- Metoda **Factorial()** va returna o valoare de tip **bool** care va indica dacă metoda a terminat calculul corespunzător.
- Cel mai ușor mod de a calcula factorialul este prin utilizarea unui ciclu. Realizați următorii pași pentru a adăuga funcționalitate metodei:
  - Declarați o variabilă **k** în metodă. Aceasta va fi folosită ca iterator în ciclu.
  - Declarați încă o variabilă **f** care va păstra produsele parțiale specifice fiecărui pas din ciclu. Inițial această variabilă este inițializată cu 1.
  - Folosiți instrucțiunea **for** pentru a implementa iterarea. Variabila **k** va fi cuprinsă între 2 și **n**, iar la fiecare pas va crește cu o unitate.
  - La fiecare pas, înmulțiți **f** cu valoarea curentă a lui **k** și păstrați rezultatul tot în **f**.
  - Rezultatul unui factorial poate fi foarte mare chiar pentru valori de intrare destul de mici. De aceea, asigurați-vă că toate calculele cu întregi se află într-un bloc de verificare în care veți putea trata erorile precum depășirea aritmetică.
  - Atribuiți lui **answer** valoarea lui **f**.
  - Returnați **true** dacă calculul s-a produs fără nicio eroare (valoare negativă a lui **n** sau depășire aritmetică) sau **false** în caz contrar.

```
namespace Utils
{
      using System;
      public class Utils
      {
            ... existing code omitted for clarity ...
            //
            // Calculate factorial
            // and return the result as an out parameter
            //
            public static bool Factorial(int n, out int answer)
            {
                  // Check the input value


(continuare – pagina următoare)
```

(**continuare**)

```
                // Calculate the factorial value as the
                // product of all of the numbers from 2 to n

                try
                {
                      checked
                      {
                      }
                }

                catch(Exception)
                {
                      // If something goes wrong in the
                      calculation,
                      // catch it here. All exceptions
                      // are handled the same way: set the result
                      // to zero and return false.
                }
                // Assign result value
                // Return to caller
          }
     }
}
```

Testarea metodei Factorial

- Editați metoda **Main()** din clasa **Test** creată la Exercițiul 1 după cum urmează:
  - Creați o variabilă de tip **bool** numită **ok** care să păstreze rezultatul returnat de **Factorial**.
  - Declarați o variabilă întreagă, numită **f,** care să păstreze valoarea factorialului.
  - Cereți introducerea unui număr de la tastatură și atribuiți această valoarea lui **x**.
  - Apelați metoda **Factorial()** cu parametrii **x** (primul) și **f** (al doilea). Păstrați rezultatul obținut în **ok**.
  - Dacă **ok** este **true** atunci afișați valorile lui **x** și **f,** altfel afișați un mesaj de eroare.
- Compilați și rulați programul.

```
static void Main( )

{

      ... existing code omitted for clarity ...
      // Get input for factorial
      // Test the factorial function
      // Output factorial results
}
```

➢ Exercițiul 4 – Implementarea unei metode utilizând recursivitatea

În acest exercițiu veți reimplementa metoda **Factorial** de la Exercițiul 3 utilizând recursivitatea în loc de iterare cu ajutorul lui **for**.

Factorialul poate fi definit recursiv prin următoarea definiție:

*Dacă n=0, atunci Factorial(n) = 1; altfel este n * Factorial(n-1)*

Modificarea metodei Factorial

- În clasa **Utils** modificați metoda **Factorial** astfel încât să folosească recursivitatea în loc de iterare. Parametrii și tipul de returnare vor fi la fel ca anterior cu excepția lui **f** care va fi parametru de tip **ref**. Modificarea va apărea în funcționalitatea metodei. Dacă vreți să păstrați și metoda de la Exercițiul 3 modificați-i numele sau redenumiți altfel metoda de la exercițiul acesta.