## Table of Contents



**Item 19: Treat class design as type design**

در C++، همانند سایر زبان‌های برنامه‌نویسی شیءگرا، تعریف یک کلاس جدید به مثابه تعریف یک type جدید می‌باشد. بیشتر زمان شما به عنوان توسعه‌دهنده‌ی C++ صرف نوشتن و تغییر دادن type system ها خواهد شد.این بدین معنی است که شما تنها یک طراح کلاس نیستید، بلکه طراح type هستید. توابع سربارگذاری و اپراتورها، کنترل کردن تخصیص حافظه و رهاسازی حافظه، تعریف initialization شیء و finalization، این‌ها همه در دست شما خواهد بود. بنابراین باید رویکرد طراحی کلاس شما مشابه طراحی‌ای باشد که زبان C++ در مورد typeهای built-in رعایت می‌کند.

طراحی کلاس خوب یک کار پرچالش می‌باشد چرا که type های خوب پرچالش می‌باشد. typeهای خوب دارای یک syntax طبیعی هستند، دارای معنای بصری، و یک یا چندین پیاده‌سازی کارآمد دارند. در C++، یک طرح ضعیف باعث می‌شود که به هیچکدام از این هدف‌ها نرسیم. حتی ممکن است که کارآمدی توابع عضو کلاس نیز تحت تاثیر چگونگی این طراحی قرار بگیرد.

در این صورت این سوال مطرح است که آیا شما کلاس‌های کارآمدی رو طراحی می‌کنید یا خیر؟ در ابتدا، نیاز داریم که با مشکلاتی که در این راه روبه‌رو هستیم آشنا شویم. تقریبا در مورد هر کلاسی نیازمند هستیم که با سوالاتی که در ادامه خواهیم آورد روبه‌رو شوید، پاسخ‌هایی که به این سوالات میدهید ممکن است که طراحی شما را محدود کند.

- **چطور باید اشیاء با استفاده از new ساخته و نابود شوند؟** چگونگی این کار بر روی سازنده‌ی کلاس و مخرب کلاس تاثیر می‌گذارد، همچنین توابع تخصیص حافظه و رهاسازی آن نیز(new operator,operator new[],operator delete[],operator delete برای اطلاعات بیشتر فصل هشتم را مطالعه نمایید) مواردی هستند که در این سوال روی آن‌ها تاثیر گذاشته می‌شود.
- **بین initialization شیء با انتساب شیء چه تفاوتی باید وجود داشته باشد؟** پاسخی که به این سوال می‌دهیم، تفاوت سازنده‌ی کلاس و اپراتور انتساب را مشخص می‌کند. این خیلی مهم است که تفاوت بین initialization و assignment را درک کنیم، چرا که این‌ها دارای توابع متفاوتی هستند که فراخوانی می‌شود(برای این مورد آیتم را ببینید).
- **پاس دادن شیء کلاستان به صورت pass-by-value چه معنایی خواهد داشت؟** به یاد بیاورید، که کپی سازنده(copy constructor) این را مشخص میکرد که چطور pass-by-value برای یک type پیاده‌سازی می‌شود.

- **چه محدودیت‌هایی بر روی مقادیر قابل قبول روی این type جدید وجود دارد؟** معمولا، فقط برخی از ترکیب‌ها از مقادیر برای داده‌های عضو کلاس قابل قبول یا valid هستند. این ترکیب‌ها مشخص‌کننده‌ی تنوع کلاس شما بوده که شما باید آن را حفظ کنید. این تنوع مشخص کننده‌ی یک سری error checkingهایی بوده که باید در داخل توابع عضو بیاورید، مخصوصا در مورد سازنده‌ی کلاس ، اپراتور انتساب و توابع setter . همچنین این مورد تاثیر خود را بر روی exception هایی که توابع دارند نیز میگذارد.
- **آیا این type جدید منطبق بر روی گراف ارث‌بری می‌باشد؟** اگر شما کلاس را از یک کلاس موجود ارث‌بری کرده‌اید، در این صورت شما با طراحی‌ای که در آن کلاس‌ها شده محدود هستید، مخصوصا این که توابع آن ها به صورت virtual بوده یا نه(آیتم ۳۴ و ۳۶ را ببینید). اگر این اجازه را میدهید که کلاس‌های دیگر از کلاس شما ارث‌بری کنند، در این صورت این مورد روی این که توابع را به صورت virtual تعریف کنید تاثیر می‌گذارد، مخصوصا بر روی مخرب کلاس(آیتم ۷ را ببینید).
- **چه تبدیلاتی بر روی type شما اجازه داده شده؟** type شما در دریایی است که همه جور type در آن وجود دارد، بنابراین آیا باید بین type شما و سایر type ها یک تبدیل وجود داشته باشد؟ اگر دوست داشته باشید که اشیاء با نوع T1 بتوانند به صورت غیر صریح به اشیاء با نوع T2 تبدیل شوند، در این صورت باید یا یک تبدیل نوع در کلاس T1 داشته باشید(یعنی اپراتور T2) و یا یک سازنده‌ی غیرصریح در کلاس T2 داشته باشید که با یک آرگومان بتوان آن را فراخوانی کرد.

  اگر می‌خواهید که تنها تبدیلات صریح انجام شود، در این صورت باید یک تابع برای چنین تبدیلاتی بنویسید، ولی نیاز خواهید داشت که از تبدیل با operator و یا سازنده‌ی غیر صریح اجتناب کنید.( برای یک مثال برای هر دو تبدیل صریح و غیر صریح آیتم ۱۵ را ببینید).
- **چه اپراتورها و توابعی برای type جدید لازم است؟** پاسخی که به این سوال میدهیم تعیین کننده‌ی توابعی است که برای کلاس تعریف می‌کنیم. برخی توابع، توابع عضو خواهند بود، ولی برخی دیگر نه(آیتم ۲۳و ۲۴و ۴۶ را ببینید).
- **چه توابع استانداردی اجازه‌ی استفاده شدن ندارند؟** توابعی که به این صورت اجازه‌ی استفاده شدن ندارند باید به صورت private اعلان شوند(آیتم ۶ را ببینید).
- **چه کسانی اجازه‌ی دسترسی به اعضای کلاس را دارند؟** این سوال به شما کمک می‌کند تا بفهمید که کدام‌یک از اعضا به صورت public باید تعریف شوند و کدام یک به صورت private بایستی تعریف شوند، و کدام یک باید به صورت protected تعریف شوند. همچنین این به شما کمک می‌کند تا بفهمید چه کلاس‌ها و یا توابعی بایستی دوست باشند، همچنین این که باعث می‌شود که یک کلاس را درون یک کلاس دیگر تعریف کنیم.

- **در مورد type جدید، چه چیزی undeclared interface می‌باشد؟** با داشتن کارآمدی خوب، امنیت استثناء(آیتم ۲۹)، و استفاده از منابع(یعنی lock ها و حافظه‌های داینامیک) چه نوع تضمینی را ارایه می‌دهد؟ ضمانتی که شما پیشنهاد می‌دهید ممکن است طراحی کلاس را محدود کند.
- **Type جدید شما چقدر جامع است؟** شاید شما واقعا یک type جدید را طراحی نمی‌کنید. شاید در حال تعریف کردن یک خانواده‌ای از type ها هستید. در این صورت، نیازی به تعریف یک کلاس جدید نیست، بلکه شما نیاز به یک کلاس template دارید.
- **آیا واقعا این type جدید چیزی است که نیاز دارید؟** اگر دارید یک کلاس جدید مشتق شده تعریف می‌کنید که تنها چند تابع به کلاس موجود اضافه کنید، شاید بهتر است که برای نیل به این هدف یک یا چند تابع غیر عضو و یا template تعریف کنید.

پاسخ دادن به این سوالات دشوار است، بنابراین تعریف کلاس‌های موثر و کارآمد یک کار چالش برانگیز می‌باشد. با این وجود، کلاس‌های تعریف شده توسط کاربر اگر به خوبی طراحی شوند می‌توانند به اندازه‌ی type های built-in خوب باشند، و این باعث می‌شود که همه‌ی این کارها ارزش خود را داشته باشد.