Coding Guidelines for C# 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 7.1, 7.2 and 7.3 Cheat Sheet

Design & Maintainability (level 1 and 2 only)

Basic Principles

- The Principle of Least Surprise
- Keep It Simple Stupid
- You Ain't Gonna Need It
- Don't Repeat Yourself
- OOP: Encapsulation, abstraction, inheritance, polymorphism

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 suppress compiler warnings using 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 other objects an object depends on (AV1014)
- Avoid bidirectional dependencies (AV1020)
- Classes should have state and behavior (AV1025)
- Classes should protect the consistency of their internal state (AV1026)

Member Design

- Allow properties to be set in any order (AV1100)
- Don't use mutually exclusive properties (AV1110)
- A property, method or local function 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, arguments and return values representing strings or collections should never be `null` (AV1135)
- Define parameters as specific as possible (AV1137)

Miscellaneous Design

- Throw exceptions rather than returning some kind of status value (AV1200)
- Provide a rich and meaningful exception message text (AV1202)
- Don't swallow errors by catching generic exceptions (AV1210)
- Properly handle exceptions in asynchronous code (AV1215)
- Always check an event handler delegate for `null` (AV1220)
- Use a protected virtual method to raise each event (AV1225)
- Don't pass `null` as the `sender` argument when raising an event (AV1235)
- Use generic constraints if applicable (AV1240)
- Evaluate the result of a LINQ expression before returning it (AV1250)
- Do not use `this` and `base` prefixes unless it is required (AV1251)

Maintainability

- Methods should not exceed 7 statements (AV1500)
- Make all members `private` and types `internal sealed` 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)
- Assign each variable in a separate statement (AV1522)
- 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` loop (AV1530)
- Avoid nested loops (AV1532)
- Always add a block after the keywords `if`, `else`, `do`, `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` clause (AV1537)
- Be reluctant with multiple `return` statements (AV1540)
- Don't use an `if-else` construct instead of a simple (conditional) assignment (AV1545)
- Encapsulate complex expressions in a property, method or local function (AV1547)
- Call the more overloaded method from other overloads (AV1551)
- Only use optional arguments to replace overloads (AV1553)
- Avoid using named arguments (AV1555)
- Don't declare signatures with more than 3 parameters (AV1561)
- Don't use `ref` or `out` parameters (AV1562)
- Avoid signatures that take a `bool` flag (AV1564)
- Prefer `is` patterns over `as` operations (AV1570)
- Don't comment out code (AV1575)

Framework Guidelines

- Use C# type aliases instead of the types from the `System` namespace (AV2201)
- Prefer language syntax over explicit calls to underlying implementations (AV2202)
- Build with the highest warning level (AV2210)
- Use lambda expressions instead of anonymous methods (AV2221)
- Only use the `dynamic` keyword when talking to a dynamic object (AV2230)
- Favor `async`/`await` over `Task` continuations (AV2235)

Coding Guidelines for C# 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 7.1, 7.2 and 7.3 Cheat Sheet

Naming & Layout (level 1 and 2 only)

Pascal Casing

Namespace	`System.Drawing`
Type parameter	`TView`
Interface	`IBusinessService`
Class, struct	`AppDomain`
Enum	`ErrorLevel`
Enum member	`FatalError`
Resource key	`SaveButtonTooltipText`
Constant field	`MaximumItems`
Private static readonly field	`RedValue`
Non-private field	`MainPanel`
Property	`BackColor`
Event	`Click`
Method	`ToString`
Local function	`FormatText`

Tuple element names	`(string First, string Last) name = ("John", "Doe");`
	`var name = (First: "John", Last: "Doe");`
	`(string First, string Last) GetName() => ("John", "Doe");`

Camel Casing

Private field	`listItem`
Parameter	`typeName`
Variables declared using tuple syntax	`(string first, string last) = ("John", "Doe");` `var (first, last) = ("John", "Doe");`
Local variable	`listOfValues`

Naming

- Use US English (AV1701)
- Don't prefix fields (AV1705)
- Don't use abbreviations (AV1706)
- Name members, parameters and variables according to their meaning and not their type (AV1707)
- Name types using nouns, noun phrases or adjective phrases (AV1708)
- Name generic type parameters with descriptive names (AV1709)
- Don't repeat the name of a class or enumeration in its members (AV1710)
- Avoid short names or names that can be mistaken for other names (AV1712)
- Properly name properties (AV1715)
- Name methods and local functions using verbs or verb-object pairs (AV1720)
- Use a verb or verb phrase to name an event (AV1735)
- Postfix asynchronous methods with `Async` or `TaskAsync` (AV1755)
- Use an underscore for irrelevant parameters (AV1739)

Documentation

- Write comments and documentation in US English (AV2301)
- Document all `public`, `protected` and `internal` types and members (AV2305)
- Write XML documentation with other developers in mind (AV2306)
- Avoid inline comments (AV2310)
- Only write comments to explain complex algorithms or decisions (AV2316)

Layout

- Maximum line length is 130 characters
- Indent 4 spaces, don't use tabs
- Keep one space between keywords like `if` and the expression, but don't add spaces after `(` and before `)`
- Add a space around operators, like `+`, `-`, `==`, etc.
- Always add curly braces after the keywords `if`, `else`, `do`, `while`, `for`, `foreach` and `case` (AV1535)
- Always put opening and closing curly braces on a new line
- Don't indent object/collection initializers and initialize each property on a new line
- Don't indent lambda statement blocks
- Keep expression-bodied-members on one line; break long lines after the arrow sign
- Put the entire LINQ statement on one line, or start each keyword at the same indentation
- Add parentheses around every binary expression, but don't add parentheses around unary expressions
- Be reluctant with #region (AV2407)
- Use expression-bodied members appropriately (AV2410)

Empty lines

- Between multi-line statements
- Between multi-line members
- After the closing curly brace
- 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 static readonly fields
4. Factory methods
5. Constructors and the finalizer
6. Events
7. Public properties
8. Other methods and private properties in calling order