

**Coding Standards**

University of North Alabama

CS 455

Software Engineering Project

Spring 2015

**Overview**

These coding standards have been developed by the Tune Squad to aid in developing a reliable and robust program. The standards focus on the C# programming language. Included in these standards are details about: (1) File organization; (2) Document Styles; (3) General Code Style; (4) Statements; (5) Comments; (6) Naming Conventions. Following the details about the styles is an example. Every developer working on the L.I.F.E.S. project will read the coding standards and review the example before any code is written.

GitHub shall be used for source code revision control. The Tune Squad’s repository on Github will be a point of reference for all testing guidelines.

The default font for text shall be Times New Roman Size 12 (Excluding the Specification Document). Source code placed in documents shall be in Consolas size 10.

**File Organization**

C# Source Files

Every class will be contained in a separate file. For example, class ExampleClasswill be contained in a file named ExampleClass.cs.

Directory Layout

There will be a directory created for every namespace. For example, FirstNameSpace.SecondNameSpacewill be contained inside of FirstNameSpace/SecondNameSpace.

**General**

A tab shall equal 4 spaces.

No line shall be longer than 80 characters.

All variables shall be declared before used.

All function/methods will begin with a capital letter.

All variables will begin with a lowercase letter.

All global variables, constants, and defined names shall be in all caps. Multiple words shall be separated with an underscore.

int CONTANT\_VAR = 1;

**Statements**

Write only one statement per line.

Write only one declaration per line.

If continuation lines are not indented automatically, indent them one tab stop (four spaces).

There will be one blank line before a method header and the method will follow directly after the header without a blank line.

Trailing braces shall be used everywhere on a new line (if, else-if, else, functions, structures, typedef, class definitions, etc.)

if (true)   
{

}

The else statement shall start on a new line after the last closing brace with one blank line in-between.

if ( x )   
{

}   
  
else if (y)   
{

}   
  
else   
{

}

Parenthesized expressions will not be padded with spaces.

if (x)   
{

}

And

x = y \* (z + 5);

Not

if ( x )   
{

}

And

x = y \* ( z + 5 );

There will be exactly one space between operators.

x = y + z \* a;

not

x=y+z\*a;

**Commenting**

Place the comment on a separate line, not at the end of a line of code.

Begin comment text with an uppercase letter.

End comment text with a period.

Insert one space between the comment delimiter (//) and the comment text, as shown in the following example.

// This is a sample comment.

The use of multi-line comments is allowed.

/\*

\* Multi-line comments

\* can be used.

\*/

All methods will have a header of the following format:

/\*

\* Method: Name

\* Parameters: number and type of parameters

\*

\* Description: Describe the purpose of the method.

\*

\*/

All classes will have a header of the following format:

/\*

\* ClassName.cs

\*

\* Description:

\*

\*/

**Naming**

All namespace, class, and method names will use PascalCasing. This convention capitalizes the first character of each word. For example:

public int MethodName()

File names shall also use PascalCasing. For example:

FileName.cs

Namespace components will be separated with periods.

 Microsoft.Office.PowerPoint

All variable names will use camelCasing. This convention ensures that the first letter of the name is lower case and each successive word starts with an upper case letter. For example:

int varName;

Ordering of class variables and methods shall be as follows:

1. public variables
2. protected variables
3. private variables
4. public methods
5. protected methods
6. private methods

Use predefined type names instead of system type names.

// Correct  
string firstName;  
int lastIndex;  
bool isSaved;

// Avoid  
String firstName;  
Int32 lastIndex;  
Boolean isSaved;

**Example Brace Placement**

namespace WhereIsTheBracket

{

public enum Test

{

TestMe,

TestYou

}

public class TestMeClass

{

Test test;

public Test Test

{

get

{

return test;

}

set

{

test = value;

}

}

void DoSomething()

{

if (test == Test.TestMe)

{

// ...stuff gets done

}

else if (test == Test.TestThat)

{

// ...other stuff gets done

}

else {

// ...some other stuff gets done

}

}

}

}