Coding Guidelines for C# 3.0, 4.0 and 5.0 Cheat Sheet

Design & Maintainability (level 1 and 2 only)

Basic Principles

- The Principle of Least Surprise
- Keep It Simple Stupid
- You Ain't Gonne Need It
- Don't Repeat Yourself

Class Design

- A class or interface should have a single purpose (AV1000)
- An interface should be small and focused (AV1003)
- Use an interface to decouple classes from each other (AV1005)
- Don't hide inherited members with the `new` keyword (AV1010)
- It should be possible to treat a derived object as if it were a base class object (AV1011)
- Don't refer to derived classes from the base class (AV1013)
- Avoid exposing the objects an object depends on (AV1014)
- Avoid bidirectional dependencies (AV1020)
- Classes should have state and behavior (AV1025)

Member Design

- Allow properties to be set in any order (AV1100)
- Don't use mutual exclusive properties (AV1110)
- A method or property should do only one thing (AV1115)
- Don't expose stateful objects through static members (AV1125)
- Return an `IEnumerable<T>` or `ICollection<T>` instead of a concrete collection class (AV1130)
- Properties, methods and arguments representing strings or collections should never be `null` (AV1135)
- Define parameters as specific as possible (AV1137)

Miscellaneous Design

- Throw exceptions rather than returning status values (AV1200)
- Provide a rich and meaningful exception message text (AV1202)
- Don't swallow errors by catching generic exceptions (AV1210)
- Always check an event handler delegate for `null` (AV1220)
- Properly handle exceptions in asynchronous code (AV1215)
- Use a protected virtual method to raise each event (AV1225)
- Don't pass `null` as the sender parameter when raising an event (AV1235)
- Use generic constraints if applicable (AV1240)
- Evaluate the result of a LINQ expression before returning it (AV1250)

Maintainability

- Methods should not exceed 7 statements (AV1500)
- Make all members `private` and types `internal` by default (AV1501)
- Avoid conditions with double negatives (AV1502)
- Don't use "magic numbers" (AV1515)
- Only use `var` when the type is very obvious (AV1520)
- Declare and initialize variables as late as possible (AV1521)
- Favor Object and Collection Initializers over separate statements (AV1523)
- Don't make explicit comparisons to `true` or `false` (AV1525)
- Don't change a loop variable inside a `for` or `foreach` loop (AV1530)
- Avoid nested loops (AV1532)
- Always add a block after keywords such `if`, `else`, `while`, `for`, `foreach` and `case` (AV1535)
- Always add a `default` block after the last `case` in a `switch` statement (AV1536)
- Finish every `if-else-if`statement with an `else`-part (AV1537)
- Be reluctant with multiple `return` statements (AV1540)
- Don't use `if-else` statements instead of a simple (conditional) assignment (AV1545)
- Encapsulate complex expressions in a method or property (AV1547)
- Call the most overloaded method from other overloads (AV1551)
- Only use optional arguments to replace overloads (AV1553)
- Avoid using named arguments (AV1555)
- Don't allow methods and constructors with more than three parameters (AV1561)
- Don't use `ref` or `out` parameters (AV1562)
- Avoid methods that take a `bool` flag (AV1564)
- Always check the result of an `as` operation (AV1570)
- Don't comment-out code (AV1575)

Framework Guidelines

- Use C# type aliases instead of the types from the `System` namespace (AV2201)
- Build with the highest warning level (AV2210)
- Use Lambda expressions instead of delegates (AV2221)
- Only use the `dynamic` keyword when talking to a dynamic object (AV2230)
- Favor `async/await` over the `Task` (AV2235)

Dennis Doomen
Version 5.0.0 (October 18, 2016)

www.csharpcodingguidelines.com
www.continuousimprover.com
www.avivasolutions.nl

Coding Guidelines for C# 3.0, 4.0 and 5.0 Cheat Sheet

Naming & Layout (level 1 and 2 only)

Pascal Casing

Class, Struct	`AppDomain`
Interface	`IBusinessService`
Enumeration type	`ErrorLevel`
Enumeratiion values	`FatalError`
Event	`Click`
Protected field	`MainPanel`
Const field	`MaximumItems`
Read-only static field	`RedValue`
Method	`ToString`
Namespace	`System.Drawing`
Property	`BackColor`
Type Parameter	`TEntity`

Camel Casing

Private field	`listItem`
Variable	`listOfValues`
Const variable	`maximumItems`
Parameter	`typeName`

Naming

- Use US English (AV1701)
- Don't include numbers in variables, parameters and type members (AV1704)
- Don't prefix fields (AV1705)
- Don't use abbreviations (AV1706)
- Name members, parameters or variables according its meaning and not its type (AV1707)
- Name types using nouns, noun phrases or adjective phrases (AV1708)
- Don't repeat the name of a class or enumeration in its members (AV1710)
- Avoid short names or names that can be mistaken with other names (AV1712)
- Name methods using verb-object pair (AV1720)
- Name namespaces using names, layers, verbs and features (AV1725)
- Use an underscore for irrelevant lambda parameters (AV1739)

Documentation

- Write comments and documentation in US English (AV2301)
- Document all public, protected and internal types and members (AV2305)
- Avoid inline comments (AV2310)
- Only write comments to explain complex algorithms or decisions (AV2316)
- Don't use comments for tracking work to be done later (AV2318)

Layout

- Maximum line length is 130 characters.
- Indent 4 spaces, don't use Tabs
- Keep one white-space between keywords like `if` and the expression, but don't add white-spaces after `(` and before `)`.
- Add a white-space around operators, like `+`, `-`, `==`, etc.
- Always add parentheses after keywords `if`, `else`, `do`, `while`, `for` and `foreach`
- Always put opening and closing parentheses on a new line.
- Don't indent object initializers and initialize each property on a new line.
- Don't indent lambda statements
- Put the entire LINQ statement on one line, or start each keyword at the same indentation.
- Add braces around comparison conditions, but don't add braces around a singular condition.

Empty lines

- Between members
- After the closing parentheses
- Between multi-line statements
- Between unrelated code blocks
- Around the `#region` keyword
- Between the `using` statements of different root namespaces.

Member order

1. Private fields and constants
2. Public constants
3. Public read-only static fields
4. Factory Methods
5. Constructors and the Finalizer
6. Events
7. Public Properties
8. Other methods and private properties in calling order

Important Note

These coding guidelines are an extension to Visual Studio's Code Analysis functionalty, so make sure you enable that for all your projects. Check the full document for more details.

Dennis Doomen
Version 5.0.0 (October 18, 2016)

www.csharpcodingguidelines.com
www.continuousimprover.com
www.avivasolutions.nl