فاصله ها را رعایت کنیم

فاصله گذاری صحیح چیزی بیشتر از رعایت چند قانون سخت گیرانه است. فاصله ها به کمک ما می آیند تا بتوانیم برنامه ای که نوشته ایم را بهتر درک کنیم و با کمترین زحمت و در سریع ترین زمان اطلاعات قابل قبولی از چیدمان اجزای مختلف برنامه بدست آوریم.
به نوشته زیر دقت کنید:

```
Video provides a powerful way to help you prove your point When you cli
Click Insert and then choose the elements you want from the different
galleries Themes and styles also help keep your document coordinated.
When you click Design and choose a new Theme the pictures charts and Sm
headings change to match the new theme Save time in Word with new butto
```

احتمالا شما هم متوجه شده این که نوشته بالا به شدت *ناخوانا* است، اگرچه هیچ اشکال لغوی یا دستوری در آن وجود ندارد. به عبارت دیگر برای درک آن باید زمان و دقت زیادی صرف این شود. حال آن را به شکل زیر اصلاح می کنیم:

```
    Video provides a powerful way to help you prove your point. When yo

    To make your document look professionally produced, Word provides h
Click Insert and then choose the elements you want from the different
galleries.

    Themes and styles also help keep your document coordinated.
When you click Design and choose a new Theme, the pictures, charts, and
SmartArt graphics change to match your new theme. When you apply styles
headings change to match the new theme.

    Save time in Word with new buttons that show up where you need them
for layout options appears next to it. When you work on a table, click
you want to add a row or a column, and then click the plus sign.
```

به همین سادگی! دیدیم که چگونه فاصله گذاری مناسب و اصولی می تواند یک متن خسته کننده را به یک نوشته خوانا تبدیل کرد، به طوری که بتوان به یک *نگاه* سریع از آن سر در آورد.
مشابه مثال بالا، برنامه ها به ما نیاز دارند تا با کمی تمیزکاری آن ها را از آشفتگی سابق نجات دهیم و آن ها را سازمان مند کنیم. در ادامه خواهیم دید که چگونه می توانیم کدمان را مرتب کنیم تا ساختار منظم تر و پایاتری پیدا کند.

خطوط نه چندان بلند

طولانی بودن بیش از اندازه خطوط باعث میشود تا خوانایی کد به شدت کاهش یابد. طول هر خط بهتر حداکثر به اندازه 80 تا 120 کاراکتر باشد. این مقدار متن در اکثر نمایشگرها و برنامه های نگارشی در یک صفحه قابل نمایش است.

تورفتگی ها (Indentation)

اکثر برنامه هایی که با آن ها سروکار داریم دارای یک ساختار سلسله مراتبی هستند: فایل ها، کلاس ها، متد های درون کلاس ها، بلاک های درون متد ها و در حالت کلی، *بلاک های درون بلاک های دیگر*.
برای اینکه این محدوده ها را به بهترین شکل از یکدیگر تمیز دهیم، باید متناسب با هر کدام از فاصله گذاری مناسب استفاده کنیم. عبارات موجود در سطح فایل ، مانند اکثر *تعاریف کلاس* ، به هیچ وجه فرورفته نیستند. *متدهای درون یک کلاس* در یک سطح فرورفتگی به سمت راست *کلاس* قرار دارند. بلاک های به کار رفته در پیاده سازی متد نیز در یک سطح فرورفتگی به سمت راست متد قرار می گیرند.
و در حالت کلی، *هر بلاک داخلی باید نسبت به بلاک بیرونی یک سطح تورفتگی به سمت راست داشته باشد*. به عنوان مثال:

```
public class Test{
    private int number;

    public void method(){
        innerBlock(){
            operations;
            innerInnerBlock(){
                operations;
            }
        }
    }

    public void anotherMethod(){
        operations;
    }
}
```

کاراکتر Tab

در اکثر برنامه های ویرایش متن، امکان تغییر طول کاراکتر Tab بر حسب تعداد فاصله (space) وجود دارد. این مقدار معمولا 4 یا 8 فاصله در نظر گرفته می شود. توجه کنید که در تمام کدتان از یک فاصله مشخص استفاده شده باشد تا تراز آن در تمامی خطوط یکسان و مشخص باشد.

کاراکترهای متشخص

هر برنامه ای پر از دستورات و عملیات ریاضی و کاراکترهای غیر حرفیست که موارد زیر به افزایش خوانایی آنها کمک می کند :

- همواره **قبل از باز کردن** پرانتز و **بعد از بستن آن** یک فاصله قرار دهید. اما میان پرانتزها با عبارت داخلی آن نباید هیچ فاصله ای وجود داشته با شد.

```
//valid
if (a == 2);

//invalid
if( a == 2 );

//invalid
if(a==2);
```

- بهتر است همواره قبل و بعد از عملگرهای ریاضی یا دودویی از یک فاصله استفاده کنیم. (البته به جز عملگرهای ++ یا ==)

```
//valid
a = b + c;
c = a > b;
b = a && c;

//invalid
a=b+c;
a ++;
```

- بعد از کاراکتر نقطه ویرگول در حلقه ها یک فاصله میگذاریم اما قبل از آن هیچ فاصله ای نباید وجود داشته باشد.

```
//valid
for (int i = 0; i < 10; ++i);

//invalid
for (int j = 0 ;j < 10;++j);
```

کامنت ها

شاید تا کنون کمتر به نقش و اهمیت کامنت ها در برنامه نویسی فکر کرده باشید. کامنت گذاری ابزار قدرتمندی است که خیلی از اوقات می تواند نقش فرشته نجات ما را بازی کند و به طور چشمگیری به صرفه جویی در وقت و هزینه به ما کمک کند.
در ادامه نکات کلیدی را در مورد کامنت ها مرور خواهیم کرد.

کامنت گذاری

چرا کامنت؟

کامنت ها راهنماهایی هستند که به کمک می کنند تا وقتی با یک برنامه بیگانه روبرو می شویم، بدون درگیر شدن بیش از اندازه با جزئیات و نحوه پیاده سازی با کلیتی از عملکرد برنامه، نحوه استفاده از توابع، چیدمان اصلی فایل ها و مواردی از این دست آشنا شویم. همچنین کامنت ها نقش حیاتی در کتابخانه های مختلف بازی می کنند، چرا که می توانند اغلب توضیحات جامعی از نحوه استفاده از امکانات و توابع آن در اختیار کاربر قرار دهند.

یک کامنت خوب

کامنت ها باید به ما بگویند که چه *اتفاقی* دارد می افتد، چگونه انجام می شود، *هر کدام از پارامترها یا آرگومان های مورد استفاده* چه *معنایی دارند و احیانا چه محدودیت ها یا ایراداتی ممکن است وجود داشته باشند.* این موارد به ویژه در مورد متدها بسیار مهم و کاربردی هستند. توجه داشته باشید که کامنت های طولانی را در چند خط بنویسید و از نوشتن همه آن در یک خط خودداری کنید.

البته بسیاری از اوقات ارائه چنین توضیحاتی نیاز نیست، مثلا زمانی که می خواهیم یک متغیر یا عملیات کوتاهی را توصیف کنیم، اغلب از کامنت های یک خطی استفاده می کنیم که صرفا به ماهیت(what) آن اشاره می کنند، و نه به چگونگی(how).

مثلا برای ارائه توضیحات در مورد تابعی که میانگین را محاسبه می کند، کافیست از عبارت "compute mean value" به جای "sum of values divided by n" استفاده کنیم.

زیاده روی نکنیم

کامنتها همانطور که می توانند در بسیاری از موارد به ما کمک کنند، اما استفاده بیش از اندازه از آنها نتیجه عکس داشته و گاهی به کابوسی مهیب بدل می شوند. نباید این نکته را فراموش کنیم که این کامنت ها هستند که در کنار کد ما قرار میگیرند و نه برعکس!

به علاوه درنظر داشته باشید که استفاده از کامنت برای شفاف سازی و ارائه توضیحات بیشتر همواره *آخرین* راه حل می باشد.به عبارت دیگر، **نباید برای یک کد کثیف کامنت گذاشت. بهتر است فکر دیگری به حال آن بکنید!.**
خیلی وقت ها توجه با نامگذاری ها، فاصله ها، طراحی و پیاده سازی اجزا و ... می توانند خیلی بیشتر از یک کامنت طولانی و مبهم به تمیزی کد ما کمک کنند.

کامنت های TODO

گاهی اوقات منطقی است که یادداشت های "TODO" را در قالب کامنت‌های TODO // انجام دهید. TODO کارهایی است که به نظر برنامه نویس باید انجام شود ، اما به دلایلی فعلاً نمی تواند انجام دهد. این ممکن است یک یادآوری برای حذف یک ویژگی منسوخ شده یا یک درخواست برای شخص دیگری برای بررسی یک مشکل باشد. حتی ممکن است درخواستی باشد که شخص دیگری به نام بهتری فکر کند یا یادآوری ای باشد برای ایجاد تغییری وابسته به یک برنامه ریزی. TODO هرچه باشد ، بهانه ای برای گذاشتن کد بد در برنامه نیست.

امروزه ، اکثر IDE های خوب ویژگی های خاصی را برای یافتن همه کامنت‌های TODO ارائه می دهند ، بنابراین احتمالاً گم نمی شوند. اگر هم نمی خواهید کد شما با TODO پر شود، مرتباً از طریق آنها اسکن کرده و مواردی را که می توانید حذف کنید.

```
public void verifyUser (User user){
    //TODO Add some useful code here to verify the user
}
```