**Coding Standards**

Our team will be using C# with Visual Studios.

**Naming conventions**

* Use camel case, starting with upper case for custom methods

Ex. MethodNumberOne

* Use camel case starting with lower case for GUI elements such as buttons, windows, etc.

Ex. btnEnter

* Event handlers have the standard visual studio naming convention composed of the GUI element name and the event that is being handled

Ex. btnEnter\_Click

* Capitalize class names
* Interface names start with a capital I

Ex. IComparable

* Final variables are in all caps with underscores separating words

Ex. final int MY\_INT

* Field names for custom objects have an underscore preceding them

Ex. \_myField

**Format**

* Place fields at the top whenever possible
* Methods start after fields
* Constructors come first
* Getters/setters are on their own single line when possible
* Fields go in the following order:

Static, const, private, internal, public

* No public fields within custom objects, use gets whenever possible
* Function main goes at the bottom

**Whitespace / Bracket placement**

* No space between method name and parenthesis
* Method parameters are separated by both a comma and a single space
* Operators are spaced apart, except in the case of using increment or decrement operators
* Standard use of tabs
* Opening bracket is on the line below the method declaration
* One line conditionals do not require brackets

Ex.

MethodNumberOne(int x, int y)

{

if(x == y)

x++;

x = x + y;

}

**Comment standards**

* Comments on the line above any method that is complex or who’s function/purpose may be unclear
* Comments after the closing bracket of any complex or lengthy methods

Ex.

// this method does nothing because it’s an example

ComplexMethod( … )   
{

.

.

.

(more lines)

}// end complexMethod

**Modularity considerations**

* Modularity for readability
* Break lengthy methods into smaller methods