virtual 함수 소개

[프로젝트 생성]
(1) "Visual C++" -> "일반" –> "빈 프로젝트" -> 새 소스 파일 준비
(2) Item 클래스 준비

```
#include <iostream>
using namespace std;

class Item {
public:
    int m_x1, m_y1, m_x2, m_y2;

    Item(int x1, int y1, int x2, int y2) {
        m_x1 = x1; m_y1 = y1; m_x2 = x2; m_y2 = y2;
    }
    void Print() {
        cout << "Item : (" << m_x1 << ", " << m_y1
        << ") (" << m_x2 << ", " << m_y2 << ")" << endl;
    }
};
```

virtual 함수 소개

(3) main 함수에서 object 2개 할당 받아 Print()

```
int main() {
    Item *items[2] = { NULL, NULL };
    int i;

    items[0] = new Item(10, 10, 20, 20);
    items[1] = new Item(100, 100, 200, 200);

    for (i = 0; i < 2; i++)
        items[i]->Print();

    for (i = 0; i < 2; i++)
        delete items[i];
    return 0;
}
```

virtual 함수 소개

(5) Item 에서 상속받아 클래스 2개 생성

```
class Rect : public Item  {
public:
    Rect(int x1, int y1, int x2, int y2) : Item(x1, y1, x2, y2)
    {}
    void Print() {
        cout << "Rect : (" << m_x1 << ", " << m_y1
        << ") (" << m_x2 << ", " << m_y2 << ")" << endl;
    }
};
class Circle : public Item {
public:
    Circle(int x1, int y1, int x2, int y2) : Item(x1, y1, x2, y2)
    {}
    void Print() {
        cout << "Circle : (" << m_x1 << ", " << m_y1
        << ") (" << m_x2 << ", " << m_y2 << ")" << endl;
    }
};
```

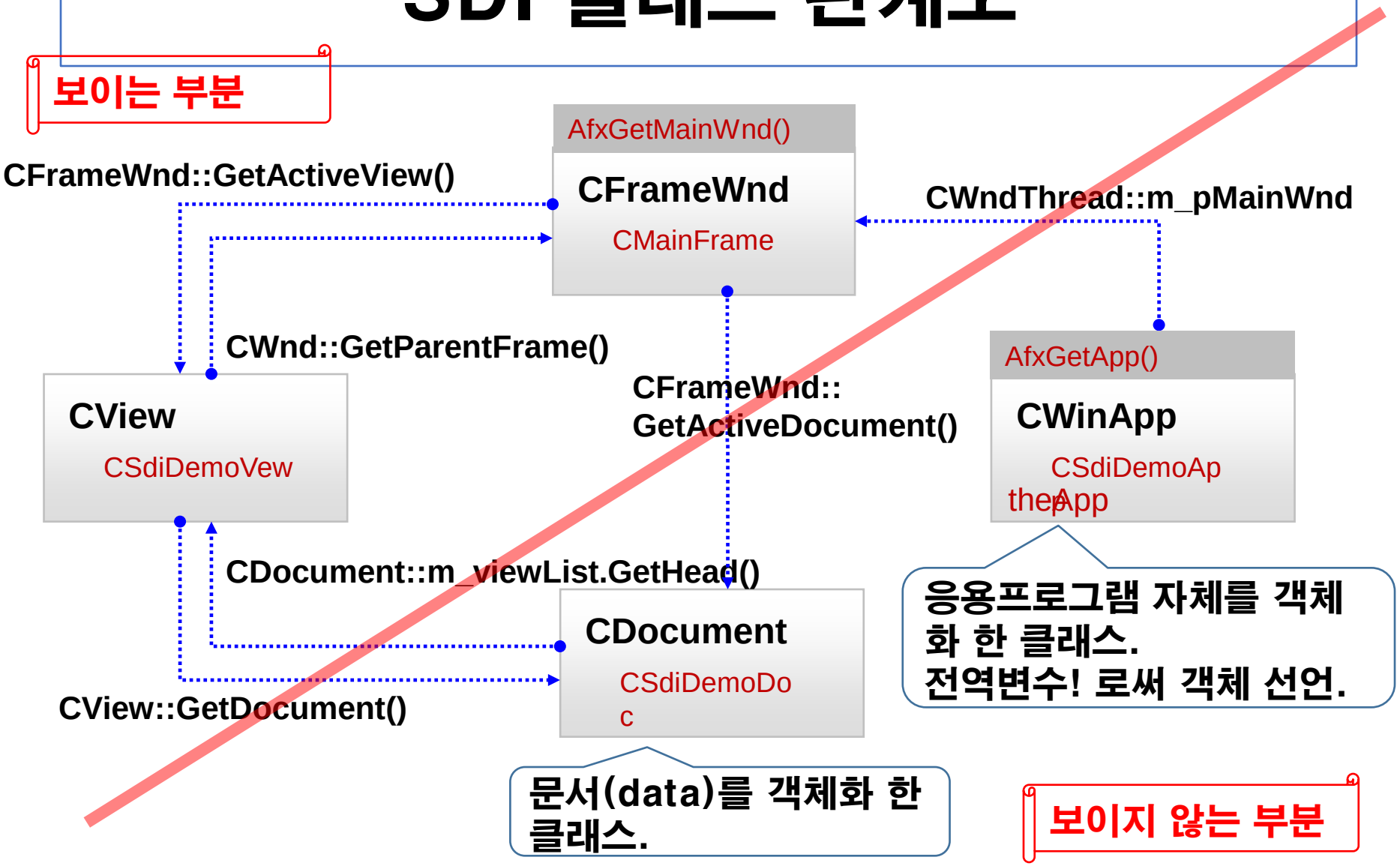
virtual 함수 소개

(6) main 에서 Item 대신 Rect, Circle object를 생성하여 Item pointer에 저장.

```
int main() {
    Item *items[2] = { NULL, NULL };
    int i;

    //items[0] = new Item(10, 10, 20, 20);
    //items[1] = new Item(100, 100, 200, 200);
    items[0] = new Rect(10, 10, 20, 20);
    items[1] = new Circle(100, 100, 200, 200);

    for (i = 0; i < 2; i++)
        items[i]->Print();

    for (i = 0; i < 2; i++)
        delete items[i];
    return 0;
}
```

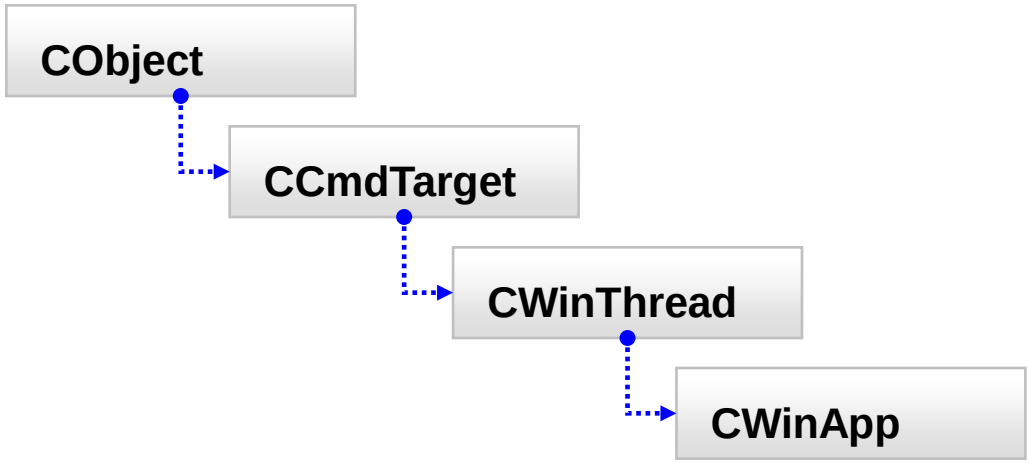
실제 객체는 Rect, Circle 타입이지만 Item pointer 사용하여 Print()하면 Item::Print()가 수행됨!
-> Print() 함수를 virtual로 만들자.

SDI 클래스 관계도


보이는 부분
AfxGetMainWnd()
CFrameWnd::GetActiveView()
CFrameWnd
CMainFrame
CWndThread::m_pMainWnd
CWnd::GetParentFrame()
CFrameWnd::
GetActiveDocument()
AfxGetApp()
CView
CSdiDemoVew
CWinApp
CSdiDemoApp
theApp
CDocument::m_viewList.GetHead()
응용프로그램 자체를 객체화 한 클래스.
전역변수! 로써 객체 선언.
CDocument
CSdiDemoDoc
CView::GetDocument()
문서(data)를 객체화 한 클래스.
보이지 않는 부분

CWinApp

응용 프로그램 자체를 구현한 클래스. 그러므로 이의 객체는 "응용 프로그램".



CWinApp 주요 멤버

멤버 변수	기능
m_hInstance	현재 응용 프로그램의 인스턴스 핸들입니다. WinMain( ) 함수의 첫 번째 파라미터인 hInstance와 같은 것입니다.
m_lpCmdLine	WinMain( ) 함수의 lpCmdLine 파라미터와 같은 것입니다. 프로그램을 실행하였을 때 명령줄(Command-line) 정보가 들어 있습니다.
m_nCmdShow	WinMain( ) 함수의 마지막 파라미터인 nCmdShow와 같은 것입니다.
m_pActiveWnd	응용 프로그램의 최상위 프레임 윈도우에 대한 포인터입니다. SDI 구조에서 이 값은 CMainFrame 클래스 객체의 포인터입니다.
m_pszAppName	응용 프로그램의 제목에 해당하는 문자열의 포인터입니다. CreateDemo 예제의 경우 이 값이 CreateDemo였습니다. 문자열의 좀더 정확한 정보는 문자열 테이블(String Table)에 들어 있는 AFX_IDS_APP_TITLE에 해당하는 값입니다.
m_pszExeName	빌드한 실행 파일에서 확장자(.exe)를 제외한 파일명입니다.

가상 함수	기능
InitInatance()	응용 프로그램 초기화시점에 호출됨. 처음에 재정의 되어 있음.
ExitInstance()	응용 프로그램 종료직전에 호출됨. (사용한 자원 정리 등)
Run()	Message loop를 포함하고 있음. 이 함수에서 return하면 프로그램 종료.

CFrameWnd

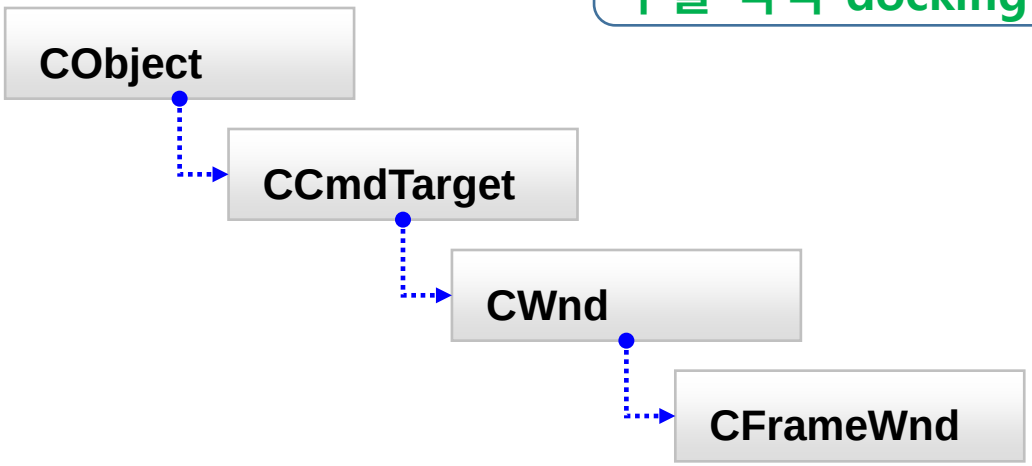
부모 윈도우가 NULL인(바탕화면이 부모) 최상위 윈도우.
다양한 형태의 자식 윈도우 가짐. (주로 컨트롤 윈도우 다룸)

OnCreate() 함수에서 아래 부분 수정해 보기.

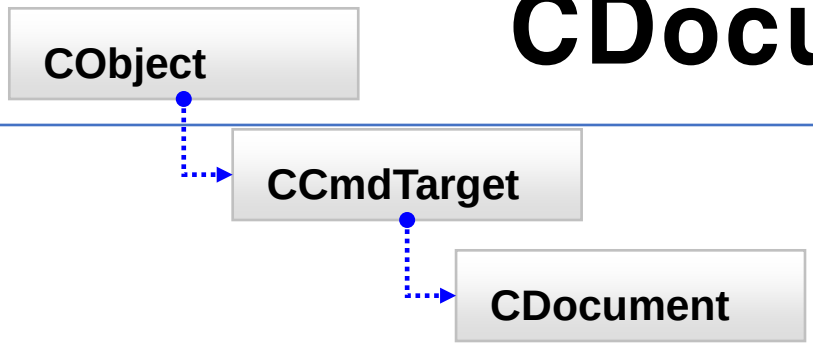
```
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
```

"선언으로 이동" 하여 다른 값으로 바꿔보기.

Toolbar 윈도우와 mainframe 윈도우를 각각 docking 시켜보기.



CDocument



다양한 가상함수로써 자료처리를 도와 줌.
(예) 프로그램 시작되면 OnNewDocument() 1회 호출 -> 빈 파일 생김.
[실습] 위 함수안에서 AfxMessageBox(_T("OnNewDocument"), MB_OK); 호출해 보기.

[편집기 프로젝트 만들기]
이전과 같이 만들되, 프로젝트명 : "HelloSdi", "유니코드 사용 안함", view의 base class를 "CEditView" 로.

-"클래스뷰" -> C...Document에서 R-click하여 "클래스마법사" -> "가상함수"
: OnSaveDocument(), OnOpenDocument(), OnCloseDocument() 등을 재정의하면 경우에 맞게 호출 됨.
-SetModifiedFlag() : 문서가 변경되었음을 설정/해제.
(실험) OnNewDocument()에서 SetModifiedFlag(TRUE); 하고 문서 닫아보기.

Message Map

- Win32 API WindowProc()함수의 switch case 문을 대체.
- 특정 메시지가 발생했을 때 어떤 함수를 호출해야 하는지 명시하는 매크로의 집합체.

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_CLOSE()
END_MESSAGE_MAP()
```

- [방법1] 3 곳을 수정하여 message handler 등록.
  - 헤더 파일(.h)에 afx_msg void OnClose();
  - 소스 파일(.cpp)에 void CMainFrame::OnClose() { … }
  - 메시지 맵에 ON_WM_CLOSE()
- [방법2] 클래스 마법사를 통해 message handler 등록.
  - “클래스뷰” -> CMainFrame에서 R-click하여 “클래스마법사” -> “메세지” -> “WM_CLOSE” 선택하고 “처리기 추가” -> 코드 편집”
- [방법3] 클래스 속성을 통해 message handler 등록.

MFC 응용프로그램
(MouseMFC)

(2) 좌표값을 보관할 멤버 변수 선언 (MouseMFCView.h)

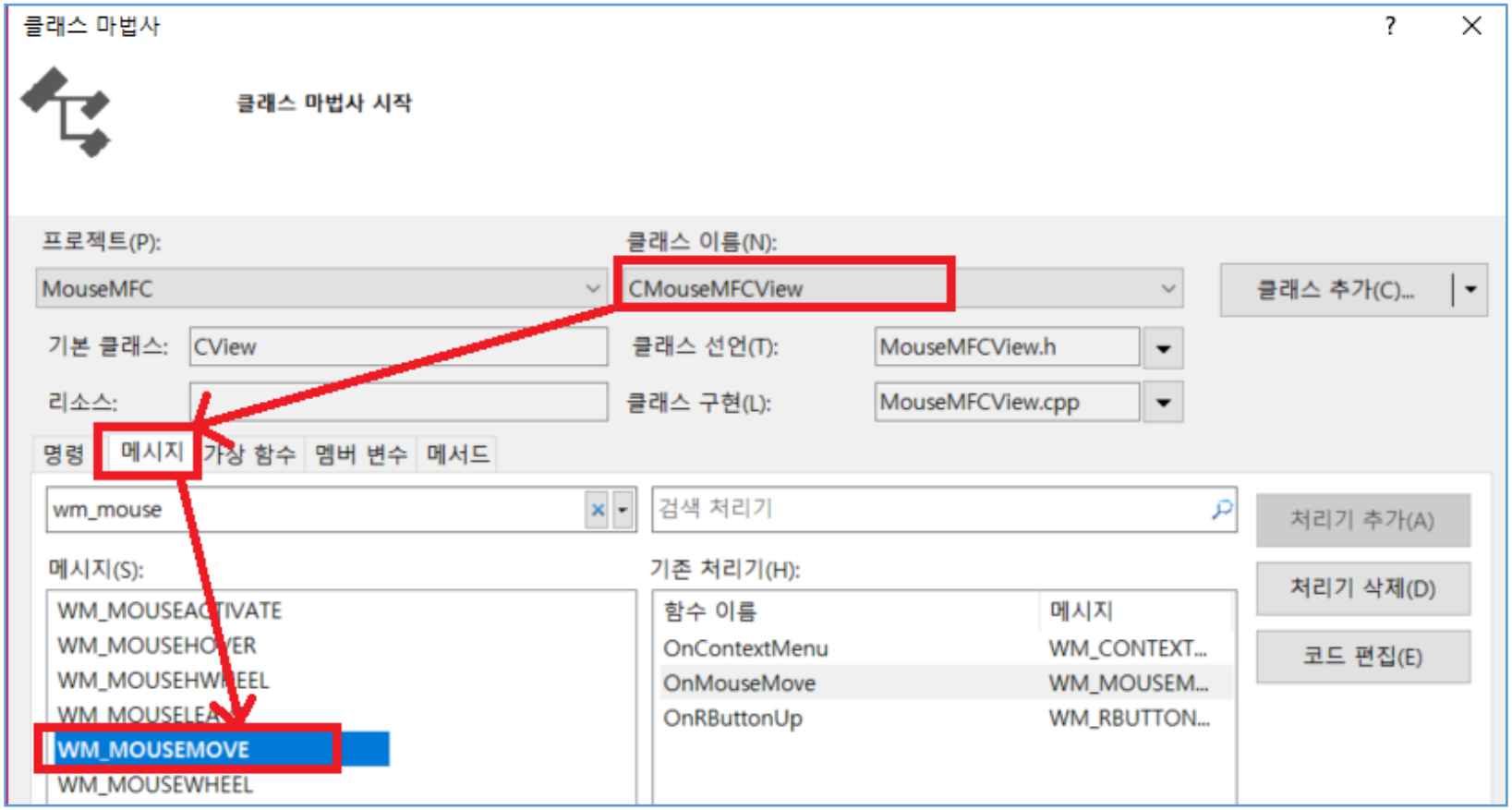
```
    // 작업입니다.
    public:
    CPoint m_Pos;
```

(3) 좌표값을 display (MouseMFCView.cpp)

```
void CMouseMFCView::OnDraw(CDC* /*pDC*/)
{
    CMouseMFCDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
    return;

    CClientDC dc(this);
    CString strPoint;
    strPoint.Format(_T("마우스 좌표 (%4d, %4d)"), m_Pos.x, m_Pos.y);
    dc.TextOutW(0, 0, strPoint);
}
```

MFC 응용프로그램
(MouseMFC)

(4) 클래스 마법사에서 View의 WM_MOUSEMOVE 메시지 핸들러 함수 추가.




```
void CMouseMFCView::OnMouseMove(UINT nFlags, CPoint point) {
    m_Pos = point;
    Invalidate();
    CView::OnMouseMove(nFlags, point);
}
```