

داشتن آگاهی در مورد ساختارهای داده‌ها، الگوریتم‌ها و یا عملگرهای بیتی بسیار عالی است و یا تسلط بر نحوه‌ی کارکرد ابزارهایی مانند SharePoint و امثال آن این روزها ضروری است. اما باید در نظر داشت، کدی که امروز تهیه می‌شود شاید فردا یا ماه دیگر یا چند سال بعد نیاز به تغییر داشته باشد، بنابراین دانش زیبا نوشتن یک قطعه کد که خواندن آنرا ساده‌تر می‌کند و در آینده افرادی که از آن نگهداری خواهند کرد زیاد "زجر" نخواهند کشید، نیز ضروری می‌باشد. (اگر کامنت‌های سایت را خوانده باشید یکی از دوستان پیغام گذاشته بود، اگر به من بگویند یک میلیون بگیرید و برنامه فعلی را توسعه دهید یا رفع اشکال کنید، حاضرم 10 هزار تومان بگیرم و آنرا از صفر بنویسم! متاسفانه این یک واقعیت تلخ است که ناشی از عدم خوانا بودن کدهای نوشته شده می‌باشد.)

در ادامه یک سری از اصول زیبا نویسی کدها را بررسی خواهیم کرد.

1- سعی کنید میزان تو در تو بودن کدهای خود را محدود کنید.  
 لطفا به مثال زیر دقت نمایید:

```
void SetA()
{
    if(a == b)
    {
        foreach(C c in cs)
        {
            if(c == d)
            {
                a = c;
            }
        }
    }
}
```

توصیه شده است فقط یک سطح تو در تو بودن را در یک تابع لحاظ کنید. تابع فوق 4 سطح تو رفتگی را نمایش می‌دهد (برای رسیدن به a=c باید چهار بار از tab استفاده کنید). برای کاهش این تعداد سطح می‌توان به صورت زیر عمل کرد:

```
void SetA()
{
    if(a != b)
        return;

    foreach(C c in cs)
        a = GetValueOfA(c);
}

TypeOfA GetValueOfA(C c)
{
    if(c == d)
        return c;

    return a;
}
```

خوانا تر نشد؟!

افزونه‌های CodeRush و refactor pro مجموعه‌ی DevExpress از لحاظ مباحث refactoring در ویژوال استودیو حرف اول را می‌زنند. فقط کافی است برای مثال قطعه کد if داخلی را انتخاب کنید، بلافاصله سه نقطه زیر آن ظاهر شده و با کلیک بر روی آن

امکان استخراج یک تابع از آنرا برای شما به سرعت فراهم خواهد کرد.

```

10
11 void SetA()
12 {
13     int a = 0, b = 0, d = 0;
14     List<int> cs = new List<int>();
15
16     if (a == b)
17     {
18         foreach (var c in cs)
19         {
20             SetAExtracted(ref a, d, c);
21             if (c == d)
22             {
23                 a = c;
24             }
25         }
26     }
27 }
28
29
30
31

```

**Refactor**

- Extract Method
- Introduce ForEachAction

**Extract Method**

Creates a new method from the selected code block. The selection is replaced with appropriate calling code to invoke the newly-declared method.

Parameter	Count
Input	2
Reference Value	1

مثالی دیگر:

```

if (foo) {
    if (bar) {
        // do something
    }
}

```

به صورت زیر هم قابل نوشتن است (جهت کاهش میزان nesting):

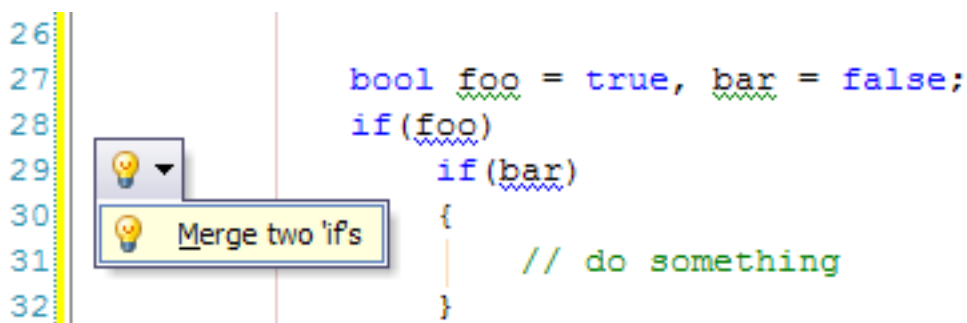
```

if (foo && bar) {
    // do something
}

```

}

افزونه‌ی Resharper امکان merge خودکار این نوع if ها را به همراه دارد.



```

26
27 bool foo = true, bar = false;
28 if(foo)
29     if(bar)
30     {
31         // do something
32     }

```

و یا یک مثال دیگر:

میزان تو در تو بودن این تابع جاوا اسکریپتی را ملاحظه نمائید:

```

function findShape(flags, point, attribute, list) {
    if(!findShapePoints(flags, point, attribute)) {
        if(!doFindShapePoints(flags, point, attribute)) {
            if(!findInShape(flags, point, attribute)) {
                if(!findFromGuide(flags, point)) {
                    if(list.count() > 0 && flags == 1) {
                        doSomething();
                    }
                }
            }
        }
    }
}

```

آنرا به صورت زیر هم می‌توان نوشت با همان کارایی اما بسیار خواناتر:

```

function findShape(flags, point, attribute, list) {
    if(findShapePoints(flags, point, attribute)) {
        return;
    }

    if(doFindShapePoints(flags, point, attribute)) {
        return;
    }

    if(findInShape(flags, point, attribute)) {
        return;
    }

    if(findFromGuide(flags, point)) {
        return;
    }

    if (!(list.count() > 0 && flags == 1)) {
        return;
    }

    doSomething();
}

```

2- نام‌های با معنایی را برای متغیرها و توابع خود انتخاب کنید.

با وجود پیشرفت‌های زیادی که در طراحی و پیاده سازی IDE ها صورت گرفته و با بودن ابزارهای تکمیل سازی خودکار متن تایپ شده در آن‌ها، این روزها استفاده از نام‌های بلند برای توابع یا متغیرها مشکل ساز نیست و وقت زیادی را تلف نخواهد کرد. برای مثال به نظر شما اگر پس از یک سال به کدهای زیر نگاه کنید کدامیک خود توضیح دهنده‌تر خواهند بود (بدون مراجعه به مستندات موجود)؟

```
void UpdateBankAccountTransactionListWithYesterdaysTransactions()
//or?
void UpdateTransactions()
```

3- تنها زمانی از کامنت‌ها استفاده کنید که لازم هستند.

اگر مورد 2 را رعایت کرده باشید، کمتر به نوشتن کامنت نیاز خواهد بود. از توضیح موارد بدیهی خودداری کنید، زیرا آن‌ها بیشتر سبب اتلاف وقت خواهند شد تا کمک به افراد دیگر یا حتی خود شما. همچنین هیچگاه قطعه کدی را که به آن نیاز ندارید به صورت کامنت شده به مخزن کد در یک سیستم کنترل نگارش ارسال نکنید.

```
//function thisReallyHandyFunction() {
//    someMagic();
//    someMoreMagic();
//    magicNumber = evenMoreMagic();
//    return magicNumber;
//}
```

زمانیکه از ورژن کنترل استفاده می‌کنید نیازی به کامنت کردن قسمتی از کد که شاید در آینده قرار است مجدداً به آن بازگشت نمود، نیست. این نوع سطرها باید از کد شما حذف شوند. تمام سیستم‌های ورژن کنترل امکان revert و بازگشت به قبل را دارند و اساساً این یکی از دلایلی است که از آن‌ها استفاده می‌شود! به صورت خلاصه جهت نگهداری سوابق کدهای قدیمی باید از سورس کنترل استفاده کرد و نه به صورت کامنت قرار دادن آن‌ها.

از کامنت‌های نوع زیر پرهیز کنید که بیشتر سبب رژه رفتن روی اعصاب خواننده می‌شود تا کمک به او! (خواننده را بی‌سواد فرض نکنید)

```
// Get the student's id
thisId = student.getId();
```

کامنت زیر بی معنی است!

```
// TODO: This is too bad. FIX IT!
```

اگر شخص دیگری به آن مراجعه کند نمی‌داند که منظور چیست و دقیقاً مشکل کجاست. شبیه به افرادی که به فروم‌ها مراجعه می‌کنند و می‌گویند برنامه کار نمی‌کند و همین! طرف مقابل علم غیب ندارد که مشکل شما را حدس بزند! به توضیحات بیشتری نیاز است.

4- عدم استفاده از عبارات شرطی بی‌مورد هنگام بازگشت دادن یک مقدار bool:

مثال زیر را در نظر بگیرید:

```
if (foo>bar) {
```

```
return true;
} else {
return false;
}
```

آن را به صورت زیر هم می توان نوشت:

```
return foo>bar;
```

5- استفاده از متغیرهای بی مورد:

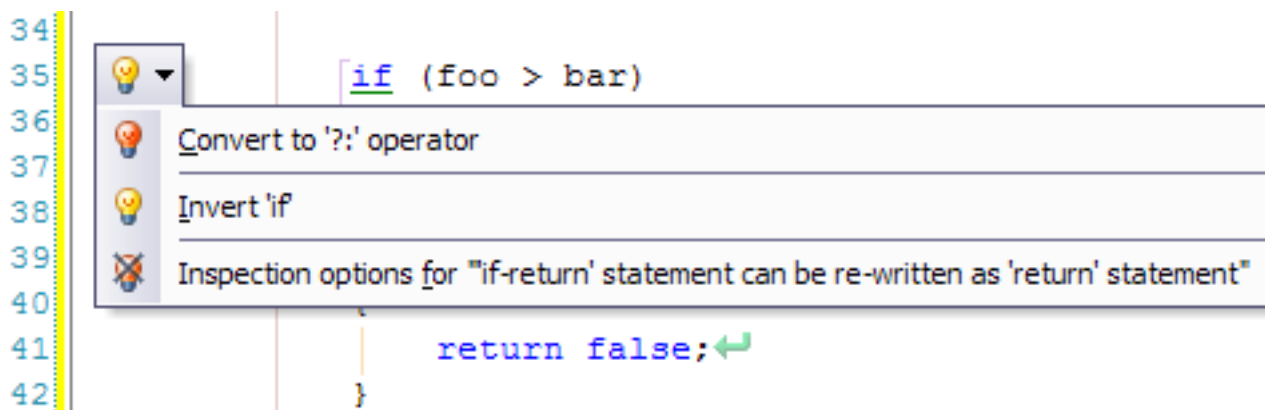
برای مثال:

```
Something something = new Something(foo);
return something;
```

که می شود آن را به صورت زیر هم نوشت:

```
return new Something(foo);
```

البته یکی از خاصیت های استفاده از افزونه ی Resharper ویژوال استودیو، گوشزد کردن و اصلاح خودکار موارد 4 و 5 است.



6- در نگارش های جدید دات نت فریم ورک استفاده از ArrayList منسوخ شده است. بجای آن بهتر است از لیست های جنریک استفاده شود. کدی که در آن از ArrayList استفاده می شود طعم دات نت فریم ورک 1 را می دهد!

7- لطفا بین خطوط فاصله ایجاد کنید. ایجاد فواصل مجانی است!

دو تابع جاوا اسکریپتی زیر را (که در حقیقت یک تابع هستند) در نظر بگیرید:

```
function getSomeAngle() {
// Some code here then
radAngle1 = Math.atan(slope(center, point1));
}
```

```

radAngle2 = Math.atan(slope(center, point2));
firstAngle = getStartAngle(radAngle1, point1, center);
secondAngle = getStartAngle(radAngle2, point2, center);
radAngle1 = degreesToRadians(firstAngle);
radAngle2 = degreesToRadians(secondAngle);
baseRadius = distance(point, center);
radius = baseRadius + (lines * y);
p1["x"] = roundValue(radius * Math.cos(radAngle1) + center["x"]);
p1["y"] = roundValue(radius * Math.sin(radAngle1) + center["y"]);
pt2["x"] = roundValue(radius * Math.cos(radAngle2) + center["x"]);
pt2["y"] = roundValue(radius * Math.sin(radAngle2) + center["y"]);
// Now some more code
}

```

```

function getSomeAngle() {
  // Some code here then
  radAngle1 = Math.atan(slope(center, point1));
  radAngle2 = Math.atan(slope(center, point2));

  firstAngle = getStartAngle(radAngle1, point1, center);
  secondAngle = getStartAngle(radAngle2, point2, center);

  radAngle1 = degreesToRadians(firstAngle);
  radAngle2 = degreesToRadians(secondAngle);

  baseRadius = distance(point, center);
  radius = baseRadius + (lines * y);

  p1["x"] = roundValue(radius * Math.cos(radAngle1) + center["x"]);
  p1["y"] = roundValue(radius * Math.sin(radAngle1) + center["y"]);

  pt2["x"] = roundValue(radius * Math.cos(radAngle2) + center["x"]);
  pt2["y"] = roundValue(radius * Math.sin(radAngle2) + center["y"]);
  // Now some more code
}

```

کدامیک خواناتر است؟

استفاده از فاصله بین خطوط در تابع دوم باعث بالا رفتن خوانایی آن شده است و این طور به نظر می‌رسد که سطرهایی با عملکرد مشابه در یک گروه کنار هم قرار گرفته‌اند.

8- توابع خود را کوتاه کنید.

یک تابع نباید بیشتر از 50 سطر باشد (البته در این مورد بین علما اختلاف هست!). اگر بیشتر شد بدون شک نیاز به refactoring داشته و باید به چند قسمت تقسیم شود تا خوانایی کد افزایش یابد. به صورت خلاصه یک تابع فقط باید یک کار را انجام دهد و باید بتوان عملکرد آن را در طی یک جمله توضیح داد.

9- از اعداد جادویی در کدهای خود استفاده نکنید!

کد زیر هیچ معنایی ندارد!



```

if(mode == 3){ ... }
else if(mode == 4) { ... }

```

بجای این اعداد بی مفهوم باید از enum استفاده کرد:

```
if(mode == MyEnum.ShowAllUsers) { ... }  
else if(mode == MyEnum.ShowOnlyActiveUsers) { ... }
```

10- توابع شما نباید تعداد پارامتر زیادی داشته باشند  
اگر نیاز به تعداد زیادی پارامتر ورودی وجود داشت (بیش از 6 مورد) از struct و یا کلاس جهت معرفی آنها استفاده کنید.

## نظرات خوانندگان

نویسنده: nima

تاریخ: ۱۳۸۷/۱۲/۰۵ ۱۴:۳۱:۰۰

سلام استاد نصیری  
ممنون از مطلب مفیدتون  
این افزونه های CodeRush رو براتون امکان داره جایی آپلود کنین؟  
سپاسگزار

نویسنده: وحید نصیری

تاریخ: ۱۳۸۷/۱۲/۰۵ ۱۵:۱۰:۰۰

سلام،  
نگارش کامل اون‌ها رو در فوروم board4all دات cz می‌تونید پیدا کنید ؛)  
تعداد ایرانی‌هایی را که در اون سایت می‌تونید پیدا کنید باعث تعجب شما خواهد شد!

نویسنده: Sirasad

تاریخ: ۱۳۸۷/۱۲/۰۵ ۲۰:۴۰:۰۰

ممنون آقای نصیری ...  
فکر کنم یکی دیگر از راههای استفاده مجدد از کد ها نوشتن Document برای کد ها و اجزای سیستم می باشد . من خودم به  
شخصه در شرکت با برنامه نویس هام تست کردم , خیلی خوب نتیجه داد .

موفق باشید

نویسنده: وحید نصیری

تاریخ: ۱۳۸۷/۱۲/۰۵ ۲۱:۵۶:۰۰

بله مسلما همینطور است. مطلب فوق فقط بیانگر مواردی بود که از کامنت سوء استفاده شده است.

نویسنده: افشار محبی

تاریخ: ۱۳۸۷/۱۲/۰۷ ۰۸:۲۶:۰۰

خیلی خوب و کاربردی بود. البته شخصا بعضی وقت‌ها مجبور هستم موارد ۴ و ۵ را به خاطر راحتی debug رعایت نکنم.

نویسنده: hajloo

تاریخ: ۱۳۸۷/۱۲/۰۷ ۱۲:۱۳:۰۰

واقعا مفید بود. فک ای کاش لینک ابزارها رو هم برای دانلود می‌زاشتید.  
در ضمن در صورت امکان یک مطلب در مورد استفاده بهینه از خود ویژوال استودیو بنویسید. مثلا اینکه چطور از پنجره های  
موجود استفاده بهتری بکنیم.  
با سپاس فراوان

نویسنده: peyman naji

تاریخ: ۱۳۸۸/۰۷/۱۱ ۲۰:۳۴:۰۱

مهندس ممنون بسیار عالی بود