



# Naming Identifiers

Naming Variables, Methods, Classes, Etc.

---

**Telerik Software Academy**  
Learning & Development  
<http://academy.telerik.com>



# Table of Contents

1. General Naming Guidelines
  - The Power of Meaningful Names
2. Naming Classes / Types / Applications
  - Naming Classes, Interfaces, Types, Delegates, Enumerations, Namespaces, Files, Folders, Assemblies, Applications
3. Naming Methods and Method Parameters
4. Naming Variables and Constants
5. Other Naming Guidelines



# General Naming Guidelines

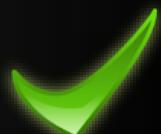
# General Naming Guidelines

- ◆ Always use English
  - How will you feel if you read Vietnamese code with variables named in Vietnamese?
  - English is the only language that all software developers speak
- ◆ Avoid abbreviations
  - Example: **scrpCnt** vs. **scriptsCount**
- ◆ Avoid hard-to-pronounce names
  - Example: **dtbgRegExPtrn** vs. **dateTimeBulgarianRegExPattern**



# Use Meaningful Names

- ◆ Always prefer meaningful names
  - ◆ Names should answer these questions:
    - ◆ What does this class do? What is the intent of this variable? What is this variable / class used for?
    - ◆ Examples:
      - ◆ FactorialCalculator, studentsCount, Math.PI, configFileName, CreateReport
    - ◆ Incorrect examples:
      - ◆ k, k2, k3, junk, f33, KJJ, button1, variable, temp, tmp, temp\_var, something, someValue



# Names Should Be Meaningful in Their Context

- ◆ Whether a name is meaningful or not depends on its context (its enclosing type)
- ◆ Examples of meaningful names:
  - `Generate()` in the class `LabyrinthGenerator`
  - `Find(string fileName)` in the class `FileFinder`
  - `Deposit(decimal amount)` in the class `Account`
- ◆ Examples of meaningless names:
  - `Generate()` in the class `Program`
  - `Find(string name)` in the class `Program`



# Fake Meaningful Names

- ◆ Junior developers often use “fake” meaningful names that are in fact meaningless
  - ◆ Bad naming examples:
    - ◆ Topic6Exercise12, LoopsExercise12, Problem7, OOPLecture\_LastExercise
    - ◆ Yes, Topic6Exercise12 indicates that this is solution to exercise 12, but what is it about?
      - ◆ Sum of numbers or Tetris game?
  - ◆ Better naming:
    - ◆ MaximalNumbersSubsequence



# Naming Classes, Types and Application Components



# Naming Classes and Types

- ◆ Naming types (classes, structures, etc.)
  - ◆ Use PascalCase character casing
    - ◆ In C#, JavaScript, Java, PHP
  - ◆ Examples:
    - ◆ **RecursiveFactorialCalculator, TreeSet, XmlDocument, IEnumerable, Color, TreeNode, InvalidTransactionException, MainForm**
  - ◆ Incorrect examples:
    - ◆ **recursiveFactorialCalculator, recursive\_factorial\_calculator, RECURSIVE\_FACTORIAL\_CALCULATOR**



# Naming Classes and Structures in C#, JavaScript, C++ and Java

- ◆ Use the following formats:

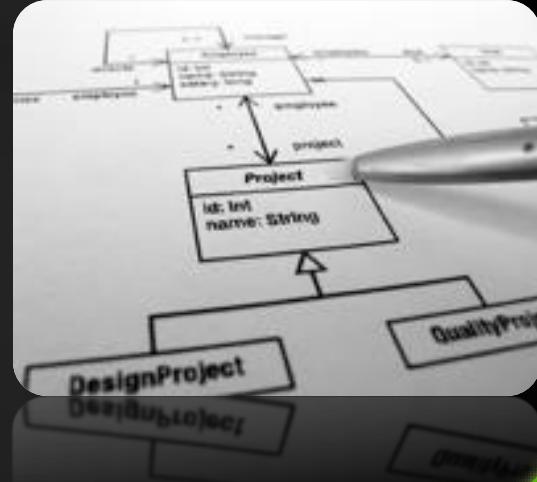
- ◆ [Noun]
  - ◆ [Adjective] + [Noun]

- ◆ Examples:

- ◆ Student, FileSystem, BinaryTreeNode,  
Constants, MathUtils, CheckBox, Calendar

- ◆ Incorrect examples:

- ◆ Move, FindUsers, Fast, ExtremelyFast,  
Optimize, Check, FastFindInDatabase



# Naming Interfaces in C#

- ◆ Following formats are acceptable:

- ◆ 'I' + [Verb] + 'able'
  - ◆ 'I' + [Noun], 'I' + [Adjective] + [Noun]



- ◆ Examples:

- ◆ **IEnumerable**, **IFormattable**, **IDataReader**, **IList**, **IHttpModule**, **ICommandExecutor**

- ◆ Incorrect examples:

- ◆ **List**, **iFindUsers**, **IFast**, **IMemoryOptimize**, **Optimizer**, **FastFindInDatabase**, **CheckBox**



# Naming Interfaces in JS / Java

- ◆ Following formats are acceptable:

- ◆ [Verb] + 'able'
- ◆ [Noun], [Adjective] + [Noun]

- ◆ Examples:

- ◆ **Serializable, Enumerable, Comparable, Runnable, CharSequence, OutputStream**

- ◆ Incorrect examples:

- ◆ **list, FindUsers, Run, Inumber, OPTIMIZER, IMemoryOptimize, FastFindInDatabase**

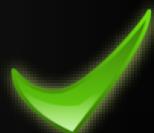
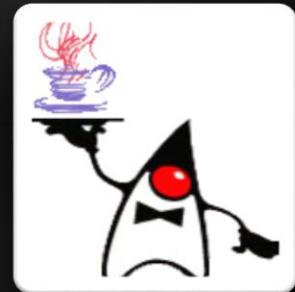


# Naming Enumerations in C#

- ◆ Several formats are acceptable:
  - ◆ [Noun] or [Verb] or [Adjective]
- ◆ Use the same style for all members
- ◆ Examples:
  - ◆ `enum Day {Monday, Tuesday, Wednesday, ...},  
enum AppState {Running, Finished, ...},  
enum WindowState {Normal, Maximized, ...}`
- ◆ Incorrect examples:
  - ◆ `enum Color {red, green, blue, white},  
enum PAGE_FORMAT {A4, A5, A3, LEGAL, ...}`



- ◆ Several formats are acceptable:
  - ◆ [Noun] or [Verb] or [Adjective]
- ◆ Use PascalCase for the enumeration and CAPITALS for its members
- ◆ Examples:
  - ◆ `enum Suit {CLUBS, DIAMONDS, HEARTS, SPADES}, enum Color {RED, GREEN, BLUE, ...}`
- ◆ Incorrect examples:
  - ◆ `enum Color {red, green, blue, white}, enum PAGE_FORMAT {A4, A5, A3, LEGAL, ...}`



# Naming Special Classes

## ◆ Attributes

- ◆ Add 'Attribute' as suffix
- ◆ Example: `WebServiceAttribute`
- ◆ Incorrect example: `WebService`

## ◆ Collection Classes

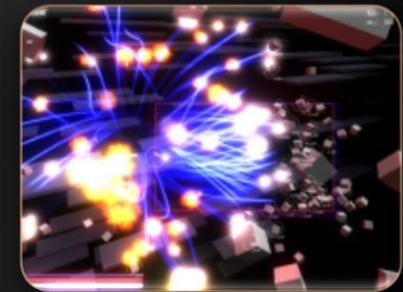
- ◆ Add 'Collection' as suffix
- ◆ Example: `StringsCollection`
- ◆ Incorrect example: `ListOfStrings`



# Naming Special Classes (2)

## ◆ Exceptions

- ◆ Add 'Exception' as suffix
- ◆ Use informative name
- ◆ Example: `FileNotFoundException`
- ◆ Incorrect example: `FileNotFoundException`



## ◆ Delegate Classes

- ◆ Add 'Delegate' or 'EventHandler' as suffix
- ◆ Example: `DownloadFinishedDelegate`
- ◆ Incorrect example: `WakeUpNotification`

# The Length of Class Names

- ◆ How long could be the name of a class / struct / interface / enum / delegate?
  - The name should be as long as required
  - Don't abbreviate the names if this could make them unclear
  - Your IDE has autocomplete, right?
- ◆ Examples: `FileNotFoundException`, `CustomerSupportNotificationService`
- ◆ Incorrect examples: `FNFException`, `CustSuppNotifSrvC`



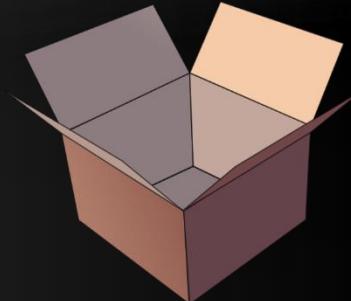
# Naming Namespaces in C#

- ◆ Namespaces naming guidelines
  - ◆ Use PascalCase
- ◆ Following formats are acceptable:
  - ◆ Company . Product . Component . ...
  - ◆ Product . Component . ...
- ◆ Example:
  - ◆ Telerik.WinForms.GridView
- ◆ Incorrect examples:
  - ◆ **Telerik\_WinControlsGridView, Classes**



# Naming Java Packages / JS Namespaces

- ◆ Packages naming guidelines
  - ◆ Use camelCase
- ◆ Following formats are acceptable:
  - ◆ com . company . product . component . ...
  - ◆ product . component . ...
- ◆ Example:
  - ◆ com.apple.quicktime, hibernate.core
- ◆ Incorrect examples:
  - ◆ IBM.DB2.Data, ibm.db2\_data, Tetris.UI



# Naming Project Folders

- ◆ Project folders' names should follow the project namespaces / packages

- ◆ Examples:

- ◆ com
  - ◆ apple
  - ◆ quicktime



- ◆ Telerik.WinForms.GridView

- ◆ Incorrect examples:

- ◆ com\_apple\_quicktime, quicktime.src



# Naming Files in C# / Java

- ◆ Files with source code should have names matching their content
  - ◆ File containing a class **Student** should be names **Student.cs** / **student.java**
- ◆ Example:
  - ◆ **StudentDAO.cs**, **Constants.java**,  
**CryptographyAlgorithms.cs**
- ◆ Incorrect examples:
  - ◆ **Program.cs**, **SourceCode.java**, **\_d2.cs**,  
**WebApplication1.jsp**, **Page1.aspx**



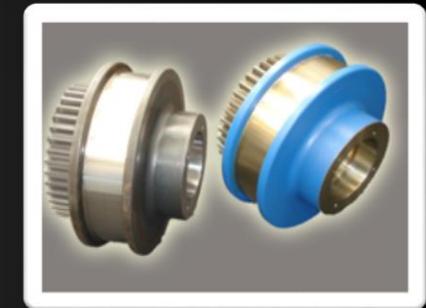
# Naming Files in JavaScript

- ◆ Use small letters and hyphens for JavaScript file names (+ optionally .min + version)
  - ◆ Put a single library / component in a single file
- ◆ Examples:
  - ◆ jquery-1.8.2.min.js, widgets.js, kendo.common.min.js, scriptaculous.js
- ◆ Incorrect examples:
  - ◆ KendoUI.js, jQuery\_classes.js, MyAjax.Library.js, jQuery-1.8.2.js



# Naming .NET Assemblies

- ◆ .NET assembly names should follow the root namespace in its class hierarchy
- ◆ Examples:
  - ◆ Oracle.DataAccess.dll
  - ◆ Interop.CAPICOM.dll
  - ◆ Telerik.WinForms.GridView.dll
- ◆ Incorrect examples:
  - ◆ OracleDataAccess.dll
  - ◆ Telerik\_WinControlsGridView.dll



# Naming JAR Files in Java

- ◆ JAR files names should consist of single word or several words separated by hyphen
  - ◆ Can contain version information
- ◆ Examples:
  - ◆ xalan25.jar
  - ◆ ant-apache-log4j.jar
- ◆ Incorrect examples:
  - ◆ Ant.Apache.Log4J.jar
  - ◆ Oracle.JDBC.Drivers.jar



# Naming Applications

- ◆ Applications should be named meaningfully
  - ◆ Use [Noun] or [Adjective] + [Noun]
  - ◆ Use PascalCase
- ◆ Examples:
  - ◆ BlogEngine 
  - ◆ NewsAggregatorService
- ◆ Incorrect examples:
  - ◆ ConsoleApplication4, WebSite2
  - ◆ zadacha\_14, online\_shop\_temp2

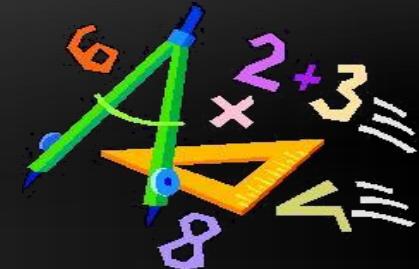




# Naming Methods and Method Parameters

# Naming Methods

- ◆ Methods naming guidelines
  - ◆ Use meaningful method names
  - ◆ Method names should answer the question:
    - ◆ What does this method do?
    - ◆ If you cannot find a good name for a method, think about whether it has a clear intent
- ◆ Examples: `FindStudent`, `LoadReport`, `Sinus`
- ◆ Incorrect examples: `Method1`, `DoSomething`,  
`HandleStuff`, `SampleMethod`, `DirtyHack`



# Naming Methods (2)

- ◆ Use PascalCase for C# and camelCase for JavaScript, PHP and Java
  - Example (C#): LoadSettings
  - Example (JS/PHP/Java): loadSettings
- ◆ Prefer the following formats:
  - [Verb], [Verb] + [Noun],  
[Verb] + [Adjective] + [Noun]
  - Examples: Show, LoadSettingsFile, FindNodeByPattern, ToString, PrintList
  - Incorrect examples: Student, Generator, Counter, White, Approximation, MathUtils



# Methods Returning a Value

- ◆ Methods returning values should describe the returned value
- ◆ Examples:
  - ◆ ConvertMetersToInches, not MetersInches or Convert or ConvertUnit 
  - ◆ Meters2Inches is still acceptable 
  - ◆ CalculateSinus is good but Sinus is still acceptable 
- ◆ Ensure that the unit of measure is obvious
  - ◆ Prefer MeasureFontInPixels to MeasureFont

the work done by the force field along  $C$  is equal to the potential energy of an object at the position  $\mathbf{r}(b)$  minus the potential energy of an object at the position  $\mathbf{r}(a)$ . So we have  $\mathbf{F} = -\nabla P$ . Then by Theorem 12.4, we have

$$\begin{aligned} W &= \int_C \mathbf{F} \cdot d\mathbf{r} = - \int_C \nabla P \cdot d\mathbf{r} \\ &= -[P(\mathbf{r}(b)) - P(\mathbf{r}(a))] \\ &= P(A) - P(B) \end{aligned}$$

# Single Purpose of All Methods

- ◆ Methods should have a single purpose!
  - ◆ Otherwise they cannot be named well
  - ◆ How to name a method that creates annual incomes report, downloads updates from internet and scans the system for viruses? 
  - ◆ `CreateAnnualIncomesReportDownloadUpdatesAndScanForViruses` is a nice name, right?
- ◆ Methods that have multiple purposes (weak cohesion) are hard to be named
  - ◆ Need to be refactored instead of named

# Consistency in Methods Naming

- ◆ Use consistent naming in the entire project
  - LoadFile, LoadImageFromFile, LoadSettings, LoadFont, LoadLibrary, but not ReadTextFile
- ◆ Use consistently the opposites at the same level of abstraction:
  - LoadLibrary vs. UnloadLibrary, but not FreeHandle
  - OpenFile vs. CloseFile, but not DeallocateResource
  - GetName vs. SetName, but not AssignName



# The Length of Method Names

- ◆ How long could be the name of a method?
  - The name should be as long as required
  - Don't abbreviate
  - Your IDE has autocomplete
- ◆ Examples (C#): 

  - LoadCustomerSupportNotificationService,  
CreateMonthlyAndAnnualIncomesReport

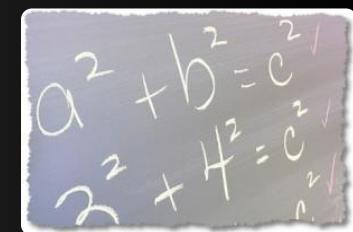
- ◆ Incorrect examples: 

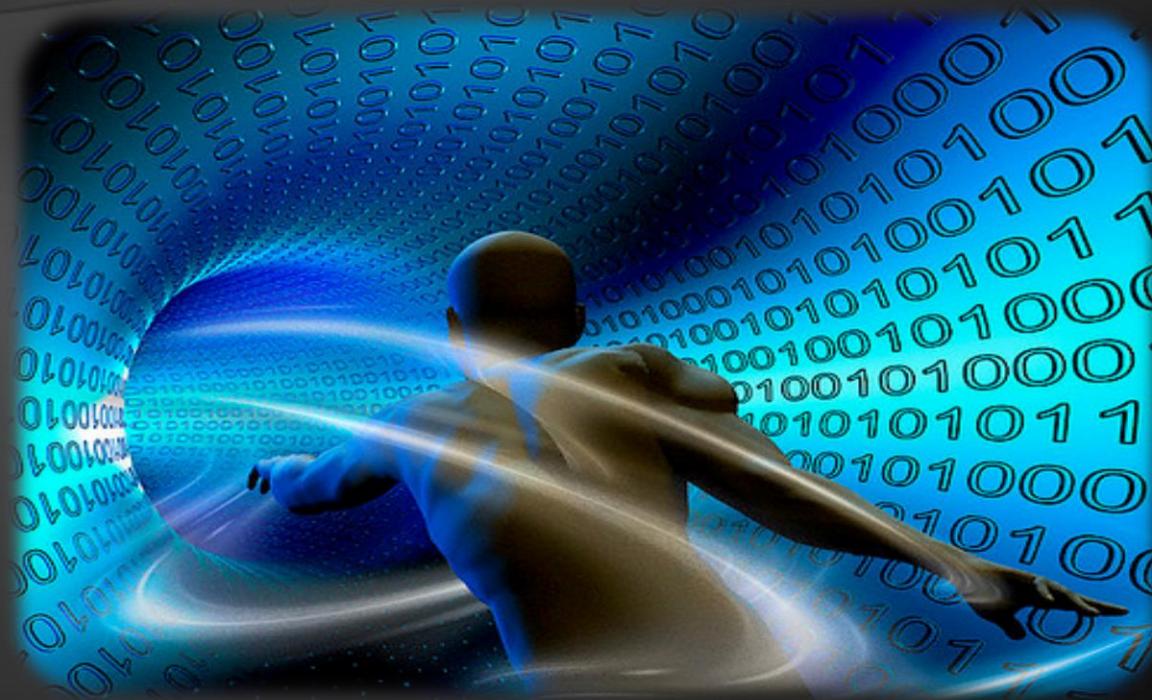
  - LoadCustSuppSrvc, CreateMonthIncReport



# Naming Method Parameters

- ◆ Method parameters names
  - ◆ Preferred form: [Noun] or [Adjective] + [Noun]
  - ◆ Should be in camelCase
  - ◆ Should be meaningful
  - ◆ Unit of measure should be obvious
- ◆ Examples: `firstName`, `report`, `speedKmH`,  
`usersList`, `fontSizeInPixels`, `font` 
- ◆ Incorrect examples: `p`, `p1`, `p2`, `populate`,  
`LastName`, `last_name`, `convertImage` 

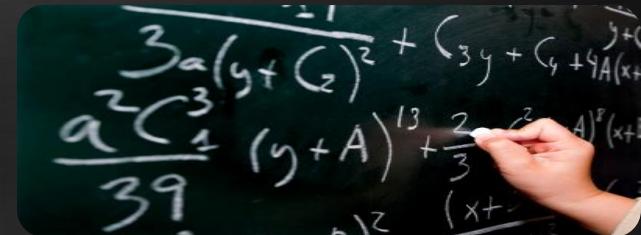




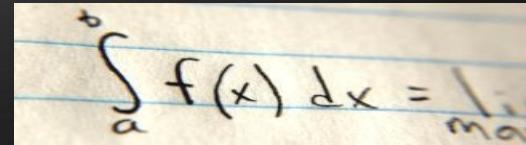
# Naming Variables and Constants

# Naming Variables

- ◆ Variable names
  - Should be in camelCase
  - Preferred form: [Noun] or [Adjective] + [Noun]
  - Should explain the purpose of the variable
    - If you can't find good name for a variable check if it has a single purpose
    - Exception: variables with very small scope, e.g. the index variable in a 3-lines long for-loop
  - Names should be consistent in the project



# Naming Variables – Example



- ◆ Examples:

- ◆ **firstName, report, config, usersList, fontSize, maxSpeed, font, startIndex, endIndex, charsCount, configSettingsXml, dbConnection, createUserSqlCommand**



- ◆ Incorrect examples:

- ◆ **foo, bar, p, p1, p2, populate, LastName, last\_name, LAST\_NAME, convertImage, moveMargin, MAXSpeed, \_firtName, \_\_temp, firstNameMiddleNameAndLastName**



# More about Naming Variables

- ◆ The name should address the problem we solve, not to the means used to solve it
  - ◆ Prefer nouns from the business domain to computer science terms
- ◆ Examples:
  - ◆ **accounts, customers, customerAddress, accountHolder, paymentPlan, vipPlayer**
- ◆ Incorrect examples:
  - ◆ **paymentsPriorityQueue, playersArray, accountsLinkedList, customersHashtable**



# Naming Boolean Variables

- ◆ Give to boolean variables names that imply true or false
- ◆ Use positive boolean variable names
  - ◆ Incorrect example: `if (! notFound) { ... }` 
- ◆ Examples:
  - ◆ `hasPendingPayment`, `customerFound`, `validAddress`, `positiveBalance`, `isPrime`
- ◆ Incorrect examples:
  - ◆ `notFound`, `findCustomerById`, `player`, `programStop`, `run`, `list`, `isUnsuccessfull`  

# Naming Special Variables



- ◆ Naming counters
  - Establish a convention, e.g. [Noun] + 'Count'
  - Examples: `ticketsCount`, `customersCount`
- ◆ State
  - Establish a convention, e.g. [Noun] + 'State'
  - Examples: `blogParseState`, `threadState`
- ◆ Variables with small scope and span
  - E.g. loop counters
  - Short names can be used, e.g. `index`, `i`, `u`

# Temporary Variables

- ◆ Do you really think temporary variables exist?
  - ◆ All variables in the program are temporary because are used temporary only during the program execution, right?
- ◆ Temporary variables can always be named better than **temp** or **tmp**:

```
// Swap a[i] and a[j]
int temp = a[i];
a[i] = a[j];
a[j] = temp;
```



```
// Swap a[i] and a[j]
int oldValue = a[i];
a[i] = a[j];
a[j] = oldValue;
```



# The Length of Variable Names

- ◆ How long could be the name of a variable?
  - Depends on the variable scope and live time
  - More "famous" variables should have longer and more descriptive name
- ◆ Acceptable naming examples:

```
for (int i=0; i<users.Length; i++)  
    if (i % 2 == 0)  
        sum += users[i].Weight;
```



```
class Student {  
    public string lastName;  
}
```



- ◆ Unacceptable naming examples:

```
class LinkedList {  
    public int flag { get; set; }  
}
```



```
class Student {  
    private int i;  
}
```



# Naming Constants in C#

- ◆ Use CAPITAL LETTERS for const fields and PascalCase for readonly fields
- ◆ Use meaningful names that describe their value
- ◆ Examples:

```
private const int READ_BUFFER_SIZE = 8192;  
public static readonly PageSize DefaultPageSize = PageSize.A4;  
private const int FONT_SIZE_IN_POINTS = 16;
```

- ◆ Incorrect examples:

```
public const int MAX = 512; // Max what? Apples or Oranges?  
public const int BUF256 = 256; // What about BUF256 = 1024?  
public const string GREATER = ">"; // GREATER_HTML_ENTITY  
public const int FONT_SIZE = 16; // 16pt or 16px?  
public const PageSize PAGE = PageSize.A4; // Maybe PAGE_SIZE?
```



# Naming Constants in JavaScript, Java, PHP and C++

- ◆ Use CAPITAL LETTERS for JavaScript / Java / PHP / C++ constants
- ◆ Use meaningful names
  - ◆ Constants should describe their value
- ◆ Examples:

```
public static final int READ_BUFFER_SIZE = 8192;  
public static final PageSize DEFAULT_PAGE_SIZE = PageSize.A4;  
public static final int FONT_SIZE_IN_POINTS = 16;
```

- ◆ Incorrect examples:

```
public static final int NAME = "BMW"; // What name? Car name?  
public static final int BufSize = 256; // Use CAPITALS  
public static final int font_size_pixels = 16; // CAPITALS
```

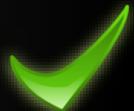




# Other Naming Guidelines

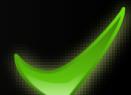
# Names to Avoid

- ◆ Don't use numbers in the identifiers names
  - ◆ Example:
    - ◆ PrintReport and PrintReport2
    - ◆ What is the difference?
  - ◆ Exceptions:
    - ◆ When the number is part of the name itself,  
e.g. RS232Port, COM3, Win32APIFunctions
- ◆ Don't use Cyrillic or letters from other alphabet
  - ◆ FindСтудентByName, DisplayΩ2Protein



# Never Give Misleading Name!

- ◆ Giving a misleading name is even worse than giving a totally unclear name
- ◆ Example:
  - Consider a method that calculates the sum of all elements in an array
  - Its should be named **Sum** or **CalculateSum**
  - What about naming it **CalculateAverage** or **Max** or **CheckForNegativeNumber**?
  - It's crazy, but be careful with "copy-paste"



# What's Wrong with This Code?

```
FileStream fs = new FileStream(FILE_NAME, FileMode.CreateNew);
// Create the writer for data.
BinaryWriter w = new BinaryWriter(fs);
// Write data to Test.data.
for (int i = 0; i < 11; i++)
{
    w.Write( (int) i);
}
w.Close();
fs.Close();
// Create the reader for data.
fs = new FileStream(FILE_NAME, FileMode.Open, FileAccess.Read);
BinaryReader r = new BinaryReader(fs);
// Read data from Test.data.
for (int i = 0; i < 11; i++)
{
    Console.WriteLine(r.ReadInt32());
}
r.Close();
fs.Close();
```



Source: <http://msdn.microsoft.com/en-us/library/36b93480.aspx>

# Naming Identifiers

Questions?

1. Refactor the following examples to produce code with well-named C# identifiers:

```
class class_123
{
    const int max_count=6;
    class InClass_class_123
    {
        void Метод_на_Class_InClass_class_123(bool promenliva)
        {
            string promenlivakatoString=promenliva.ToString();
            Console.WriteLine(promenlivakatoString);
        }
    }
    public static void Метод_За_Вход()
    {
        class_123.InClass_class_123 инстанция =
            new class_123.InClass_class_123();
        инстанция.Метод_на_Class_InClass_class_123(true);
    }
}
```

C#

2. Refactor the following examples to produce code with well-named identifiers in C#:

```
class Hauptklasse
{
    enum Пол { ултра_Батка, Яка_Мацка };

    class чуек
    {
        public Пол пол { get; set; }
        public string име_на_Чуека { get; set; }
        public int Възраст { get; set; }
    }
}
```

C#

*// continues at the next slide ...*

# Homework (3)

```
public void Make_Чуек(int магический_НомерНаЕДИНЧОВЕК)
{
    чуек new_Чуек = new чуек();
    new_Чуек.Възраст = магический_НомерНаЕДИНЧОВЕК;
    if (магический_НомерНаЕДИНЧОВЕК%2 == 0)
    {
        new_Чуек.име_на_Чуека = "Батката";
        new_Чуек.пол = Пол.ултра_Батка;
    }
    else
    {
        new_Чуек.име_на_Чуека = "Мацето";
        new_Чуек.пол = Пол.Яка_Мацка;
    }
}
```

C#

3. Refactor the following examples to produce code with well-named identifiers in JavaScript

```
function _ClickON_TheButton( THE_event, argumenti) {  
    var moqProzorec= window;  
    var brauzyra = moqProzorec.navigator.appCodeName;  
    var ism=brauzyra=="Mozilla";  
    if(ism)  
    {  
        alert("Yes");  
    }  
    else  
    {  
        alert("No");  
    }  
}
```

JavaScript

# Homework (5)

4. Refactor and improve the naming in the C# source project "3. Naming-Identifiers-Homework.zip". You are allowed to make other improvements in the code as well (not only naming) as well as to fix bugs.

# Free Trainings @ Telerik Academy

- ◆ C# Programming @ Telerik Academy

- ◆ [csharpfundamentals.telerik.com](http://csharpfundamentals.telerik.com)



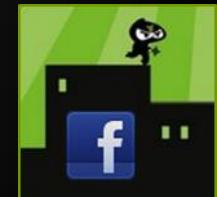
- ◆ Telerik Software Academy

- ◆ [academy.telerik.com](http://academy.telerik.com)



- ◆ Telerik Academy @ Facebook

- ◆ [facebook.com/TelerikAcademy](https://facebook.com/TelerikAcademy)



- ◆ Telerik Software Academy Forums

- ◆ [forums.academy.telerik.com](http://forums.academy.telerik.com)

