 两段构造也是声名狼藉得很，比之于MFC，好不了多少，貌似MFC中到处都是两段构造，难道两段构造的声誉也是受MFC所累。定义完了一个对象变量之后，还要再调用一次该对象的Create函数，而且还要Create成功了之后，才能对该对象做进一步的操作，否则对象将一直处于非法状态。这种代码方式写起来确实很恶心，为何不直接在构造函数中直接Create，不成功就抛出异常，然后对象就流产了，好过它半死不活地一直苟延残喘于世上，累己累人。其实，MFC选择两段构造也是有苦衷：1、先是很久很久以前，VC编译器对异常的支持不怎么好，当然，现在的VC编译器，自然今时不比往日，但是，还要兼容以往的代码；2、然后是MFC的设计，它只是对API做了一层薄薄的包装，薄薄的意思，就是，不管怎么捣鼓，都难以将WINDOWS系统中的各种对象包装成一个干净的C++对象了，因为，API本身就采用两段构造。可不是吗？定义一个句柄变量，然后CreateXXX返回结果，返回值非法，表示创建失败。失败了，还要霸王硬上弓，后果会怎么样，这谁也不知道。  
        理论上，构造函数抛出异常确实很优雅，代码也更具美感，并且，其行为也更加明确，要么就处理，要么，就等着程序异常退出。但是，实际上，异常这种东西，真正实现执行起来，却相当的困难。更何况，如果完全丢弃两段法，除了异常，还会引入一些新的问题，正所谓：“前门驱虎，后门进狼”，进来不只是一只狼，而是好几只。生活的奥妙，就在于制造出新的问题，以解决旧的问题。  
        构造函数中直接调用Create，就表示了用户一定义一个类型变量，程序就会马上启动Create函数，也就意味着可能将创建窗口对象、内核对象、甚至启动新的线程等等，这些操作都不是省油的灯，构造函数中做了太多事情，会有隐藏太多细节之嫌，代码本来就是为了隐藏细节，这个多事之罪名暂且不论；但是，用户没法对创建过程Say NOT，也即是说，用户一定义对象变量，就只能接受它的高昂的创建过程。难道，一开始就让对象进入有效状态，这都有错吗？确实是的。有时候，用户只是先想声明（定义）对象，等必要（时机成熟）的时候，再让它进入有效状态。咦，用户这样写代码，不太好吧，应该强制他/她等到了那个时候，再定义对象变量。变量怎么可以随随便便就定义呢？应该在要使用的时候，才定义它，这才是良好的代码风格。但是，有些情况，确实需要先暂时定义非法状态下的对象变量，比如，这个对象是另一个对象（拥有者）的成员变量时，那也没什么，强制用户在必要的时候，才定义拥有者对象变量。但是，假如这个拥有者必须是全局变量，那该怎么办？那也没什么，将拥有者定义为指针变量就是了？好了，本来只是要对象创建失败的情况，现在还要考虑内存分配的细节，然后接着就是new delete，然后就是各种智能指针闪亮登台演出，更糟糕的是，对象有效无效的问题依然没有根除，因为，只要引入指针，每次使用指针，就必须检查指针是否有效，咦，难道操作空指针不会抛出异常吗？C++规范中，操作空指针属后果未确定的行为，对C++而言，未确定往往就是最糟糕的意思。此外，鉴于对象只能一直处于有效状态，它就不可能提供让对象进入无效状态的操作。如果想要让对象无效，唯一的办法，就是让它死去，强制对象启动析构函数，方法是离开作用域强者delete它。下次要使用它的时候，就再new一次或者定义一次，不，它已经是另外一条新生命了。但是，对于两段构造的对象，只须Destroy又或者Create，对象可以永远只有一个。此外，二段构造颇具扩展性，很轻易地就可搞成三段构造，每一步，用户都有选择的权利。但构造异常就没有这个优点。  
        考虑到构造函数中的参数问题，比如，月份的参数，大家都知道，有效值只在1-12月之间。不讨论这种情况下，非法的参数传递是否属于代码的逻辑问题。对此，构造异常指导下的对象是不可能出现无参（没有参数或者参数都有缺省值）的构造函数，因此，它们也都不能用于数组，难以应用于全局变量、静态变量、作为其他对象的数据成员，如果非要在这些场合下使用它们，比如占位符的作用，唯有用上指针，于是伴随而来的，又如上文所述，使用指针之前，必须检查指针的有效性，只怕不会比检查二段构造的有效性好多少。  
        二段构造不轻易剥夺用户的权利，提供更多选择，可用于数组、堆栈、STL中的容器，要它死，它就死，要它活，它就活，但是，它可以从来都未曾消失过，要做的，仅仅是在使用它时，清楚它是死是活就行了，不过多加几次判断而已。相比之下，构造异常就更具侵入性了，一旦用上，就只能被迫遵照它的规则行事。  
        其实，两段构造与构造异常，都很恶心，只要一处代码中用到了它，所有与之相关的代码都没法脱身。差别不过在于谁比谁恶心而已，这个，视各人的口味而不同。对于本人这种害怕分配内存，释放内存，更加畏惧异常的人来说（这并不表示本人写不出异常安全的代码），当然优先选择二段构造，MORE EFFECTIVE的条款中，声称，如无必要，不要提供缺省的构造函数，以免对象陷入半死不活的状态中。而我的习惯作法则是，如无必要，必须提供缺省的构造函数，不要轻易剥夺用户想要使用对象数组的权利，或者是由于不提供缺省的构造函数，而由此引起的种种不便。  
        好了，既然程序中决定用二段构造了，那么，假如用户定义了一个对象，忘了再构造一次，但是又要执行其他操作，怎么办？嗯，那也没什么，既然用户不遵守契约，我们的对象自然可以做出种种不确定的行为。当然，别忘了，在其他的每一个操作上都添加几条assert语句，尽管这很恶心，也聊胜于无，减少点罪恶感，以便于在调试版中找出问题。