

객체 지향 프로그래밍 응용

Chap 2. 프로그램의 뼈대

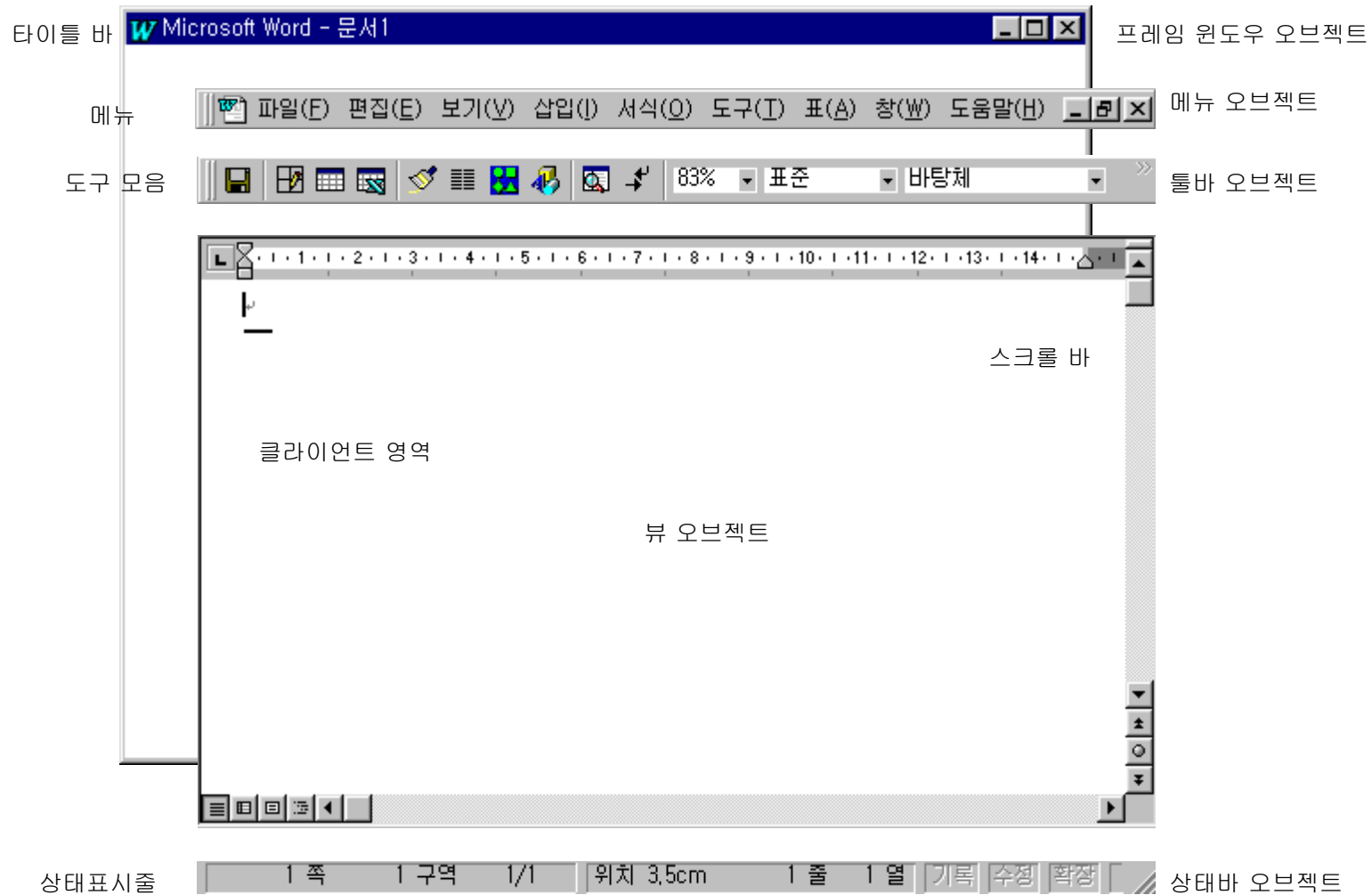


2013.09.23.

오 병 우

컴퓨터공학과
금오공과대학교

윈도우 프로그램 각 부분의 명칭



API 프로그래밍

Win32 Application

◆ A typical “Hello World!” application

```
int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine,
                    int nCmdShow)
{
    // TODO: Place code here.
    MSG msg;
    HACCEL hAccelTable;

    // Initialize global strings
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_WINAPI, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }

    hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_WINAPI);

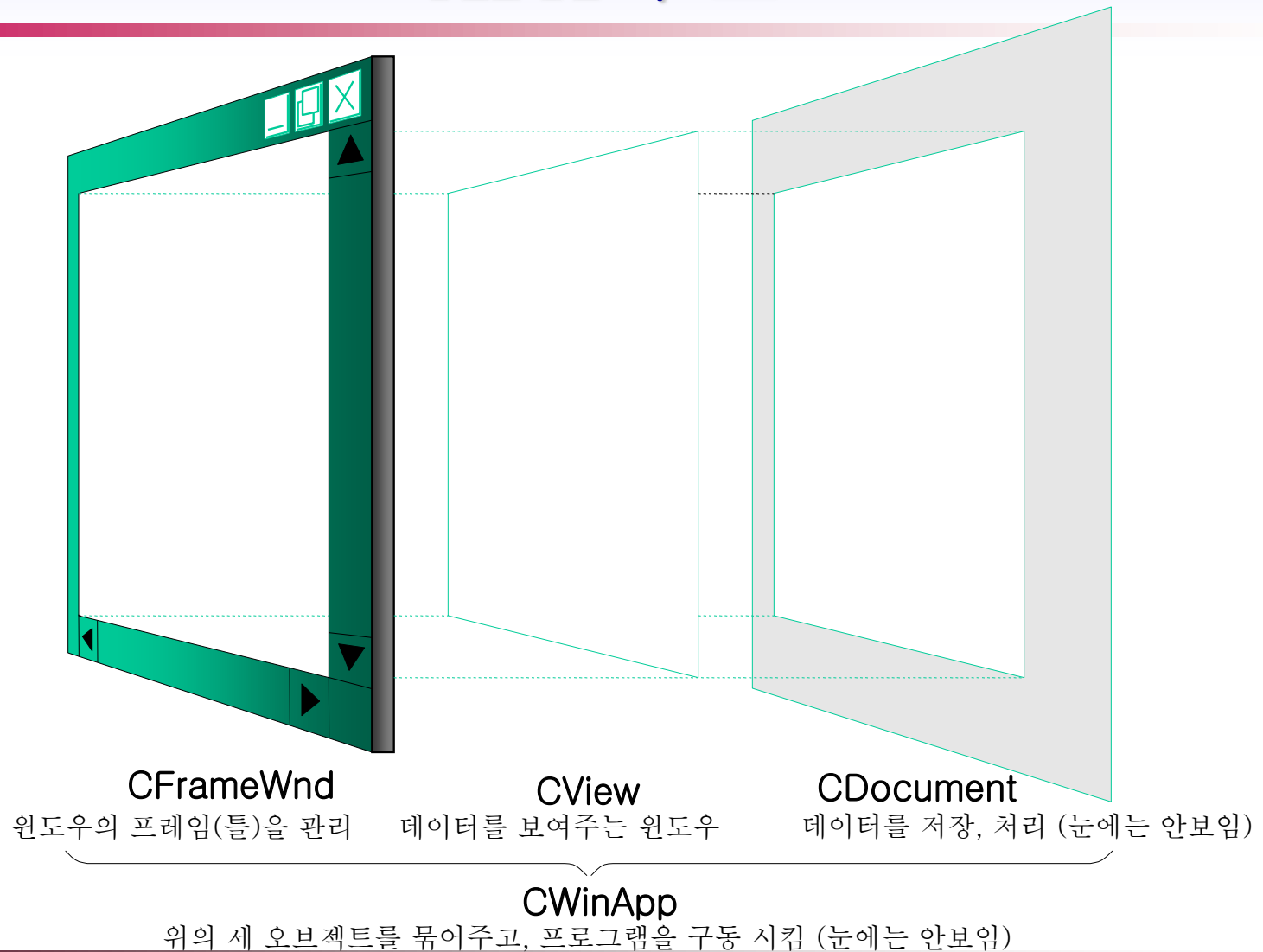
    // Main message loop:
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }

    return msg.wParam;
}
```

```
// PURPOSE: Processes messages for the main window.
//
// WM_COMMAND - process the application menu
// WM_PAINT - Paint the main window
// WM_DESTROY - post a quit message and return
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;
    TCHAR szHello[MAX_LOADSTRING];
    LoadString(hInst, IDS_HELLO, szHello, MAX_LOADSTRING);

    switch (message)
    {
        case WM_COMMAND:
            wmId = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Parse the menu selections:
            switch (wmId)
            {
                case IDM_ABOUT:
                    DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX, hWnd, (DLGPROC)About);
                    break;
                case IDM_EXIT:
                    DestroyWindow(hWnd);
                    break;
                default:
                    return DefWindowProc(hWnd, message, wParam, lParam);
            }
            break;
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            // TODO: Add any drawing code here...
            RECT rt;
            GetClientRect(hWnd, &rt);
            DrawText(hdc, szHello, strlen(szHello), &rt, DT_CENTER);
            EndPaint(hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

AFX 구조



윈도우 프로그램의 객체지향적 분할

● 프레임 윈도우와 뷰를 분리한 이유

- ◆ 일관된 사용자 인터페이스
- ◆ 프레임에 부여된 역할 즉, 창의 크기 조절, 확대, 축소 등을 사용자가 할 필요 없이 기본적으로 제공
- ◆ 뷰에서는 단지 실제 보여주어야 할 데이터 만을 사용자가 적절하게 계산하여 제공

● 도큐먼트와 뷰를 분리한 이유

- ◆ 데이터 저장 및 처리(CDocument)와 이를 보여주는(CView)을 분리 함으로써
 - 클래스의 역할을 분담 → 클래스를 단순화 시킴
 - 같은 데이터라도 보여주는 형태가 다를 수 있음
 - 즉, 하나의 Document에는 한 개 이상의 View가 존재

프로그램의 뼈대를 이루는 클래스 계층구조



CWinAPP 클래스

응용 프로그램 전체를 관리

- ◆ 초기화, 실행, 종료 담당
- ◆ 프레임 창 생성

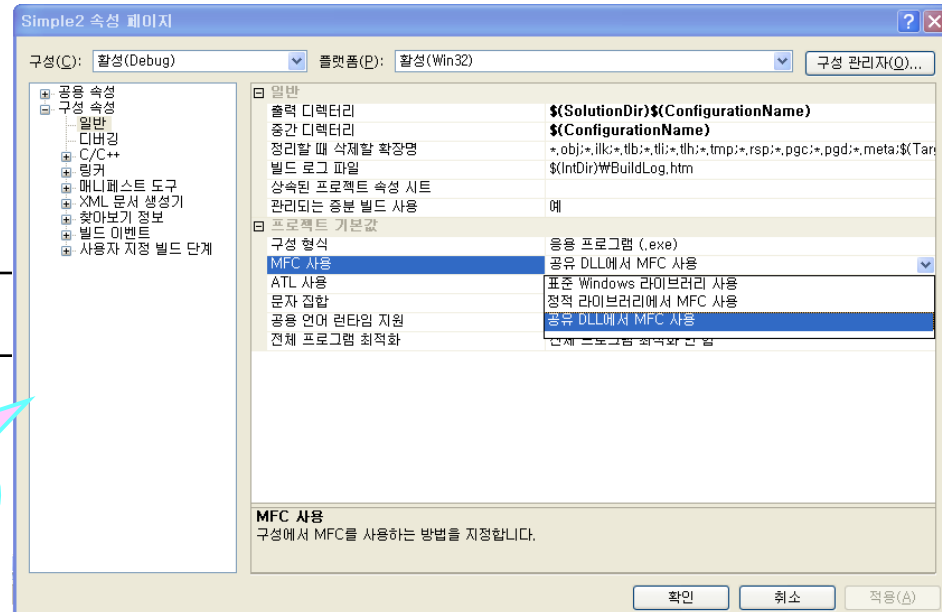
MFC 응용 프로그램

- ◆ CWinAPP로부터 파생된 클래스의 오브젝트를 오직 하나만 가져야 함
- ◆ 반드시 전역 변수여야 함

가장 간단한 MFC 프로그램

```
#include <afxwin.h>
CWinApp app;
```

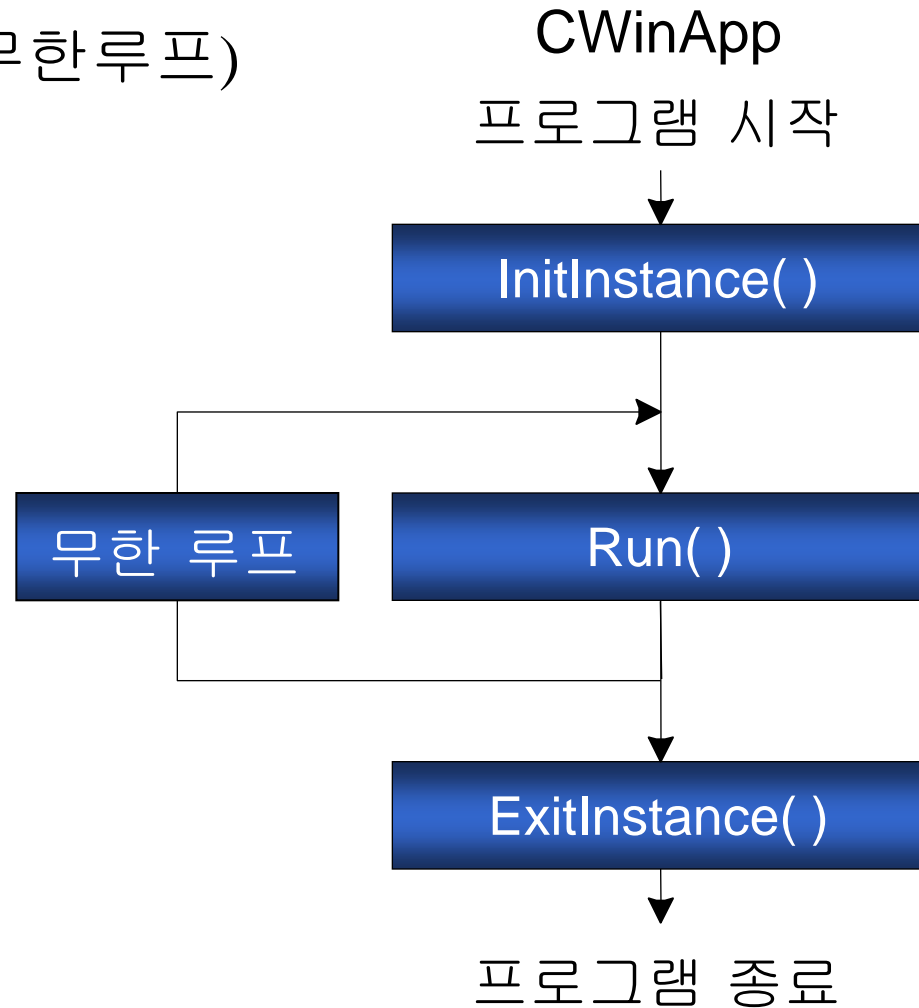
프로젝트 - (맨 밑에)
속성 - 구성
속성 - 일반



CWinAPP 클래스 (계속)

메시지 루프 (무한루프)

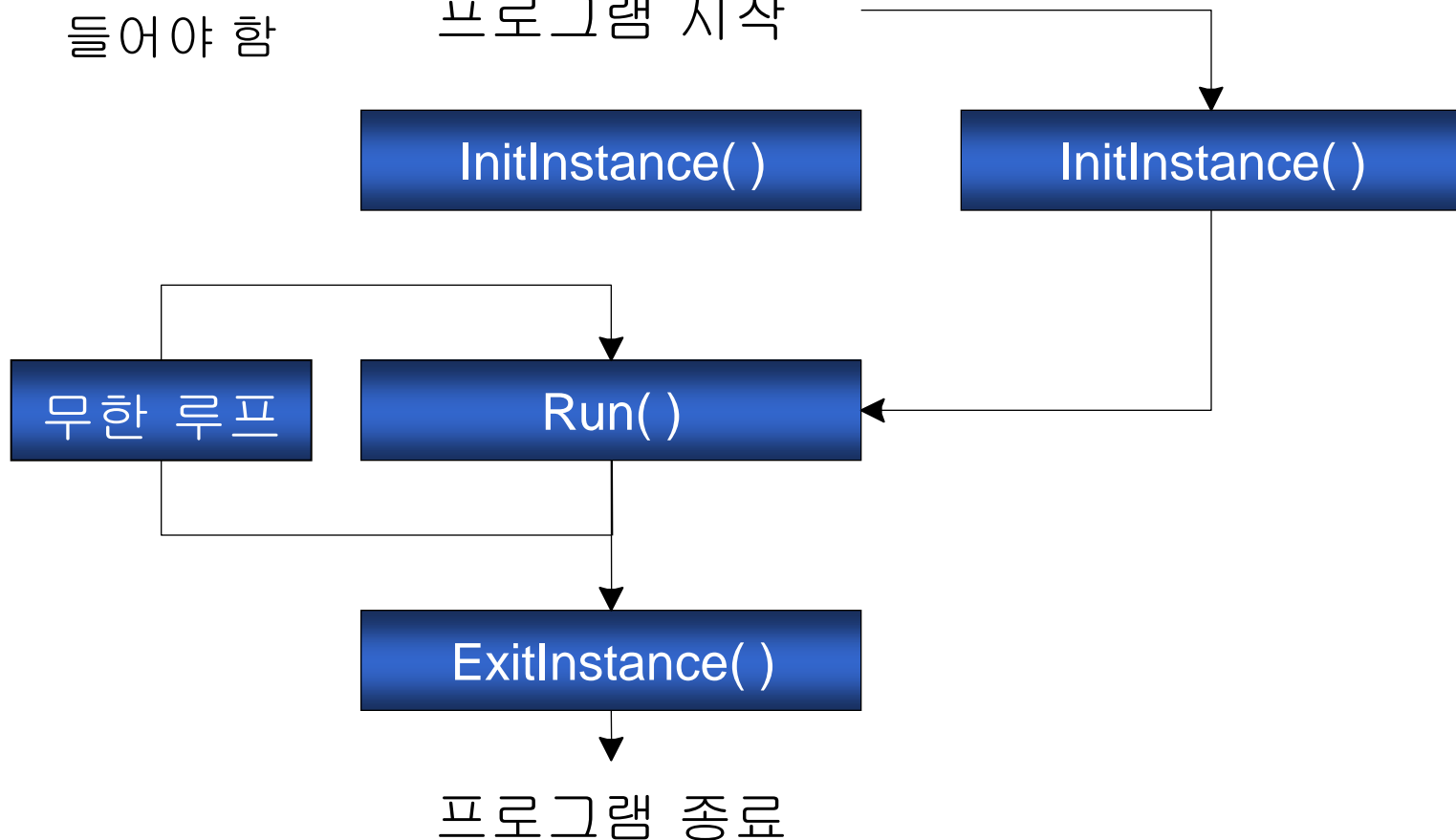
- ◆ InitInstance();
- ◆ Run();
- ◆ ExitInstance();



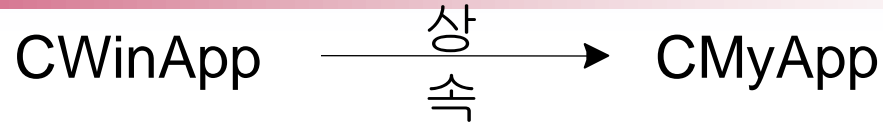
CWinApp 파생 클래스의 동작

● 멤버 함수 Override CWinApp $\xrightarrow[\text{상속}]{}$ CMyApp

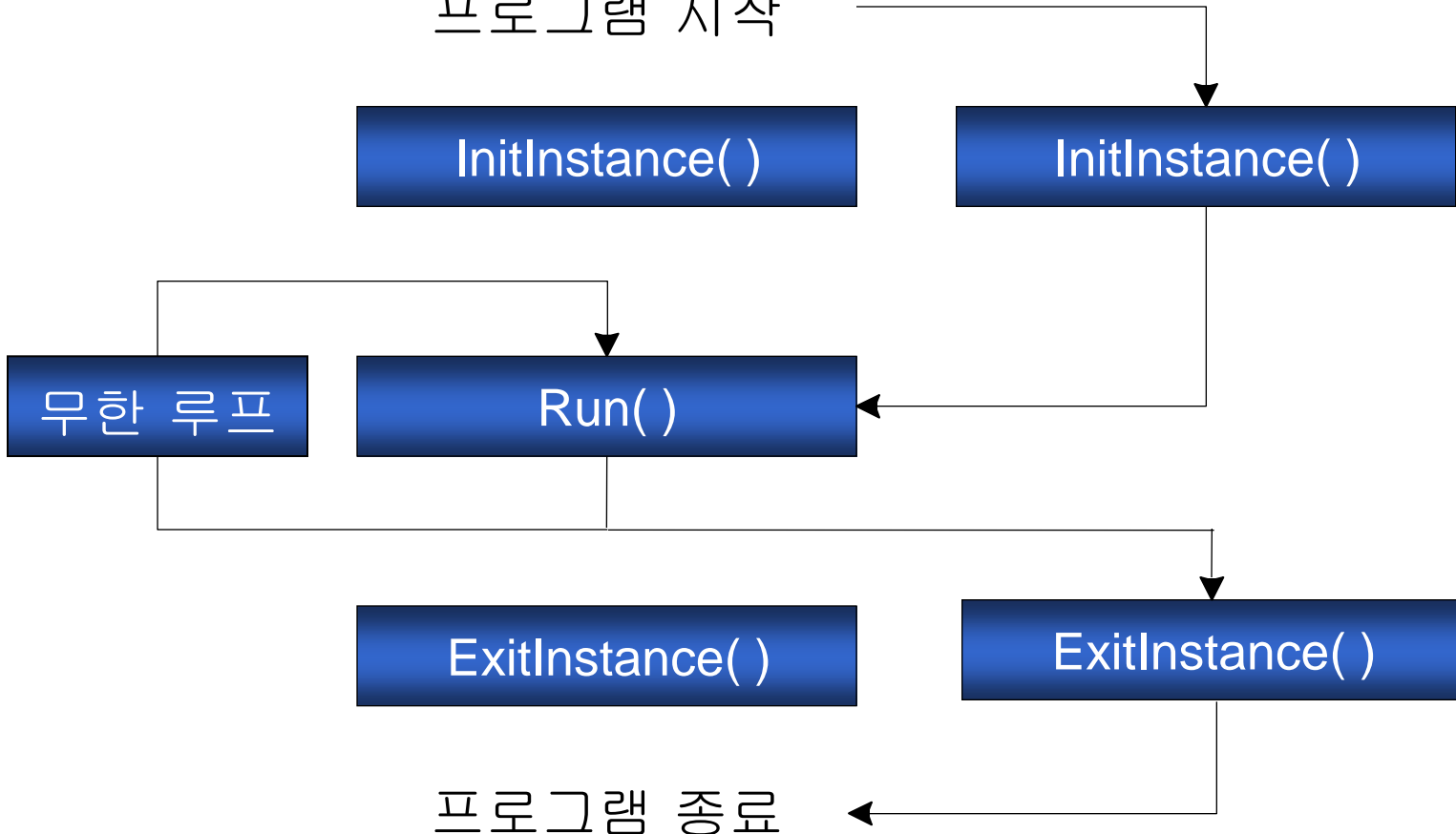
◆ 파생 클래스를 만
 들어야 함 프로그램 시작



CWinApp 파생 클래스의 동작



프로그램 시작



사용자 인터페이스 클래스

● 사용자에게 나타나는 영역을 갖는 클래스

◆ 윈도우 클래스

- CWnd: 다른 창 클래스들에 대한 기반 클래스로서 메시지들에 대한 기본 핸들러를 제공
- CFrameWnd: 프레임 윈도우 관리 및 자식 윈도우 관리
- CControlBar: 도구모음, 상태표시줄, 기타 컨트롤 표시줄 클래스의 기반 클래스
- CDialog: 다이얼로그 처리
- CView: Document/View 모형의 응용 프로그램에서 클라이언트 영역 관리 (Document 데이터 출력 및 사용자 입력 처리)

◆ 그래픽 클래스

- CDC: 디바이스 컨텍스트 클래스들에 대한 기반 클래스로서 그래픽 기능 제공
- CGdiObject: GDI (Graphic Device Interface) 오브젝트(펜, 브러시, 폰트 등)의 기반 클래스

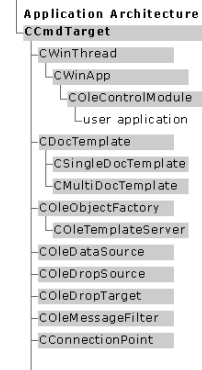
◆ 메뉴 클래스

- CMenu: 메뉴 관리

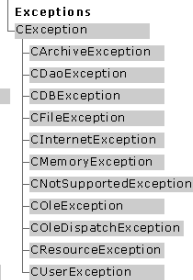
MFC Classes

Microsoft Foundation Class Library Version 6.0

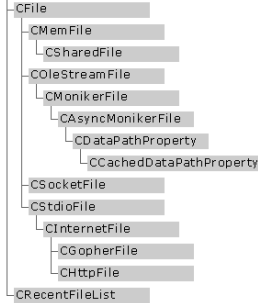
Object



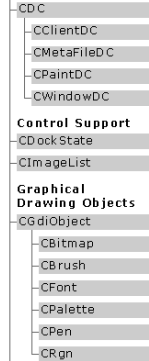
user objects



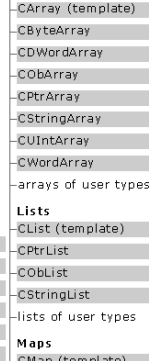
File Services



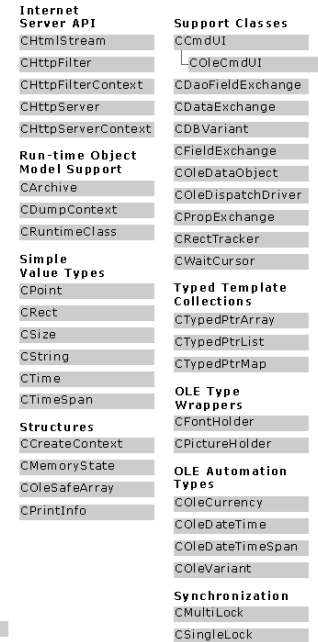
Graphical Drawing



Arrays



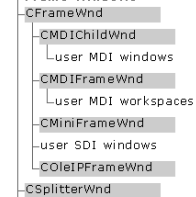
Classes Not Derived from Object



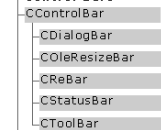
Window Support

CWnd

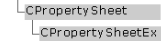
Frame Windows



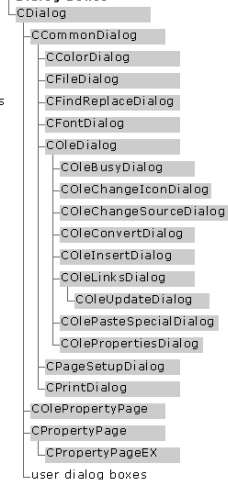
Control Bars



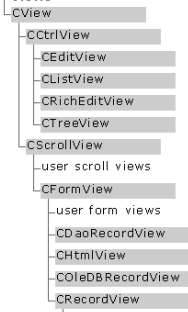
Property Sheets



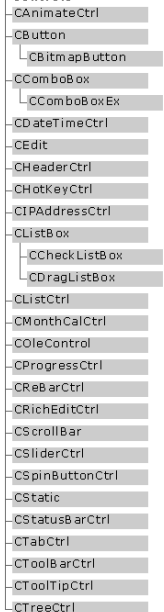
Dialog Boxes



Views



Controls



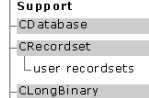
Menus



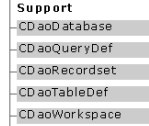
Command Line



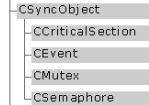
ODBC Database Support



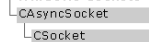
DAO Database Support



Synchronization



Windows Sockets



MFC 배대 이해 실습

```
#include <afxwin.h>
CWinApp app;
```

1. F10 눌러서 Debug

```
extern "C" int WINAPI
_tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPTSTR lpCmdLine, int nCmdShow)
{
    // call shared/exported WinMain
    return AfxWinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow);
}
```

MFC가 제공하는 Entry Function은 `_tWinMain()`

`AfxWinMain()`

`CWinApp`의 오브젝트 얻음

Document/view 초기화 (주로 재정적 안함)

인스턴스 초기화를 위해 주로 재정적

2. F11 두번 눌러서 `AfxWinMain()` 함수 살펴볼 것

`AfxWinMain()` 호출

3. Find Source 대화상자가 나오면 Cancel 클릭, ~F5누르고, Debug 탭 살펴 볼 것

```
int AFXAPI AfxWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPTSTR lpCmdLine, int nCmdShow)
{
    ASSERT(hPrevInstance == NULL);

    int nReturnCode = -1;
    CWinThread* pThread = AfxGetThread();
    CWinApp* pApp = AfxGetApp();

    // AFX internal initialization
    if (!AfxWinInit(hInstance, hPrevInstance, lpCmdLine, nCmdShow))
        goto InitFailure;

    // App global initializations (resources, etc.)
    if (pApp != NULL && !pApp->InitInstance())
        goto InitFailure;

    // Perform specific initialization
    if (!pThread->InitInstance())
    {
        if (pThread->m_pMainWnd != NULL)
        {
            TRACE0("Warning: Destroying non-NULL m_pMainWnd\n");
            pThread->m_pMainWnd->DestroyWindow();
        }
        nReturnCode = pThread->ExitInstance();
        goto InitFailure;
    }
    nReturnCode = pThread->Run();
}

InitFailure:
#ifdef _DEBUG
// Check for missing AfxLockTimeOut
if (AfxGetModuleThreadState(0) == 0)
{
    TRACE1("Warning: Temporarily disabled AfxLockTimeOut\n");
    AfxGetModuleThreadState(0) = 1;
}
#endif
```

`CWinApp::Run()`에서 출력

메시지를 처리하다가 `WM_QUIT`을 만나면 `ExitInstance()` 호출

```
Loaded 'C:\WINDOWS\system32\mfck21oc.dll', no matching symbolic information found.
Warning: m_pMainWnd is NULL in CWinApp::Run - quitting application.
Loaded 'C:\WINDOWS\system32\WMSOFT.dll', no matching symbolic information found.
Loaded 'C:\WINDOWS\system32\msvcrt.dll', no matching symbolic information found.
The program 'C:\연구\강의\원도우\실습\ex2\Debug\ex2.exe' has exited with code 0.
```

MFC 배대 이해 실습 (계속)

```

#include <afxwin.h>

class MyApp : public CWinApp {
public:
    virtual BOOL InitInstance();
};

BOOL MyApp::InitInstance()
{
    AfxMessageBox("파생 클래스의 InitInstance() Overriding");

    return TRUE;
}

MyApp app;
    
```

CWinApp 클래스의 기반 클래스는 CWinThread 클래스

→ CWinThread::InitInstance()에서 Overriding된 함수가 실행됨
(Debugging하여 확인)

MFC 배대 이해 실습 (계속)

```

#include <afxwin.h>

class MyApp : public CWinApp {
public:
    virtual BOOL InitInstance();
};

class CMainWnd : public CFrameWnd {
public:
    CMainWnd();
};

BOOL MyApp::InitInstance()
{
    m_pMainWnd = new CMainWnd;
    m_pMainWnd->ShowWindow(m_nCmdShow);
    m_pMainWnd->UpdateWindow(); // WM_PAINT 메시지 발생

    return TRUE;
}

CMainWnd::CMainWnd()
{
    Create(NULL, "윈도우시스템프로그래밍");
}

MyApp app;
    
```

MS-Windows
운영체제 관련
기본적인 동작
(프레임)

MFC 배대 이해 실습 (계속)

<전략>

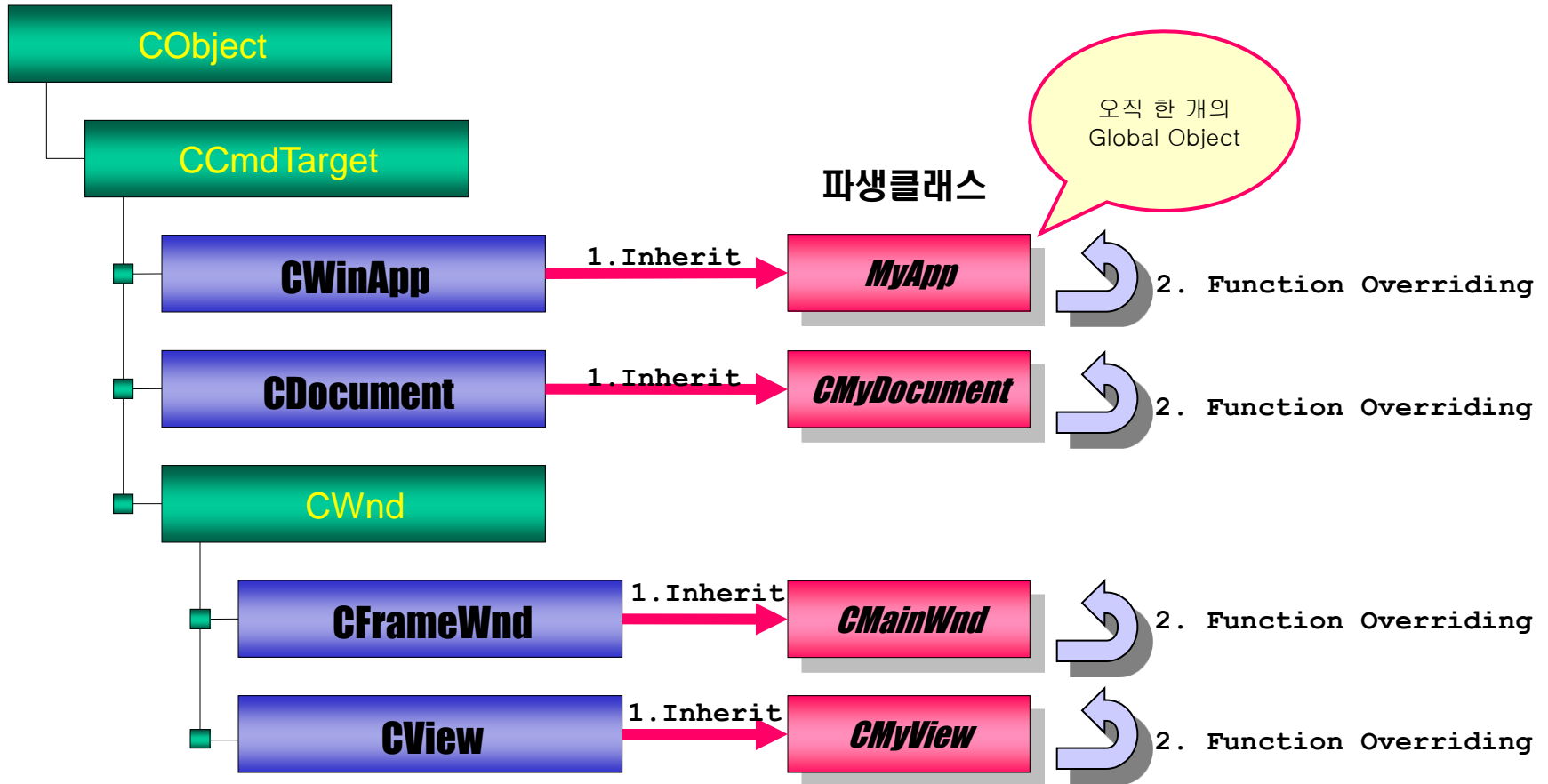
```

CMainWnd::CMainWnd()
{
    CString wc = AfxRegisterWndClass(0,
        AfxGetApp()->LoadStandardCursor(IDC_WAIT),
        (HBRUSH)::GetStockObject(LTGRAY_BRUSH),
        AfxGetApp()->LoadStandardIcon(IDI_QUESTION));

    Create(wc,
        "객체지향프로그래밍응용",
        WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX | WS_HSCROLL | WS_VSCROLL,
        CRect(0, 0, 200, 200));
}

MyApp app;
    
```


MFC 프로그래밍



Message

- Event 정보를 프로그램에서 다룰 수 있도록 구조화

- MSG 구조체로 정의

- ◆ 윈도우 핸들, 메시지 식별번호, 추가정보, 시각, 커서 위치 포함

- 메시지 종류

- ◆ 시스템 정의 메시지 (0x0000~0x7fff)

- 운영체제가 이미 식별 번호를 예약

- ◆ 사용자 정의 메시지 (0x8000~0xffff)

- 사용자가 추가로 식별 번호 할당

- 메시지 식별 매크로 (winuser.h 등)

- ◆ Prefix

- 버튼 컨트롤(BN_), 에디트 컨트롤 (EN_), 리스트 박스 컨트롤 (LBN_) 등이 있으나 일반 윈도우 메시지(WM_)가 가장 많이 사용됨

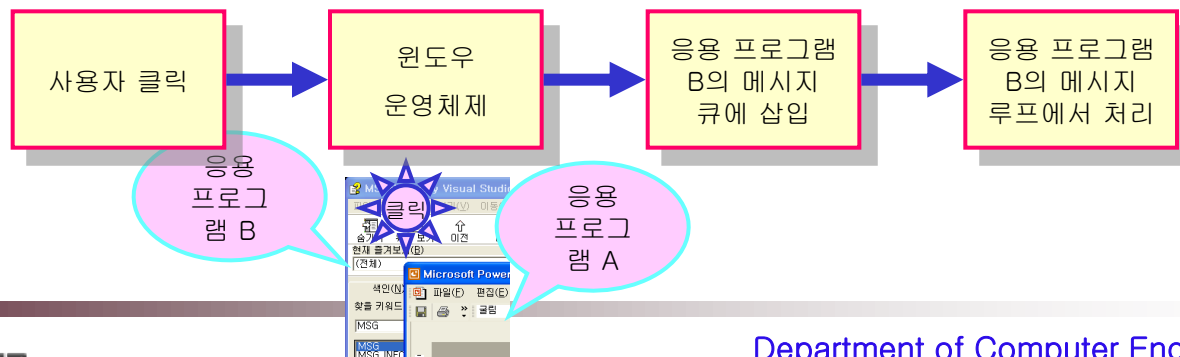
- ◆ WM_CREATE (0x0001) : 창이 생성될 때

- ◆ WM_LBUTTONDOWN: 왼쪽 마우스 버튼 누를 때

```
typedef struct tagMSG {
    HWND    hwnd;
    UINT    message;
    WPARAM  wParam;
    LPARAM  lParam;
    DWORD   time;
    POINT   pt;
} MSG;
```

Message 처리

- 발생 즉시 처리되는 메시지
 - ◆ 대부분의 윈도우 관리 메시지
 - WM_CREATE, WM_DESTROY 등
- 메시지 큐 사용
 - ◆ 우선 순위가 낮은 메시지
 - ◆ FIFO (First in, First Out)
 - ◆ Message Loop
 - CWinApp::Run() 함수는 CWinThread::Run() 함수 호출
 - WM_QUIT을 만날 때까지 GetMessage(), TranslateMessage(), DispatchMessage() 함수를 반복 수행



Message Map을 이해하기 위한 함수 포인터

● CALC_ENTRY

- ◆ 식별번호 및 함수 포인터로 구성
- ◆ 마지막에 NULL Entry 삽입
- ◆ 많은 키보드 중에서 a, s, m 만 사용

```
int add()
{
    return 5 + 3;
}
int sub()
{
    return 5 - 3;
}
int mul()
{
    return 5 * 3;
}

struct CALC_ENTRY
{
    char id;
    int (*funcptr)();
};

CALC_ENTRY _functionMap[] = {
    { 'a', &add },
    { 's', &sub },
    { 'm', &mul },
    { 0, 0 } // NULL Entry
};
```

```
void CFuncPtrDlg::OnOK()
{
    // TODO: Add extra validation here

    UpdateData(TRUE);
    if (!m_Input.Compare(""))
        return;

    char input = m_Input[0];

    int i = 0;
    while (_functionMap[i].id != 0) {
        if (_functionMap[i].id == input) {
            m_Output.Format("%d", _functionMap[i].funcptr());
            break;
        }
        else
            i++;
    }

    if (_functionMap[i].id == 0)
        m_Output = "Not Found";

    UpdateData(FALSE);

    // CDialog::OnOK();
}
```

Message Map 사용

MFC의 AppWizard가 모두 처리해 줌

1. 메시지 맵을 헤더 파일(.h)의 클래스에 선언

```
class C[프로젝트명]View : public CView
{
...
// Generated message map functions
protected:
    //{{AFX_MSG(C[프로젝트명]View)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

```
#define afx_msg // intentional placeholder
```

메시지
핸들러임을
explicitly
표시 (내용은
없음)

2. 소스 파일(.cpp)의 윗 부분에 메시지 맵 정의

```
BEGIN_MESSAGE_MAP(C[프로젝트명]View, CView)
    //{{AFX_MSG_MAP(C[프로젝트명]View)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

3. 메시지 핸들러 구현

```
void C[프로젝트명]View::OnLButtonDown(UINT nFlags, CPoint point)
{
    // 코드 작성
    CView::OnLButtonDown(nFlags, point);
}
```

Message Handler를 잘못 넣어 삭제할 때

- 앞 페이지의 3군데를 삭제 (또는 주석처리) 해야 함

1. 헤더 파일(.h)의 클래스에 선언된 부분 삭제

```
class C[프로젝트명]View : public CView
{
...
// Generated message map functions
protected:
    //{{AFX_MSG(C[프로젝트명]View)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

WM_LBUTTONDOWN
을 잘못 넣었을 때

2. 소스 파일(.cpp)의 윗 부분에 메시지 맵 정의에서 삭제

```
BEGIN_MESSAGE_MAP(C[프로젝트명]View, CView)
    //{{AFX_MSG_MAP(C[프로젝트명]View)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

3. 메시지 핸들러 메소드 삭제

```
void C[프로젝트명]View::OnLButtonDown(UINT nFlags, CPoint point)
{
    // 코드 작성
    CView::OnLButtonDown(nFlags, point);
}
```

Message Map 작동 원리

Message Map

- ◆ 메시지 식별 번호, Function Pointer 등을 포함한 (Array 형태의) Mapping Table

1. Message Map 선언 내부

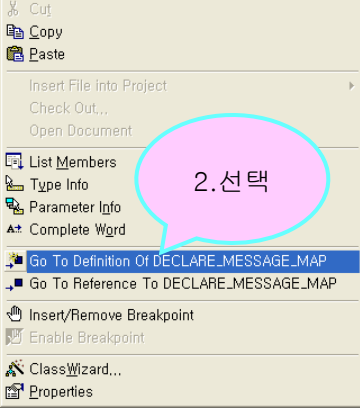
- ◆ `_messageEntries`: 각 메시지 종류에 따라서 메시지 식별 번호, Function Pointer 등을 가지는 엔트리의 Array
- ◆ `messageMap`: 기반 클래스의 `messageMap`에 대한 Pointer 및 현재 클래스의 `_messageEntries`의 Array로 구성
 - 현재 클래스의 메시지 핸들러 검색에서 찾지 못했다면 기반 클래스의 메시지 핸들러 검색

DECLARE MESSAGE MAP ()

private:
 static const AFX_MSGMAP_ENTRY **_messageEntries** [];
 protected:
 static AFX_DATA const AFX_MSGMAP **messageMap**;
 virtual const AFX_MSGMAP* **GetMessageMap**() const;

struct AFX_MSGMAP
 {
 const AFX_MSGMAP* pBaseMap;
 const AFX_MSGMAP_ENTRY* lpEntries;
 };

struct AFX_MSGMAP_ENTRY
 {
 UINT nMessage; // windows message
 UINT nCode; // control code or WM_NOTIFY code
 UINT nID; // control ID (or 0 for windows messages)
 UINT nLastID; // used for entries specifying a range of control id's
 UINT nSig; // signature type (action) or pointer to message #
 AFX_PMSG pfn; // routine to call (or special value)
 };



Message Map 작동 원리 (계속)

2. Message Map 정의 내부

◆ BEGIN_MESSAGE_MAP()

- GetMessageMap 함수 구현
- Static Member Variable인 messageMap에 value 할당
- Static Member Variable인 _messageEntries 변수 정의 및 초기화를 위한 처음 부분 (“= {”)
전개

BEGIN_MESSAGE_MAP(C[프로젝트명]View, CView)

```
#define BEGIN_MESSAGE_MAP(theClass, baseClass) \
const AFX_MSGMAP* theClass::GetMessageMap() const \
{ return &theClass::messageMap; } \
AFX_COMDAT AFX_DATADEF const AFX_MSGMAP theClass::messageMap = \
{ &baseClass::messageMap, &theClass::_messageEntries[0] }; \
AFX_COMDAT const AFX_MSGMAP_ENTRY theClass::_messageEntries[] = \
{ \
```

◆ 메시지 핸들러 등록 (ON_WM_LBUTTONDOWN())

- 메시지 식별 번호, 메시지 핸들러 함수 포인터 등의 정보를 가지고 Array에 메시지 처리
를 위한 엔트리 삽입

ON_WM_LBUTTONDOWN()

```
#define ON_WM_LBUTTONDOWN() \
{ WM_LBUTTONDOWN, 0, 0, 0, AfxSig_vwp, \
(AFX_PMSG)(AFX_PMSGW)(void (AFX_MSG_CALL CWnd::*)(UINT,CPoint)) \
&OnLButtonDown },
```

◆ END_MESSAGE_MAP()

- Array의 끝을 식별하기 위한 NULL 엔트리 삽입

END_MESSAGE_MAP()

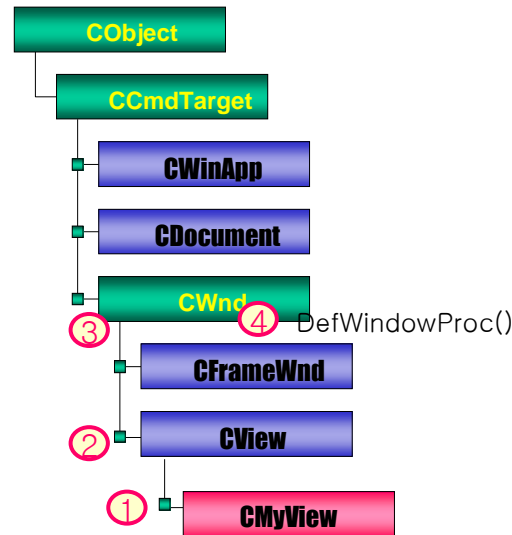
```
#define END_MESSAGE_MAP() \
{0, 0, 0, 0, AfxSig_end, (AFX_PMSG)0 } \
}; \
```


Message Map 처리 과정

처리 순서

1. 현재 클래스의 메시지 핸들러 찾아 보고 있으면 핸들러 함수 호출
2. 없으면 기반 클래스의 메시지 핸들러 찾음
3. 없으면 기반 클래스의 기반 클래스의 메시지 핸들러 찾음 (반복)
4. 그래도 없으면 `CWnd::DefWindowProc()`을 호출하여 Default 메시지 처리

- e.g., `WM_MINIMIZE`



Message Map vs. 가상함수

- Message Map은 메시지 핸들러의 엔트리 수만큼만 메모리를 사용하므로 효율적인 방법
- 가상함수의 재정의의 아님
 - ◆ 가상함수는 동적 바인딩을 위하여 run-time시에 정보를 유지
 - ◆ 응용 프로그램에서 메시지 처리를 재정의 하는 경우가 일반적으로 적음
 - ◆ 메모리의 효율적 사용을 (낭비를 없애기) 위하여 Message Map 활용

Message Map



Virtual Function



시스템 정의 Message 종류

Window Message

- ◆ 윈도우에서 발생하는 메시지 (e.g., WM_LBUTTONDOWN)
- ◆ 메시지 핸들러 함수의 이름 및 원형이 미리 결정되어 있다.
 - e.g., `afx_msg void OnLButtonDown(UINT nFlags, CPoint point);`

Control Notification Message

- ◆ 컨트롤이 부모 윈도우에게 통지하는 메시지 (e.g., BN_CLICKED)
- ◆ 메시지 핸들러 함수의 이름은 프로그래머가 임의로 부여할 수 있지만 인자와 리턴 타입은 규칙 준수

Command Message

- ◆ 메뉴, 가속기, 도구모음 등의 사용자 인터페이스에서 발생시키는 메시지 (e.g., WM_COMMAND)