Code Policy document

Social media in the Process Automation Industry

Version 1.0

Revision History

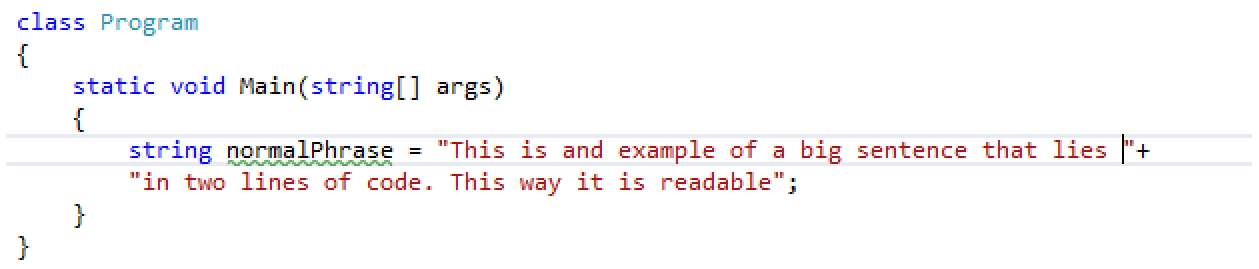
Initials	*Action*	*Date*	*Version*
RG, DK	Initial draft, added guidelines for beginners	8.11.2013	0.01

Formatting and Style

Visual Studio default Text Editor settings are configured to insert spaces for tabs. The default value is 4 spaces per tab.

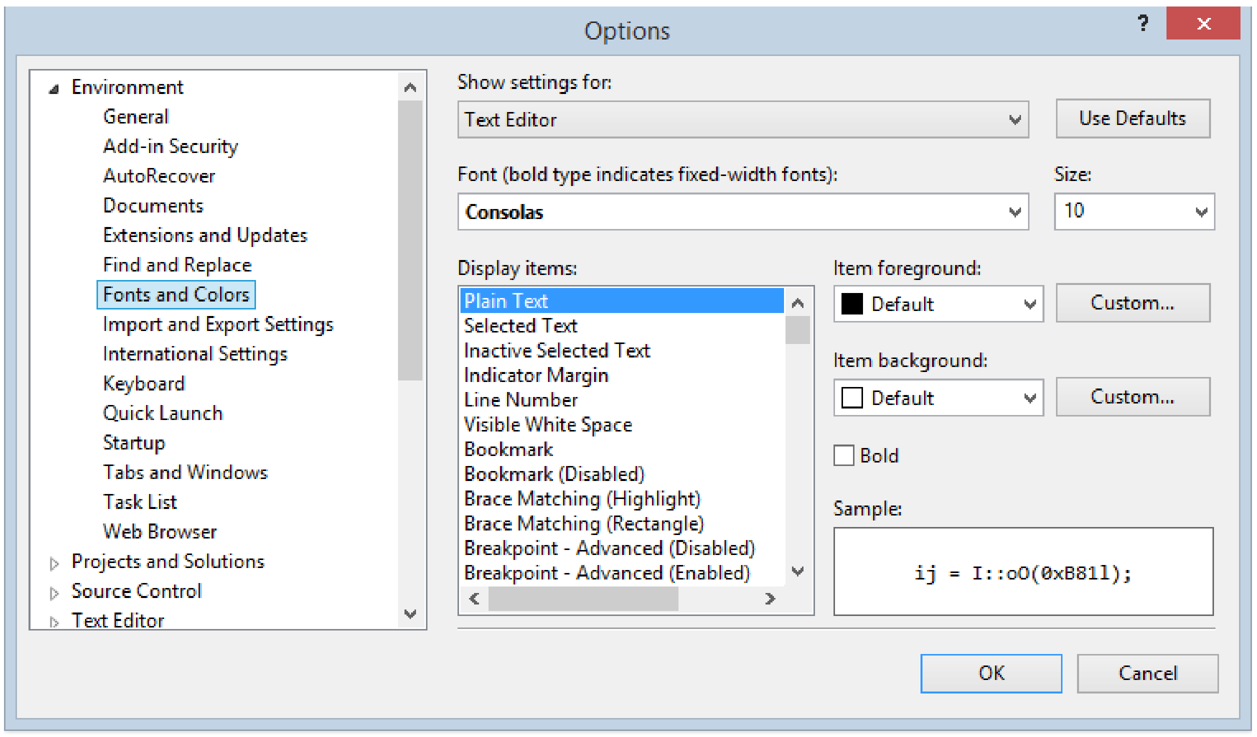
Readability is also an issue that could affect part of the code. So, a limit should be set to the maximum length of chars per line. In order to prevent this, the developer can change the line "enter" and then add indent with "Tab".

Figure 1 below, spreads the code in two lines and keeps the limit in the 88th column.



```
class Program
{
    static void Main(string[] args)
    {
        string normalPhrase = "This is and example of a big sentence that lies "+
        "in two lines of code. This way it is readable";
    }
}
```

When it comes to styling Visual Studio 2012 has as its default font Consolas and font size 10.



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): **Consolas**

Size: 10

Item foreground: Default

Item background: Default

☐ Bold

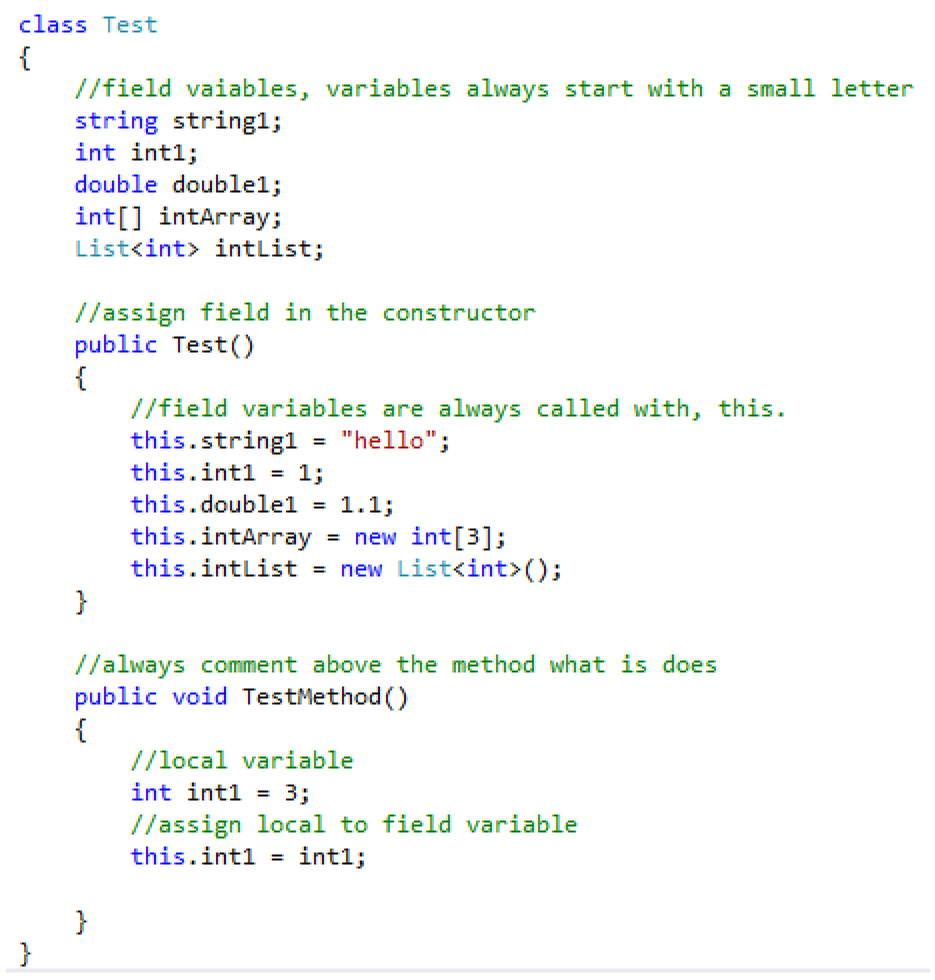
Sample:

```
ij = I::oO(0xB81l);
```

Local / Field variables

Field variables should be in the top of the class and should be initialized in the constructor. Local variables should be initialized when they are declared and preferably this should be done in one line of code.

Variables always start with a small letter and cameCase style should be used (e.g. intArray).



```
class Test
{
    //field vaiables, variables always start with a small letter
    string string1;
    int int1;
    double double1;
    int[] intArray;
    List<int> intList;

    //assign field in the constructor
    public Test()
    {
        //field variables are always called with, this.
        this.string1 = "hello";
        this.int1 = 1;
        this.double1 = 1.1;
        this.intArray = new int[3];
        this.intList = new List<int>();
    }

    //always comment above the method what is does
    public void TestMethod()
    {
        //local variable
        int int1 = 3;
        //assign local to field variable
        this.int1 = int1;

    }
}
```

Properties

All variables that should be available outside of the class, should have a property. Properties can have both get and set, or just one of them, depending on the wanted functionality. Properties starts with a capital letter.

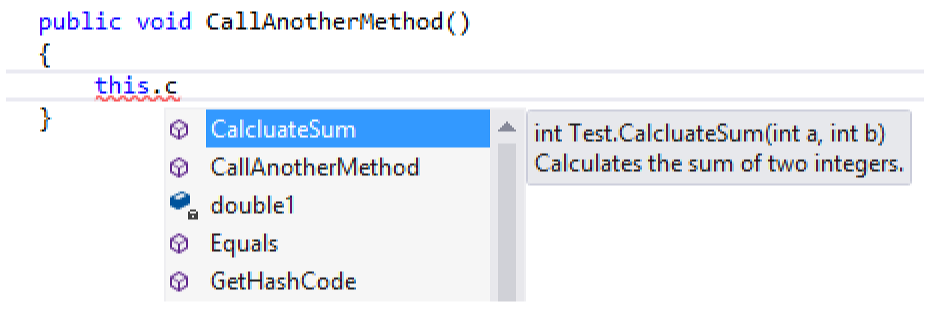
```
class Test
{
    //private variable
    string string1;

    //property, to make the private variable available outside of the class
    public string String1
    {
        get { return string1; }
        set { string1 = value; }
    }
}
```

Methods

A method always start with capital letter and should have a name so you understand what it does. If the method is getting too big, create help methods to separate code and make it more readable. Furthermore, the method should be commented using three / and the summary should be filled in. This makes it a lot easier for the developers using your code. Because, when he should call the method, the retrieve the summary inside of intelli sense. If you type / three times above the method after you written the it, visual studio will auto generate the <summary> for you, and also extra information. The extra information is not mandatory and can be deleted.

```
/// <summary>
/// Calculates the sum of two integers.
/// </summary>
public int CalcluateSum(int a, int b)
{
    return a + b;
}
```



```
public void CallAnotherMethod()
{
    this.c
}
```

int Test.CalcluateSum(int a, int b)
Calculates the sum of two integers.

Loops

```
public void TestLoops()
{
    int count = 0;
    //One line satement for-loop, no need for { }.
    for (int i = 0; i < 100; i++)
        count += i;

    // mutiple lines
    for (int i = 0; i < 100; i++)
    {
        count += i;
        count *= 10;
    }

    //one line, no need of { }
    while (count < 100)
        count = 100;

    //multiple lines
    while (count < 100)
    {
        count++;
        count = 100;
    }
}
```

Statements

Each statement should be placed in a single line and statements that are related should be distinguished with one blank line between them.

```
static void Numbers(){
    //seperate initialization
    int a = 32;
    int b = 40;
    int c = 20;

    //one blank line to seperate assignment
    a = c;
    b = c;

    int sum = 0;

    //one more blank line to seperate the for block
    for (int i = 0; i < 3; i++)
        sum += a;
}
```

Spaces

Spaces can be used to increase readability of code. They can be used like in Table X.

SumNumbers();	//no space between function name and parenthesis
CalculateProduct(n, 3, 4)	//single space after a comma
while (count < 100)	//single space before control flow statements
if (count < 100)	//single space separating operators
x = array[index];	//no space inside brackets

GUI control naming

Use the Hungarian notation when you are naming your GUI controls. A full list of examples could be found here:
http://support.microsoft.com/kb/173738

Reporting Bugs, Adds or Changes

When a bug is found in anothers code, there is something missing that you need or if there is something that has to be changed. Please insert the bug/add/change in the correct document here and follow the headings (date, type, class, method, description, solution (if any) and author).
https://drive.google.com/#folders/0B1hNhtSF9OMZR3hDal9hWUZQbm8

Before starting a coding session

Everytime before you start a new coding session, please check the documents where bugs/adds/changes are written down.
https://drive.google.com/#folders/0B1hNhtSF9OMZR3hDal9hWUZQbm8
To help others before start implementing something new is a good thing. When taking a request, write your name in responsible + the current status and when you are finished fixing, change the status to done.

If there is something unclear inside of the document, contact the author that wrote it.
Furthermore, update the GIT repository so it is up to date before you start.