Project Coding Standards

# Header Comments

The header comments should include:

* Filename
* Class Description
* Revision History

|  |  |  |
| --- | --- | --- |
| Element | Format | Example |
| Class | PascalCase | BankAccount |
| Interface | PascalCase | IEnumerable |
| Static Object | ALL UPPERCASE | DEPOSIT |
| Method Parameter | camelCase | userName |
| Variable | camelCase | firstName |
| Constant | ALL UPPERCASE | TAX\_RATE |
| Namespace | PascalCase | Property.Basement |
| Property | PascalCase | GetUserName |
| Method | PascalCase | AddUser |
| Enumeration | PascalCase | MyConstant |
| Event | PascalCase | LaunchProgram |
| Database Table | CamelCase | order |
| Database Column | CamelCase | orderID |

# Identifier Formatting

## Object Naming

The object names will be prefixed with the abbreviated name of the type of object. The objects themselves will be named with entire words with no abbreviations

Examples:

* TextBox – txtFirstName
* Label – lblFirstName
* Button – btnGenerate

## Variables/Constants

* All variables will be created on separate lines
* All variables will be initialized after they are created, on a separate line
* All variables should be accessed using properties outside of the class, not directly
* All constants will be declared globally (class scope)

Examples:

* string firstName;

firstname=””;

* int number;
* number=0;

## Curly Braces

* All of the code between curly braces should be indented 1 tab space (including nested brackets)
* All of the curly braces should appear on a separate line
* All of the content between curly braces should appear on separate lines

## Spacing

* A blank line should appear before and after a control construct (if, for, while, etc)
* A blank space should appear after each comma in a comma-separated list
* A line of code should be split into several lines if it extends further than the code window
* A space should appear before and after operators such as > < =.

## Methods

* All method parameters should be of a certain type, not generic like object
* Any methods that return a value should only have one return statement
* A single method should fulfill a single task and should not be broad in its purpose
* Any major operations (i.e. DB Access) should not be handled within an event handler, and should be called in a separate method
* Documentation comments should appear above each method indicating its purpose, parameters and return value
* Implementation comments will be provided for explaining complex logic

## Exception Handling

* A try catch block should be looking for a specific exception, not a general exception
* The catch block should not be left empty
* A try-catch block should enclose the entire method code, only the part that will throw an exception

## Others

* *Create other categories if needed and list the criteria as bullet point*