Smart Coding …
(Part 1)

0011 0010 1010 1101 0001 0100 0011

Md. Mahedee Hasan

Software Architect

Leadsoft Bangladesh Limited

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

- **What** is Smart coding?
- **Why** Smart Coding?
- **How** Smart Coding?

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

What is Smart coding?

- Keep (K)
- It (I)
- Simple (S)
- Stupid (S)

keep the code as simple as possible

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Why Smart coding?

- Increase the **readability** of code
- **Hardly** any software is maintained for its whole life by the original **author**
- **80% of the lifetime cost** of a software goes to maintenance, the cleaner the code the easier it to maintain
- **Your source code is a product** you need to make sure it is as well-packaged and clean.

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

How Smart coding?

???

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Source file organization

- One Class per File
- Ordering
  - Using
  - Namespace
  - Class & Interface declaration

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Source file organization (cont.)

- Namespace and Using Statements

```
using System.Data;
namespace Business.Framework;
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Source file organization (cont.)

- Class & Interface declaration
  1. Class/ Interface documentation
  2. Class/ Interface statement
  3. Fields : private, protected, internal, public
  4. Properties : private, protected, internal, public
  5. Constructor : private, protected, internal, public
  6. Methods : Should be grouped by functionality rather than accessibility.

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Indentation

- Always use **tab** for indentation
- Wrapping Line
  1. Break after an operator
  2. Break after a comma
  3. Prefer higher level break than lower level
  4. Indent once after a break

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Indentation (cont.)

- Example of breaking a method call
  1. Preferred:

```
longName1 = longName2 * (longName3 + longName4 - longName5) +
                                4 * longname6;
```

  2. Avoid:

```
longName1 = longName2 * (longName3 + longName4
- longName5) + 4 * longname6;
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Indentation (cont.)

- Example of Indenting Method declaration

```
SomeMethod( int anArg,
Object anotherArg,
String yetAnotherArg,
Object andStillAnother)
{
...
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Indentation (cont.)

- Using If

```
if ((condition1 && condition2) ||
(condition3 && condition4) ||
!(condition5 && condition6))
{
DoSomethingAboutIt();
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Indentation (cont.)

- Using If (or)

```
if ((condition1 && condition2) || (condition3 && condition4) ||
!(condition5 && condition6))
{
        DoSomethingAboutIt();
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Indentation (cont.)

- Ternary Expression

```
alpha = (aLongBooleanExpression ? beta : gamma);

OR

alpha = (aLongBooleanExpression ?
beta :
gamma);
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Declaration

- One declaration per line is recommended

1. Preferred:

```
private int level = 2; // indentation level
private int size = 8; // size of table
```

2. Avoid:

```
private int level, size; // AVOID!!!
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Declaration (cont.)

- Initialize **local variable** where they are declared
- Put declaration at the beginning of the block

```
public void SomeMethod()
{
        int int1 = 0; // Beginning of method block.
        if (condition)
        {
                int int2 = 0; // Beginning of "if" block.
        ...
        }
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Declaration (cont.)

- The one **exception** to the rule is indexes of **for loops**, which in C# can be declared in the for statement:

```
for (int i = 0; i < maxLoops; i++)
{
        // Do something
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Declaration (cont.)

- For declaring Class & Interface do the following:
  1. **No space between a method name** and the parenthesis "(" starting its parameter list
  2. **Open brace "{" appears** at the beginning of the line following declaration statement and is indented to the beginning of the declaration.

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Declaration (cont.)

- For declaring Class & Interface do the following:

3. Closing brace "}" starts a line by itself indented to match its corresponding opening statement.
4. For null statements, the "}" should appear immediately after the "{" and both braces should appear on the same line as the declaration with 1 blank space separating the parentheses from the braces

```
public class Sample
{
        private int i = 0;
        public Sample()
        {
        }
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Declaration (cont.)

- Properties:
  1. If the body of the get or set method of a property consists of a **single statement**, the statement is written on the same line as the method signature.

```
public int Foo
{
        get { return this.foo; }
        set { this.foo = value; }
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements

- Simple Statements: **Each line** should contain at most **one statement**

1. Preferred:

```
argv++; // Correct
argc--; // Correct
```

2. Avoid:

```
argv++; argc--;
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- Compound Statements
  1. The enclosed statements should be indented one **more level than the compound statement.**
  2. The opening brace should be at the beginning of the line following the line that begins the compound statement and **be indented** to the beginning of the compound statement. The closing brace should begin a line and be indented to the beginning of the compound statement.
  3. Braces are used around all statements, even single statements, when they are part of a control structure, such as a if-else or for statement. This makes it easier to add statements without accidentally introducing bugs due to forgetting to add braces.

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- Return Statements
  1. A return statement with a value **should not use parentheses** unless they make the return value more obvious in some way.

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- if, if-else, if else-if else Statements

(Use following form)

```
if (condition)
{
        statements;
}
if (condition)
{
        statements;
}
else
{
        statements;
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- if, if-else, if else-if else Statements

```
if (condition)
{
        statements;
}
else if (condition)
{
        statements;
}
else
{
        statements;
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- if, if-else, if else-if else Statements (avoid the following)

```
if (condition) //AVOID! THIS OMITS THE BRACES {}!
        statement;
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- for Statements (should have the following form)

```
for (initialization; condition; update)
{
        Statement;
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- while Statements (should have the following form)

```
while (condition)
{
        statements;
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- do-while Statements (should have the following form)

```
do
{
        statements;
} while (condition);
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- switch Statements (should follow the following form at next page)

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

```
switch (condition)
{
        case 1:
                // Falls through.
        case 2:
                statements;
                break;
        case 3:
                statements;
                goto case 4;
        case 4:
                statements;
                break;
        default:
                statements;
                break;
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

Statements (cont.)

- Try-catch Statements

```
try
{
        statements;
}
catch (ExceptionClass e)
{
        statements;
}
finally
{
        statements;
}
```

Smart Coding …

0011 0010 1010 1101 0001 0100 1011

???

Thank You