

Systemnahes Programmieren

Übungsblatt 1

Übung 1.2: Grüezi-Konsole

```
class Grüezi
{
    static void Main(string[] args)
    {
        Grüezi g = new Grüezi();
        g.Meldung("Grüezi\n", false);
        g.Meldung("Grüezi\n", true);
        Console.Read();
    }

    private void Meldung(string msg, bool upperCase)
    {
        if (upperCase)
        {
            //msg in Grossbuchstaben umwandeln
            //und wieder in msg zurückschreiben
            msg = msg.ToUpper();
        }
        else
        {
            //msg in Kleinbuchstaben umwandeln
            //und wieder in msg zurückschreiben
            msg = msg.ToLower();
        }
        Console.WriteLine(msg);
    }
}
```



Übung 1.3: Konstruktor

```
class ClassA
{
    static void Main(string[] args)
    {
        //Testen Sie die Klassen mit folgendem Main-Programm:
        ClassA A = new ClassA("A");
        ClassB B0 = new ClassB();
        ClassB B1 = new ClassB("B1");
        ClassB B2 = new ClassB("B2", 2);

        //Leiten Sie nun ClassB von ClassA ab
        //und erweitern Sie das Main-Programm wie folgt:
        ClassA AA = new ClassA("AA");
        ClassA AB = new ClassB("AB");

        Console.Read();
    }
}
```

```

    /// <summary>
    /// Konstruktor ohne Parameter. (Musste hinzugefügt werden,
    /// damit ClassB abgeleitet werden kann von ClassA).
    /// </summary>
    public ClassA()
    {
        Console.WriteLine("DEBUG INFO: ClassA() instanziiert");
    }

    public ClassA(string txt)
    {
        Console.WriteLine("DEBUG INFO: ClassA(string txt) instanziiert");
        Console.WriteLine("A:"+txt+" ");
    }
}

class ClassB : ClassA
{
    private string msg;
    private int count;

    /// <summary>
    /// Konstruktor ohne Parameter
    /// </summary>
    public ClassB()
    {
        Console.WriteLine("DEBUG INFO: ClassB() instanziiert");
        this.msg = "B0";
        this.count = 0;
    }

    /// <summary>
    /// Konstruktor mit 1 Parameter
    /// </summary>
    /// <param name="msg">Nachricht</param>
    public ClassB(string msg):this()
    {
        Console.WriteLine("DEBUG INFO: ClassB(string msg) instanziiert");
        this.msg = msg;
        this.count = 1;
    }

    /// <summary>
    /// Konstruktor mit 2 Parameter
    /// </summary>
    /// <param name="msg">Nachricht</param>
    /// <param name="count">Anzahl</param>
    public ClassB(string msg, int count):this(msg)
    {
        Console.WriteLine("DEBUG INFO: ClassB(string msg, int count) instanziiert");
        Console.WriteLine("B:" + msg);
        for (int i = 0; i < count; i++)
            Console.Write(".");
        Console.WriteLine("");
    }
}
}

```

```

ex file:///D:/Visual Studio 2005/Projects/PRGSY Konstruktoren Projekt/PR
DEBUG INFO: ClassA(string txt) instanziiert
A:A
DEBUG INFO: ClassA() instanziiert
DEBUG INFO: ClassB() instanziiert
DEBUG INFO: ClassA() instanziiert
DEBUG INFO: ClassB() instanziiert
DEBUG INFO: ClassB(string msg) instanziiert
DEBUG INFO: ClassA() instanziiert
DEBUG INFO: ClassB() instanziiert
DEBUG INFO: ClassB(string msg) instanziiert
DEBUG INFO: ClassB(string msg, int count) instanziiert
B:B2
...
DEBUG INFO: ClassA(string txt) instanziiert
A:AA
DEBUG INFO: ClassA() instanziiert
DEBUG INFO: ClassB() instanziiert
DEBUG INFO: ClassB(string msg) instanziiert

```

Übung 1.4: Parameter

The **out** parameter can be used to return the values in the same variable passed as a parameter of the method. Any changes made to the parameter will be reflected in the variable.

The **ref** keyword on a method parameter causes a method to refer to the same variable that was passed as an input parameter for the same method. If you do any changes to the variable, they will be reflected in the variable.

The **out** keyword causes arguments to be passed by reference. This is similar to the **ref** keyword, except that **ref** requires that the variable be initialized before being passed. To use an **out** parameter, both the method definition and the calling method must explicitly use the **out** keyword.

When to use **ref** and **out** parameter: **out** parameter can be used when we want to return more than one value from a method. Use **ref** as a reference to another variable.

[http://msdn.microsoft.com/en-us/library/t3c3bfhx\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/t3c3bfhx(VS.80).aspx)

```
public class Test
{
    static void Main()
    {
        Test t = new Test();
        Console.Read();
    }

    public Test()
    {
        test1();
        test2();
    }

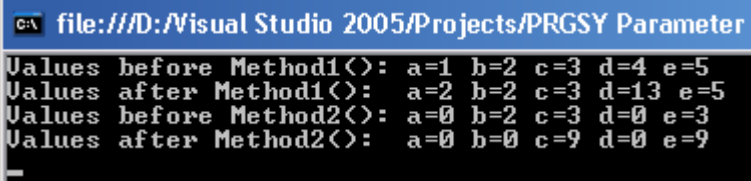
    private void test1()
    {
        int a = 1; int b = 2; int c = 3; int d = 4; int e = 5;
        Console.WriteLine("Values before Method1(): a={0} b={1} c={2} d={3} e={4}", a, b,
            c, d, e);
        Method1(ref a, out b, c, ref d, ref e);
        Console.WriteLine("Values after Method1(): a={0} b={1} c={2} d={3} e={4}", a, b,
            c, d, e);
    }

    public void Method1(ref int a, out int b, int c, ref int d, ref int e)
    {
        a = a + 1;
        b = 2;
        c = 2 * d;
        d = c + e;
    }

    private void test2()
    {
        C a = new C();
        C b = new C(); b.x = 2;
        C c = new C(); c.x = 3;
        C d = new C();
        C e = c;
        Console.WriteLine("Values before Method2(): a={0} b={1} c={2} d={3} e={4}", a.x,
            b.x, c.x, d.x, e.x);
        Method2(out a, ref b, c, d, e);
        Console.WriteLine("Values after Method2(): a={0} b={1} c={2} d={3} e={4}", a.x,
            b.x, c.x, d.x, e.x);
    }
}
```

```
public void Method2(out C a, ref C b, C c, C d, C e)
{
    b = new C();
    a = b;
    c = b;
    d = new C(); d.x = 7;
    e.x = 9;
}

public class C
{
    public int x;
}
}
```



```
c:\ file:///D:/Visual Studio 2005/Projects/PRGSY Parameter
Values before Method1(): a=1 b=2 c=3 d=4 e=5
Values after Method1(): a=2 b=2 c=3 d=13 e=5
Values before Method2(): a=0 b=2 c=3 d=0 e=3
Values after Method2(): a=0 b=0 c=9 d=0 e=9
```