اصول و قراردادهای نامگذاری در دات‌نت

نامگذاری (**Naming**) اشیا یک برنامه شاید در نگاه اول دارای اهمیت بالایی نباشه، اما تجربه نشون داده که در پروژه‌های بزرگ که با کمک چندین مجموعه به انجام میرسه نامگذاری صحیح و اصولی که از یکسری قواعد کلی و مناسب پیروی میکنه میتونه به پیشبرد اهداف و مدیریت راحتتر برنامه کمک بسیاری بکنه.
بیشتر موارد اشاره شده در این مطلب از کتاب جامع و مفید Framework Design Guidelines اقتباس شده که خوندن این کتاب مفید رو به خوانندگان توصیه میکنم.
برای کمک به نوشتن اصولی و راحتتر سورسهای برنامه‌ها در ویژوال استودیو نرم افزارهای متعددی وجود داره که با توجه به تجربه شخصی خودم نرم افزار Resharper محصول شرکت Jetbrains یکی از بهترین هاست که در مورد خاص مورد بحث در این مطلب نیز بسیار خوب عمل میکنه.
برخی از موارد موجود در مطلب جاری نیز از قراردادهای پیشفرض موجود در نرم افزار Resharper نسخه 6.0 برگرفته شده است و قسمتی نیز از تجربه شخصی خودم و سایر دوستان و همکاران بوده است.
اصل این مطلب حدود یکسال پیش تهیه شده و اگر نقایصی وجود داره لطفا اشاره کنین.

اصول و قراردادهای نامگذاری در دات‌نت

انواع نامگذاری

نامگذاری اشیا در حالت کلی را می‌توان به سه روش زیر انجام داد:

Pascal Casing : .1در این روش حرف اول هر کلمه در نام شی به صورت بزرگ نوشته می‌شود.

```
1    FirstName
```

camel Casing: .2حرف اول در اولین کلمه نام هر شی به صورت کوچک و حرف اول بقیه کلمات به صورت بزرگ نوشته می‌شود.

```
1    firstName
```

Hungarian: .3در این روش برای هر نوع شی موجود یک پیشوند درنظر گرفته می‌شود تا از روی نام شی بتوان به نوع آن پی برد. در ادامه و پس از این پیشوندها سایر کلمات بر اساس روش Pascal Casing نوشته می‌شوند.

```
1    strFirstName
2    lblFirstName
```

نکته: استفاده از این روش به جز در نامگذاری کنترل‌های UI منسوخ شده است.

قراردادهای کلی

.1نباید نام اشیا تنها در بزرگ یا کوچک بودن حروف با هم فرق داشته باشند. به عنوان مثال نباید دو کلاس با نامهای MyClassو myClass داشته باشیم. هرچند برخی از زبانها case-sensitive هستند اما برخی دیگر نیز چنین قابلیتی ندارند (مثل .VB.NET) بنابراین اگر بخواهیم کلاس‌های تولیدی ما در تمام محیطها و زبان‌های برنامه نویسی قابل اجرا باشند باید از این قرارداد پیروی کنیم.
.2تا آنجا که امکان دارد باید از به‌کار بردن مخفف کلمات در نامگذاری اشیا دوری کنیم. مثلا به جای استفاده از GetChr باید از GetCharacterاستفاده کرد.
البته در برخی موارد که مخفف واژه موردنظر کاربرد گسترده ای دارد می‌توان از عبارت مخفف نیز استفاده کرد. مثل UI به جای UserInterface و یا IO به جای.InputOutput

آ. اصول نامگذاری فضای نام(namespace)

.1اساس نامگذاری فضای نام باید از قاعده زیر پیروی کند:

```
1    <Company>.<Technology|Produt|Project>[.<Feature>][.<SubNamespace>]
```

(< > : اجباری [ ] : اختیاری)

.2برای نامگذاری فضای نام باید از روش Pascal Casing استفاده شود.
.3در هنگام تعریف فضاهای نام به وابستگی آنها توجه داشته باشید. به عنوان مثال اشیای درون یک فضای نام پدر نباید به اشیای درون فضای نام یکی از فرزندانش وابسته باشد. مثلا در فضای نام System نباید اشیایی وجود داشته باشند که به اشیای درون فضای نام System.UI وابسته باشند.
.4سعی کنید از نامها به صورت جمع برای عناوین فضای نام استفاده کنید. مثلا به جای استفاده از Kara.CSS.HQ.Manager.Entityاز Kara.CSS.HQ.Manager.Entities استفاده کنید. البته برای مواردی که از عناوین مخفف و یا برندهای خاص استفاده کرده‌اید از این قرارداد پیروی نکنید. مثلا نباید از عنوانی شبیه به Kara.CSS.Manager.IOs استفاده کنید.
.5از عنوانی پایدار و مستقل از نسخه محصول برای بخش دوم عنوان فضای نام استفاده کنید. بدین معنی که این عناوین با گذر زمان و تغییر و تحولات در محتوای محصول و یا تولیدکننده نباید تغییر کنند. مثال:

```
Microsoft.Reporting.WebForms
Kara.Support.Manager.Enums
Kara.CSS.HQ.WebUI.Configuration
```

.6از عناوین یکسان برای فضای نام و اشیای درون آن استفاده نکنید. مثلا نباید کلاسی با عنوان Manager در فضای نام Kara.CSS.Managerوجود داشته باشد.

.7از عناوین یکسان برای اشیای درون فضاهای نام یک برنامه استفاده نکنید. مثلا نباید دو کلاس با نام Package در فضاهای نام Kara.CSS.Manger و Kara.CSS.Manger.Entities داشته باشید.

ب. اصول نامگذاری کلاس‌ها وStruct ها

.1عنوان کلاس باید اسم یا موصوف باشد.

.2در نامگذاری کلاس‌ها باید از روش Pascal Casing استفاده شود.

.3نباید از عناوین مخففی که رایج نیستند استفاده کرد.

.4از پیشوندهای زائد مثل C یا Cls نباید استفاده شود.

.5نباید از کاراکترهایی به غیر از حروف (و یا در برخی موارد خیلی خاص، شماره نسخه) در نامگذاری کلاس‌ها استفاده شود.

مثال:

درست:

```
PackageManager , PacakgeConfigGenerator
Circle , Utility , Package
```

نادرست:

```
CreateConfig , classdata
CManager , ClsPackage , Config_Creator , Config1389
```

.6در نامگذاری اشیای Generic از استفاده از عناوینی چون Element, Node, Log و یا Message پرهیز کنید. این کار موجب به‌وجودآمدن کانفلیکت در عناوین اشیا می‌شود. در این موارد بهتر است از عناوینی چون FormElement, XmlNode, EventLogو SoapMessage استفاده شود. درواقع بهتر است نام اشیای جنریک، برای موارد موردنیاز، کاملا اختصاصی و درعین حال مشخص‌کننده محتوا و کاربرد باشند.

.7از عناوینی مشابه عناوین کتابخانه‌های پایه دات‌نت برای اشیای خود استفاده نکنید. مثلا نباید کلاسی با عنوان Console, Parameter, Actionو یا Data را توسعه دهید. هم‌چنین از کابرد عناوینی مشابه اشیای موجود در فضاهای نام غیرپایه دات‌نت و یا غیردات‌نتی اما مورداستفاده در پروژه خود که محتوا و کاربردی مختص همان فضای نام دارند پرهیز کنید. مثلا نباید کلاسی با عنوان Page در صورت استفاده از فضای نام System.Web.UI تولید کنید.

.8سعی کنید در مواردی که مناسب به‌نظر می‌رسد از عنوان کلاس پایه در انتهای نام کلاس‌های مشتق‌شده استفاده کنید. مثل FileStream که از کلاس Stream مشتق شده است. البته این قاعده در تمام موارد نتیجه مطلوب ندارد. مثلا کلاس Button که از کلاس Control مشتق شده است. بنابراین در به‌کاربردن این مورد بهتر است تمام جوانب و قواعد را درنظر بگیرید.

پ. اصول نامگذاری مجموعه‌ها(Collections)

یک مجموعه در واقع یک نوع کلاس خاص است که حاوی مجموعه‌ای از داده‌هاست و از همان قوانین کلاس‌ها برای نامگذاری آن‌ها استفاده می‌شود.

.1بهتر است در انتهای عنوان مجموعه از کلمه Collection استفاده شود. مثال:

```
CenterCollection , PackageCollection
```

.2البته درصورتی‌که کلاس مورد نظر ما رابط IDictionary را پیاده‌سازی کرده باشد بهتر است از پسوند Dictionary استفاده شود.

ت. اصول نامگذاریDelegate ها

.1عنوان یک delegate باید اسم یا موصوف باشد.

.2در نامگذاریdelegate ها باید از روش Pascal Casing استفاده شود.

.3نباید از عناوین مخففی که رایج نیستند استفاده کرد.

.4از پیشوندهای زائد مثل D یا del نباید استفاده شود.

.5نباید از کاراکترهایی به غیر از حروف در نامگذاریdelegate ها استفاده شود.

.6**نباید** در انتهای نام یک delegate از عبارت Delegate استفاده شود.

.7بهتر است که درصورت امکان در انتهای نام یک delegate که برای Event Handler استفاده نمی‌شود از عبارت Callback استفاده شود. البته تنها درصورتی‌که معنی و مفهوم مناسب را داشته باشد.
مثال:
نحوه تعریف یکdelegate

```
1  public delegate void Logger (string log);
2  public delegate void LoggingCallback (object sender, string reason);
```

ث. اصول نامگذاری رویدادها(Events)

.1عنوان یک رویداد باید فعل یا مصدر باشد.
.2در نامگذاری کلاس‌ها باید از روش Pascal Casing استفاده شود.
.3نباید از عناوین مخففی که رایج نیستند استفاده کرد.
.4از پیشوندهای زائد نباید استفاده شود.
.5نباید از کاراکترهایی به غیر از حروف در نامگذاری رویدادها استفاده شود.
.6بهتر است از پسوند EventHandler در عنوان هندلر رویداد استفاده شود.
.7از به کاربردن عباراتی چون AfterXXX و یا BeforeXXX برای نمایش رویدادهای قبل و یا بعد از رخداد خاصی خودداری شود. به جای آن باید از اسم مصدر (شکلing دار فعل) برای نامگذاری رویداد قبل از رخداد و همچنین شکل گذشته فعل برای نامگذاری رویداد بعد از رخداد خاص استفاده کرد.
مثلا اگر کلاسی دارای رویداد Open باشد باید از عنوان Openning برای رویداد قبل از Open و از عنوان Opened برای رویداد بعد از Open استفاده کرد.
.8نباید از پیشوند On برای نامگذاری رویداد استفاده شود.
.9همیشه پارامتر e و sender را برای آرگومانهای رویداد پیاده سازی کنید .sender که از نوع object است نمایش دهنده شیی است که رویداد مربوطه را به وجود آورده است و e درواقع آرگومانهای رویداد مربوطه است.
.10عنوان کلاس‌آرگومان‌های رویداد باید دارای پسوند EventArgs باشد.
مثال:
عنوان کلاس‌آرگومان:

```
1  AddEventArgs , EditEventArgs , DeleteEventArgs
```

عنوان رویداد:

```
1  Adding , Add , Added
```

تعریف یک:EventHandler

```
1  public delegate void <EventName>EventHandler<span> </span> (object sender, <EventName>EventArgs e);
```

نکته: نیاز به تولید یک EventHandler مختص توسعه یک برنامه به‌ندرت ایجاد می‌شود. در اکثر موارد می‌توان با استفاده از کلاس جنریک EventHandler<TEventArgs>تمام احتیاجات خود را برطرف کرد.
تعریف یک رویداد:

```
1  public event EventHandler<span> </span><AddEventArgs> Adding;
```

ج. اصول نامگذاریAttribute ها

.1عنوان یک attribute باید اسم یا موصوف باشد.
.2در نامگذاریattribute ها باید از روش Pascal Casing استفاده شود.
.3نباید از عناوین مخففی که رایج نیستند استفاده کرد.
.4از پیشوندهای زائد مثل A یا atr نباید استفاده شود.
.5نباید از کاراکترهایی به غیر از حروف در نامگذاریattribute ها استفاده شود.
.6بهتر است در انتهای نام یک attribute از عبارت Attribute استفاده شود.
مثال:

```
1  DisplayNameAttribute , MessageTypeAttribute
```

چ. اصول نامگذاریInterface ها

.1در ابتدای عنوان interface باید از حرف I استفاده شود.
.2نام باید اسم، موصوف یا صفتی باشد که interface را توصیف می‌کند.
به عنوان مثال:

```
1  IComponent  (اسم)
   IConnectionProvider (موصوف)
```

```
ICloneable (صفت)
```

.3از روش Pascal Casing استفاده شود.
.4خودداری از بکاربردن عبارات مخفف غیررایج.
.5باید تنها از کاراکترهای حرفی در نام interface استفاده شود.

ح. اصول نام‌گذاریEnumeration ها

.1استفاده از روشPascal Casing
.2خودداری از کاربرد عبارات مخفف غیررایج
.3تنها از کاراکترهای حرفی در نامEnumretion ها استفاده شود.
.4نباید از پسوند یا پیشوند Enum یا Flag استفاده شود.
.5اعضای یک Enum نیز باید با روش Pascal Casing نام‌گذاری شوند.
مثال:

```
public enum FileMode {
    Append,
    Read, …
}
```

.6درصورتی‌که enum موردنظر از نوع flag نیست باید عنوان آن مفرد باشد .
.7درصورتی‌که enum موردنظر برای کاربرد flag طراحی شده باشد نام آن باید جمع باشد. مثال:

```
[Flag]
public enum KeyModifiers {
    Alt = 1,
    Control = 2,
    Shift = 4
}
```

.8از به‌کاربردن پسوند و یا پیشوندهای اضافه در نام‌گذاری اعضای یک enum نیز پرهیز کنید. مثلا نام‌گذاری زیر نادرست است:

```
public enum OperationState {
    DoneState,
    FaultState,
    RollbackState
}
```

خ. اصول نام‌گذاری متدها

.1نام متد باید فعل یا ترکیبی از فعل و اسم یا موصوف باشد.
.2باید از روش Pascal Casing استفاده شود.
.3خودداری از بکاربردن عبارات مخفف غیررایج و یا استفاده زیاد از اختصار
.4تنها از کاراکترهای حرفی برای نام متد استفاده شود.
مثال:

```
AddDays , Save , DeleteRow , BindData , Close , Open
```

د. اصول نام‌گذاریProperty ها

.1نام باید اسم، صفت یا موصوف باشد.
.2باید از روش Pascal Casing استفاده شود.
.3خودداری از بکاربردن عبارات مخفف غیررایج.
.4تنها از کاراکترهای حرفی برای نام‌گذاری پراپرتی استفاده شود.
مثال:

```
Radius , ReportType , DataSource , Mode , CurrentCenterId
```

.5از عبارت Get در ابتدای هیچProperty ای استفاده نکنید.
.6نام خاصیت‌هایی که یک مجموعه برمی‌گرداند باید به صورت جمع باشد. عنوان اینProperty ها نباید به‌صورت مفرد به همراه پسوند Collection یا List باشد. مثال:

```
public CenterCollection Centers { get; set; }
```

.7خواص Boolean را با عناوینی مثبت پیاده‌سازی کنید. مثلا به‌جای استفاده از CantRead از CanRead استفاده کنید. بهتر است این Property ها پیشوندهایی چون Is, Can یا Has داشته باشند، البته تنها درصورتی‌که استفاده از چنین پیشوندهایی ارزش افزوده داشته و مفهوم آن را بهتر برساند. مثلا عنوان CanSeek مفهوم روشن‌تری نسبت به Seekable دارد. اما استفاده از Created خیلی بهتر از IsCreated است یا Enabled کاربرد به مراتب راحت‌تری از IsEnabled دارد.
برای تشخیص بهتر این موارد بهتر است از روشif سنجی استفاده شود. به عنوان مثال

```
if (list.Contains(item))
if (regularExpression.Matches(text))
if (stream.CanSeek)
if (context.Created)
if (form.Enabled)
```

مفهوم درست‌تری نسبت به موارد زیر دارند:

```
if (list.IsContains(item))
if (regularExpression.Match(text))
if (stream.Seekable)
if (context.IsCreated)
if (form.IsEnabled)
```

.8بهتر است در موارد مناسب عنوان Property با نام نوعش برابر باشد. مثلا

```
public Color Color { get; set; }
```

ذ. اصول نامگذاری پارامترها

پارامتر درحالت کلی به آرگومان وروی تعریف شده برای یک متد گفته می‌شود.
.1حتما از یک نام توصیفی استفاده شود. از نامگذاری پارامترها براساس نوعشان به‌شدت پرهیز کنید.
.2از روش camel Casing استفاده شود.
.3تنها از کاراکترهای حرفی برای نامگذاری پارامترها استفاده شود.
مثال:

```
firstName , e , id , packageId , centerName , name
```

.4نکاتی برای نامگذاری پارامترهای:**Operator Oveloading**
-برایoperator های دو پارامتری (binary operators) از عناوین left و right برای پارامترهای آن استفاده کنید:

```
public static MyType operator +(MyType left, MyType right)
public static bool operator ==(MyType left, MyType right)
```

-برایoperator های تک‌پارامتری (unary operators) اگر برای پارامتر مورد استفاده هیچ عنوان توصیفی مناسبی پیدا نکردید حتما از عبارت value استفاده کنید:

```
public static MyType operator ++(MyType value)
```

-درصورتی‌که استفاده از عناوین توصیفی دارای ارزش افزوده بوده و خوانایی کد را بهتر می‌کند حتما از این نوع عناوین استفاده کنید:

```
public static MyType operator /(MyType dividend, MyType divisor)
```

-نباید از عبارات مخفف یا عناوینی با اندیس‌های عددی استفاده کنید:

```
public static MyType operator -(MyType d1, MyType d2) // incorrect!
```

ر. اصول نامگذاری متغیر(Variable) ها

-نام متغیر باید اسم، صفت یا موصوف باشد.
نامگذاری متغیرها باید با توجه به نوع آن انجام شود.

.1متغیرهای عمومی (public) و protected وConstant ها

-استفاده از روشPascal Casing
-تنها از کاراکترهای حرفی برای نام متغیر عمومی استفاده شود.
مثال:

```
Area , DataBinder , PublicCacheName
```

.2متغیرهای private در سطح کلاس یا همان(field)

-نام این نوع متغیر باید با یک "_" شروع شود.
-از روش camel Casing استفاده شود.

مثال:

```
_centersList
_firstName
_currentCenter
```

.3متغیرهای محلی در سطح متد

-باید از روش camel Casing استفاده شود.

-تنها از کاراکترهای حرفی استفاده شود.

مثال:

```
parameterType , packageOperationTypeId
```

ز. اصول نامگذاری کنترل‌هایUI

.1نام باید اسم یا موصوف باشد.

.2استفاده از روش! Hungarian

.3عبارت مخفف معرفی‌کننده کنترل، باید به اندازه کافی برای تشخیص نوع آن مناسب باشد.

مثال:

```
lblName (Label)
txtHeader (TextBox)
btnSave (Button)
```

ژ. اصول نامگذاریException ها

تمام موارد مربوط به نامگذاری کلاس‌ها باید در این مورد رعایت شود .

.1باید از پسوند Exception در انتهای عنوان استفاده شود.

مثال:

```
ArgumentNullException , InvalidOperaionException
```

س. نامگذاری اسمبلی‌ها وDLL ها

.1عناوینی که برای نامگذاری اسمبلی‌ها استفاده می‌شوند، باید نمایش‌دهنده محتوای کلی آن باشند. مثل:

```
System.Data
```

.2از روش زیر برای نامگذاری اسمبلی‌ها استفاده شود:

```
<Company>.<Component>.dll
<Company>.<Project|Product|Technology>.<Component>.dll
```

مثل:

```
Microsoft.CSharp.dll , Kara.CSS.Manager.dll
```

ش. نامگذاری پارامترهای نوعGeneric (type parameter)

.1از حرف T برای پارامترهای تک‌حرفی استفاده کنید. مثل:

```
public int IComparer<T> {…}
public delegate bool Predicate<T> (T item)
```

.2تمامی پارامترهای جنریک را با عناوینی توصیفی و مناسب که مفهوم و کاربرد آنرا برساند نامگذاری کنید، مگر آنکه یافتن چنین عباراتی ارزش افزوده‌ای در روشن‌تر کردن کد نداشته باشد. مثال:

```
public int ISessionChannel<TSession> {…}
public delegate TOutput Converter<TInput, TOutput> (TInput from)
public class Nullable<T> {…}
public class List<T> {…}
```

.3در ابتدای عناوین توصیفی حتما از حرف T استفاده کنید.

.4بهتر است تا به‌صورتی روشن نوع قید قرار داده شده بر روی پارامتری خاص را در نام آن پارامتر نمایش دهید. مثلا اگر قید ISessionرا برای پارامتری قرار دادید بهتر است نام آن پارامتر را TSession درنظر بگیرید.

ص. نامگذاری کلیدResource ها

به دلیل شباهت ساختاری که میان کلیدهای resource وproperty ها وجود دارد قواعد نامگذاریproperty ها در اینجا نیز معتبر هستند.

.1از روش نام‌گذاری Pascal Casing برای کلیدهای resource استفاده کنید.
.2از عناوین توصیفی برای این کلیدها استفاده کنید. سعی کنید تا حد امکان به هیچ وجه از عناوین کوتاه و یا مخففی که مفهوم را به‌صورت ناکامل می‌رساند استفاده نکنید. درواقع سعی کنید که خوانایی بیشتر کد را فدای فضای بیشتر نکنید.
.3از کلیدواژه‌های CLR و یا زبان مورداستفاده برای برنامه‌نویسی در نام‌گذاری این کلیدها استفاده نکنید.
.4تنها از حروف و اعداد و _ در نام‌گذاری این کلیدها استفاده کنید.
.5سعی کنید از عناوین توصیفی همانند زیر برای پیام‌های مناسب خطاها جهت نمایش به کاربر برای کلیدهای مربوطه استفاده کنید. درواقع نام کلید باید ترکیبی از نام نوع خطا و یک آی‌دی مشخص‌کننده پیغام مربوطه باشد:

```
1  ArgumentExceptionIllegalCharacters
2  ArgumentExceptionInvalidName
3  ArgumentExceptionFileNotFound
```

نکاتی درمورد کلمات مرکب

کلمات مرکب به کلماتی گفته می‌شود که در آن از بیش از یک کلمه با مفهوم مستقل استفاده شده باشد. مثل Callback یا FileName.

باید توجه داشت که با تمام کلمات موجود در یک کلمه مرکب نباید همانند یک کلمه مستقل رفتار کرد و حرف اول آن را در روش‌های نام‌گذاری موجود به‌صورت بزرگ نوشت. کلمات مرکبی وجود دارند که به آن‌ها closed-form گفته می‌شود. این کلمات مرکب با اینکه از 2 یا چند کلمه دارای مفهوم مستقل تشکیل شده‌اند اما به‌خودی‌خود دارای مفهوم جداگانه و مستقلی هستند. برای تشخیص این کلمات می‌توان به فرهنگ لغت مراجعه کرد (و یا به‌سادگی از نرم‌افزار Microsoft Word استفاده کرد) و دریافت که آیا کلمه مرکب موردنظر آیا مفهوم مستقلی برای خود دارد، یعنی درواقع آیا عبارتی closed-form است. با کلمات مرکب از نوع closed-form همانند یک کلمه ساده برخورد می‌شود !

در جدول زیر مثال‌هایی از عبارات رایج و نحوه درست و نادرست استفاده از هریک نشان داده شده است.

Pascal Casing	camel Casing	Wrong
Callback	callback	CallBack
BitFlag	bitFlag	Bitflag / bitflag
Canceled	canceled	Cancelled
DoNot	doNot	Donot / Don’t
Email	email	EMail
Endpoint	endpoint	EndPoint / endPoint
FileName	fileName	Filename / filename
Gridline	gridline	GridLine / gridLine
Hashtable	hashtable	HashTable / hashTable
Id	id	ID
Indexes	indexes	Indices

Logoff / LogOut !	logOff	LogOff
Logon / LogIn !	logOn	LogOn
Signout / SignOff	signOut	SignOut
Signin / SignOn	signIn	SignIn
MetaData / metaData	metadata	Metadata
MultiPanel / multiPanel	multipanel	Multipanel
MultiView / multiView	multiview	Multiview
NameSpace / nameSpace	namespace	Namespace
OK	ok	Ok
PI	pi	Pi
PlaceHolder / placeHolder	placeholder	Placeholder
Username / username	username	UserName
Whitespace / whitespace	whiteSpace	WhiteSpace
Writeable / writeable	writable	Writable

همان‌طور که در جدول بالا مشاهد می‌شود در استفاده از قوانین عبارات مخفف دو مورد استثنا وجود دارد که عبارتند از Id و Ok.این کلمات باید همان‌طور که در اینجا نشان داده شده‌اند استفاده شوند.

نکاتی درباره عبارات مخفف

1.عبارات مخفف (Acronym) با خلاصه‌سازی کلمات (Abbreviation) فرق دارند. یک عبارت مخفف شامل حروف اول یک عبارت طولانی یا معروف است، درصورتی‌که خلاصه‌سازی یک عبارت یا کلمه از حذف بخشی از آن به‌دست می‌آید. تا آنجاکه امکان دارد از خلاصه‌سازی عبارات نباید استفاده شود. همچنین استفاده از عبارات مخفف غیررایج توصیه نمی‌شود.
مثال IO :وUI

2.براساس تعریف، یک مخفف حداقل باید 2 حرف داشته باشد. نحوه برخورد با مخفف‌های دارای بیشتر از 2 حرف با مخفف‌های دارای 2 حرف باهم متفاوت است. با مخفف‌های دارای 2 حرف همانند کلمه‌ای یک حرفی! برخورد می‌شود، درصورتی‌که با سایر مخفف‌ها همانند یک کلمه کامل چندحرفی برخورد می‌شود. به‌عنوان مثال IOStream برای روش PascalCasingو ioStream برای روش camelCasing استفاده می‌شود. همچنین از HtmlBody و htmlBody به‌ترتیب برای روش‌های Pascal و camel استفاده می‌شود.

3.هیچ‌کدام از حروف یک عبارت مخفف در ابتدای یک واژه نام‌گذاری‌شده به روش camelCasing به‌صورت بزرگ نوشته نمی‌شود!

نکته :موارد زیادی را می‌توان یافت که در ابتدا به‌نظر می‌رسد برای پیاده‌سازی آنها باید قوانین فوق را نقض کرد. این موارد شامل استفاده از کتابخانه‌های سایر پلتفرم‌ها (مثل MFC, HTML و غیره)، جلوگیری از مشکلات جغرافیایی! (مثلا در مورد

نام کشورها یا سایر موقعیت‌های جغرافیایی)، احترام به نام افراد درگذشته، و مواردی از این دست می‌شود. اما در بیشتر قریب به اتفاق این موارد هم می‌توان بدون نقض قوانین فوق اقدام به نامگذاری اشیا کرد، بدون اینکه با تغییر عبارت اصلی (که موجب تطابق با این قوانین می‌شود) لطمه‌ای به مفهوم آن بزنند. البته تنها موردی که به‌نظر می‌رسد می‌تواند قوانین فوق را نقض کند نام‌های تجاری هستند. البته استفاده از نام‌های تجاری توصیه نمی‌شود، چون این نام‌ها سریع‌تر از محتوای کتابخانه‌های برنامه‌نویسان تغییر می‌کنند!

نکته: برای درک بهتر قانون "عدم استفاده از عبارات مخففی که رایج نیستند" مثالی از زبان توسعه دهندگان دات‌نت‌فریمورک ذکر می‌شود. در کلاس Color متد زیر باOverload های مختلف در دسترس است:

```
public class Color {
    …
    public static Color FromArgb(…)
    { … }
}
```

همان‌طور که مشاهده می‌شود برای رعایت قوانین فوق به‌جای استفاده از ARGB از عبارت Argb استفاده شده است. اما این نحوه استفاده موجب شده تا این سوال به‌ظاهر خنده‌دار اما درست پیش‌آید:

"چطور می‌شود در این کلاس رنگی را از ARGB تبدل کرد؟ هرچه که من میبینم فقط تبدیل از طریق (Argb) آرگومان b است"!

حال در این نقطه به‌نظر می‌رسد که در اینجا باید قوانین فوق را نقض کرد و از عنوان FromARGB که مفهوم درست را می‌رساند استفاده کرد. اما با کمی دقت متوجه می‌شویم که این قوانین در ابتدا نیز با پیاده‌سازی نشان داده شده در قطعه کد بالا نقض شده‌اند! همه می‌دانیم که عبارت RGB مخفف معروفی برای عبارت Red Green Blue است. اما استفاده از ARGBبرای افزودن کلمه Alpha به ابتدای عبارت مذکور چندان رایج نیست. پس استفاده از مخفف Argb از همان ابتدا اشتباه به‌نظر می‌رسد. بنابراین راه‌حل بهتر می‌تواند استفاده از عنوان FromAlphaRgb باشد که هم قوانین فوق را نقض نکرده و هم مفهوم را بهتر می‌رساند.

دیگر نکات

.1قراردادهای اشاره شده در این سند حاصل کار شبانه روزی تعداد بسیاری از برنامه نویسان در سرتاسر جهان در پروژه‌های بزرگ بوده است. این اصول کلی تنها برای توسعه آسان‌تر و سریع‌تر پروژه‌های بزرگ تعیین شده‌اند و همان‌طور که روشن است تنها ازطریق تجربه دست‌یافتنی هستند. بنابراین چه بهتر است که در این راه از تجارب بزرگان این عرصه بیشترین بهره برده شود.

.2هم‌چنین توجه داشته باشید که بیشتر قراردادهای اشاره شده در این سند از راهنمای نامگذاری تیم توسعه BCL دات‌نت‌فریمورک گرفته شده است. این تیم طبق اعتراف خودشان زمان بسیار زیادی را برای نامگذاری اشیا صرف کرده‌اند و توصیه کرده‌اند که دیگران نیز برای نامگذاری، زمان مناسب و کافی را در توسعه پروژه‌ها درنظر بگیرند.