Code Style یک جنبه حیاتی در توسعه نرم افزار است، زیرا به طور مستقیم بر خوانایی، قابلیت نگهداری و کیفیت کلی کد تأثیر می گذارد. در اینجا توضیح دقیق تری از دستورالعمل های سبک کدنویسی که قبلاً ذکر کردم آورده شده است:

* قرارداد های نامگذاری

1. از PascalCase برای نام کلاس ها، نام متدها و نام ویژگی ها استفاده کنید.
2. از camelCase برای متغیرهای محلی و پارامترهای روش استفاده کنید.
3. از خط زیر برای فیلدهای خصوصی (مانند \_myPrivateField) استفاده کنید.

* ساماندهی کد

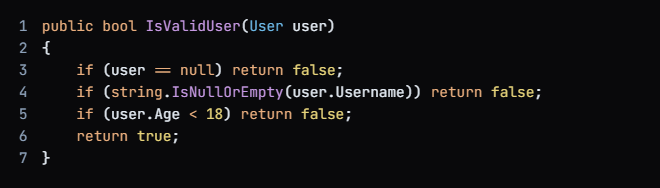
1. کلاس ها و روش ها را کوتاه نگه دارید و روی یک مسئولیت متمرکز کنید.
2. از نام های معنی دار و توصیفی برای کلاس ها، متدها و متغیرها استفاده کنید.
3. کدها را در فضاهای نام و پوشه های منطقی سازماندهی کنید.
4. از مناطق برای گروه بندی بخش های کد مرتبط استفاده کنید (به عنوان مثال، #region Constructors).

* استایل کدنویسی

1. از چهار فاصله برای تورفتگی استفاده کنید.
2. خطوط را کوتاه نگه دارید (کمتر از 120 کاراکتر) و برای خوانایی از شکستن خطوط استفاده کنید.
3. براکت باز را روی همان خط و براکت بسته را روی یک خط جدید قرار دهید.
4. از یک فاصله بین پرانتز بسته و عبارت بعدی استفاده کنید.
5. از استفاده از اختصارات و کلمات اختصاری خودداری کنید، مگر اینکه به طور گسترده شناخته شوند (مثلاً ID به جای Identity).
6. اجتناب از کد تکراری: اگر متوجه شدید که در حال کپی و Past کد هستید، احتمالاً نشانه آن است که باید یک متد یا کلاس را extract کنید.
7. استفاده از Early Return : به جای قرار دادن چند دستور if، از بازگشت های اولیه برای ساده کردن کد و کاهش تورفتگی استفاده کنید. به عنوان مثال 

در این مثال، متد از یک متغیر بولین isValid برای ردیابی نتیجه اعتبار سنجی استفاده می کند. کد تودرتو است و خواندن آن سخت است و درک منطق را دشوار می کند.

اکنون، بیایید کد را برای استفاده از بازده های اولیه تغییر دهیم:



* رسیدگی به خطا

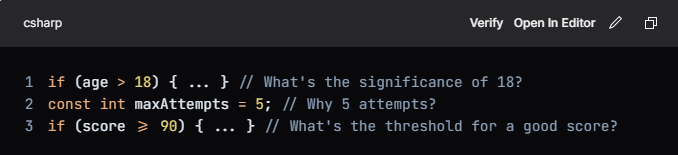
1. از try-catch برای رسیدگی به استثناها استفاده کنید.
2. خطاها و استثناها را با استفاده از یک Framework ثبت کنید (به عنوان مثال، Serilog، NLog).

* Code Smells

1. از کلاس های بزرگ با مسئولیت های زیاد دوری کنید.
2. method های طولانی را به method های کوچکتر و متمرکزتر تبدیل کنید.
3. کدهای تکراری را حذف کنید.

* خوانایی کد

1. از blank lines برای جداسازی بخش های منطقی کد استفاده کنید.
2. از کامنت برای توضیح منطق یا الگوریتم های پیچیده استفاده کنید.
3. از استفاده از اعداد در کد خودداری کنید و به جای آن از Const استفاده کنید (به عنوان مثال، به جای

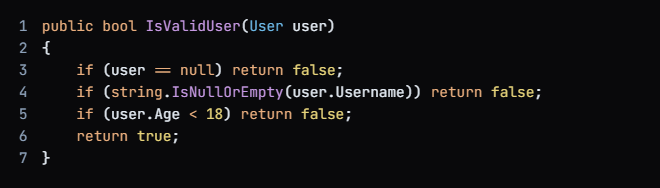


از  استفاده کن.

1. استفاده از Early Return : به جای قرار دادن چند دستور if، از بازگشت های اولیه برای ساده کردن کد و کاهش تورفتگی استفاده کنید. به عنوان مثال
2. 

در این مثال، متد از یک متغیر بولین isValid برای ردیابی نتیجه اعتبار سنجی استفاده می کند. کد تودرتو است و خواندن آن سخت است و درک منطق را دشوار می کند.

اکنون، بیایید کد را برای استفاده از بازده های اولیه تغییر دهیم:



مزایای استفاده از Early Returns عبارتند از:

خوانایی بهبود یافته: درک کد آسان تر است، زیرا هر شرط به طور جداگانه ارزیابی می شود.

تودرتوی کاهش یافته: کد کمتر تودرتو است و پیروی از منطق را آسان تر می کند.

نگهداری آسان تر: افزودن یا حذف شرایط اعتبارسنجی ساده تر است، زیرا هر یک به طور مستقل ارزیابی می شوند.

زمان استفاده از بازده های اولیه:

اعتبارسنجی: هنگام اعتبارسنجی داده‌های ورودی، از Early Return برای ساده‌سازی کد و کاهش تودرتو استفاده کنید.

مدیریت خطا: هنگام رسیدگی به Exception، از Early Return برای خروج سریع از method و اجتناب از منطق پیچیده استفاده کنید.

منطق ساده: وقتی منطق ساده و سرراست است، از Early Return استفاده کنید تا کد را خواناتر و مختصرتر کنید.

به یاد داشته باشید، هدف از استفاده از Early Return ، خوانایی، نگهداری و کارآمدتر کردن کد است.

کیفیت کد فقط نوشتن کد صحیح نیست. این در مورد نوشتن کد قابل نگهداری، کارآمد و مقیاس پذیر است. در اینجا برخی از ملاحظات پیشرفته وجود دارد: