# Design and Maintainability

## 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

An interface should be small and focused

Use an interface to decouple classes from each other

Don’t hide inherited members with the new keyword

It should be possible to treat a derived object as if it were a base class object

Don’t refer to derived classes from the base class

Avoid exposing the objects an object depends on

Avoid bidirectional dependencies

Classes should have state and behavior

## Member Design

Allow properties to be set in any order

Avoid mutual exclusive properties

A method or property should do only one thing

Don’t expose stateful objects through static members

Return an IEnumerable<T> or ICollection<T> instead of a concrete collection class

String, list and collection properties should never return a null reference

## Miscellaneous Design

Throw exceptions rather than returning status values

Provide a rich and meaningful exception message text

Don’t swallow errors by catching generic exceptions

Always check an event handler delegate for null

Use a protected virtual method to raise each event

Don’t pass null as the sender parameter when raising an event

Use generic constraints if applicable

Don’t add extension methods to the same namespace as the extended class

Evaluate the result of a LINQ expression before returning it

## Maintainability

Methods should not exceed 7 statements

Make all members private and types internal by default

Avoid conditions with double negatives

Don’t use "magic numbers"

Only use var when the type is very obvious

Initialize variables at the point of declaration

Favor Object and Collection Initializers over separate statements

Don’t make explicit comparisons to true or false

Don’t change a loop variable inside a for or foreach loop

Don’t use nested loops in a method

Add a block after all flow control keywords, even if it is empty

Always add a default block after the last *case* in a *switch* statement

Finish every *if-else*-*if* statement with an *else*-part

Be reluctant with multiple return statements

Don’t use selection statements instead of a simple assignment or initialization

Prefer conditional statements instead of simple if-else constructs

Encapsulate complex expressions in a method or property

Call the most overloaded method from other overloads

Only use optional parameters to replace overloads

Avoid using named parameters

Avoid methods with more than three parameters

Avoid use of ref or out parameters

Avoid methods that take a bool flag

Always check the result of an as operation

Don’t comment-out code

Consider abstracting an external dependency or 3rd party component

## Framework Guidelines

Use C# type aliases instead of the types from the System namespace

Build with the highest warning level

Use Lambda expressions instead of delegates

Only use the dynamic keyword when talking to a dynamic object

# Naming & Layout

|  |  |
| --- | --- |
| **Pascal Casing**  Class, Struct  Interface  Enumeration type  Enumeration values  Event  Protected field  Const field  Read-only static field  Method  Namespace  Property  Type Parameter | AppDomain IBusinessService ErrorLevel  FatalError  OnClick  MainPane  MaximumItems ReadValue  ToString System.Drawing BackColor  TEntity |
| **Camel Casing**  Private field  Variable  Const variable  Parameter | listItem  listOfValues  maximumItems  typeName |

## Naming

Do use proper US English

Don’t include numbers in identifiers

Don’t prefix member fields

Don’t use abbreviations

Name an identifier according its meaning and not its type

Name types using nouns, noun phrases or adjective phrases

Don’t repeat the name of a class or enumeration in its members

Avoid short names or names that can be mistaken with other names

Name methods using verb-object pair

## Documentation

Write comments and documentation in US English

Avoid inline comments

Don’t use /\* \*/ for comments

Only write comments to explain complex algorithms or decisions

Avoid use comments for tracking work to be done later (//TODO markers).

## Layout

Maximum line length is 130 characters.

Indent 4 spaces, don’t use Tabs

Keep one whitespace between keywords like if and the expression, but don’t add whitespaces after ( and before ).

Add a whitespace around operators, like +, -, ==, etc.

Always add braces {} 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.

## Member order

1. Private fields and constants

2. Public constants

3. Public read-only static fields

4. Constructors and the Finalizer

5. Events

6. Properties

7. Other members grouped in a functional manner.

8. Private properties

Other private methods in calling order in-line with public methods.