نکاتی که برای یکدستی و مرتب بودن سورس‌های C# لازم است رعایت کنیم:

۱. تمام اعلانات مربوط به ماجول‌های مورد استفاده (دستورات using) خارج از namespace قرار می‌گیرند.
۲. دستورات using را همیشه به صورت مرتب شده نگهداری می‌کنیم. در مرتب‌سازی دستورات using، تمام مواردی که مربوط به پلتفرم اصلی دات نت هستند (و با کلمه System شروع می‌شوند) بدون درنظر گرفتن ترتیب الفبایی، همیشه در ابتدا قرار می‌گیرند.
۳. دستورات using که استفاده نشده‌اند را همیشه حذف می کنیم (دستورات using مربوط به System و System.Collections.Generic به دلیل اینکه خیلی پایه‌ای و متداول هستند را می‌توانیم نگه داریم).
۴. تمام خط‌های خالی اضافه را از داخل تعریف کلاس‌ها و متدها حذف می‌کنیم.
۵. بین تعریف متدهای یک کلاس فقط یک خط خالی فاصله می‌گذاریم.
۶. برای فرورفتگی‌ها (indenting) در سورس به‌هیچ‌وجه از Tab استفاده نمی‌کنیم.
۷. هنگام پیاده‌سازی متدها، بین دستورات مرتبط یا بعد از آکولاد باز ({) یا قبل از آکولاد بسته (}) خط خالی قرار نمی‌دهیم.
۸. برای بالابردن خوانایی کد از خطوط خالی اضافه استفاده نمی‌کنیم.
۹. کدهای آزمایشی یا نامربوط به پروژه اصلی را حتما قبل از check-in کردن از پروژه حذف می‌کنیم.
۱۰. کدهای کامنت‌شده را فقط در صورتی درون سورس‌ها نگه می‌داریم که بعدا بخواهیم از آنها استفاده کنیم. حتی در این صورت نیز بهتر است این کدهای کامنت‌شده بصورت طولانی‌مدت داخل سورس‌ها باقی نمانند. در چنین شرایطی، حتما دلیل کامنت کردن کد را بصورت مختصر و گویا در ابتدای کد توضیح می‌دهیم.
۱۱. بین دستورات using هیچ خط خالی قرار نمی‌دهیم. این دستورات نیازی به طبقه‌بندی ندارند.
۱۲. حداکثر طول هر خط از کد را بین ۱۰۰ تا ۱۲۰ کاراکتر نگه می‌داریم. خطوط طولانی‌تر را با استفاده از فرورفتگی (indent) مناسب می‌شکنیم.
۱۳. فقط در مواردی که کد موجود پیچیدگی خاص منطقی دارد، از توضیحات (comment) استفاده می‌کنیم. در این موارد، تا حد امکان سعی می‌کنیم بصورت دوره‌ای این توضیحات را بازنگری کنیم تا صحیح و بروز باقی بمانند.
۱۴. مدیریت استثنائات (exceptions) را تا حد امکان داخل کلاس‌های مختلف پخش نکرده و بصورت مرکزی انجام می‌دهیم. در صورتی که مدیریت یک یا چند استثنا واقعا ضروری است، آن را داخل سورس انجام می‌دهیم.
۱۵. هنگام مدیریت استثنائات، در صورتی که مشکل اصلی برطرف نمی‌شود، حتما از دستور throw; استفاده کرده و هرگز exception را به حال خود رها نمی‌کنیم (اصطلاحا: آن را نمی‌بلعیم).
۱۶. مستندسازی‌های خارج از کلاس‌ها، متدها و ویژگی‌ها (properties) را تا حد امکان به زبان فارسی انجام می‌دهیم.
۱۷. در صورت استفاده از زبان انگلیسی در مستندات XML، حتما اولین کلمه توضیحات در ابتدای هر جمله را با حرف بزرگ شروع می‌کنیم.
۱۸. برای تمام کلاس‌ها، متدها، ویژگی‌ها و غیره، نوع دسترسی را همیشه بصورت صریح قید می‌کنیم (کلمات public، private و...). دسترسی پیش‌فرض بین نسخه‌های مختلف پلتفرم دات نت ممکن است تغییر کند. به‌علاوه، قرار ندادن این کلمات به وضوح کد لطمه می‌زند.
۱۹. نام تمام فیلدهای داخلی (private یا protected) را با کاراکتر زیرخط (underscore یا _) شروع کرده و برای مشخص کردن اعضای کلاس تا حد امکان از بکارگیری کلمه this پرهیز می‌کنیم.
۲۰. هنگام اضافه کردن پروژه به یک solution، نام پروژه، اسمبلی و فولدر اصلی پروژه (و در بیشتر مواقع، namespace پیش‌فرض) را یکسان درنظر می‌گیریم.
۲۱. تمام پروژه ها و namespace های مربوط به محصول تدبیررا با پیشوند SPPC.Tadbir و تمام پروژه ها و namespace های مربوط به زیرساخت تدبیر را با پیشوند SPPC.Framework نامگذاری می‌کنیم.

۲۲. در هر فایل C# بیشتر از یک کلاس قرار نمی‌دهیم و نام فایل را با نام کلاس مربوطه یکسان در نظر می‌گیریم.
۲۳. اینترفیس ها و پیاده‌سازی پیش‌فرض آنها را در یک پروژه و اسمبلی قرار نمی‌دهیم.
۲۴. اینترفیس ها را در یک پروژه و اسمبلی مرکزی تعریف کرده و با استفاده از فولدرها طبقه‌بندی می‌کنیم.
۲۵. تا حد امکان رفرنس های استفاده نشده و اضافی را از پروژه ها حذف می‌کنیم.
۲۶. متن های ثابت را (که بین دو کاراکتر " قرار می‌گیرند) هرگز داخل سورس ها استفاده نمی‌کنیم. آنها را بصورت طبقه‌بندی‌شده در یک کلاس و بعنوان ویژگی های ثابت (const) تعریف کرده و در کلاس های مورد استفاده رفرنس می‌دهیم.
۲۷. آدرس های اینترنتی (URL) را همیشه با حروف کوچک (lower-case) تعریف و استفاده می‌کنیم.
۲۸. هنگام نامگذاری کلاس ها، متدها، متغیرها و غیره از کاراکتر زیرخط (_) برای جدا کردن کلمات استفاده نمی‌کنیم.
۲۹. کدهایی که بصورت آماده و از طریق اینترنت وارد پروژه ها می‌کنیم، لازم است با دقت با استانداردهای کدنویسی موجود مطابقت بدهیم.
۳۰. هنگام تعریف یک کلاس، بخش های عمومی (public) را در ابتدا قرار می‌دهیم، پس از آن بخش های محافظت‌شده (protected) و در انتها بخش های خصوصی (private) را قرار می‌دهیم.
۳۱. برای تعریف سطح دسترسی ها، تا حد امکان از سطح دسترسی داخلی (internal) استفاده نمی‌کنیم.
۳۲. سطوح مختلف ارث‌بری (inheritance) را در کلاس ها در سطح حداقل نگه داشته و برای پیاده سازی کلاس های پیچیده، بین ارث‌بری و ترکیب (composition)، تا حد امکان مکانیزم ترکیب را ترجیح می‌دهیم.