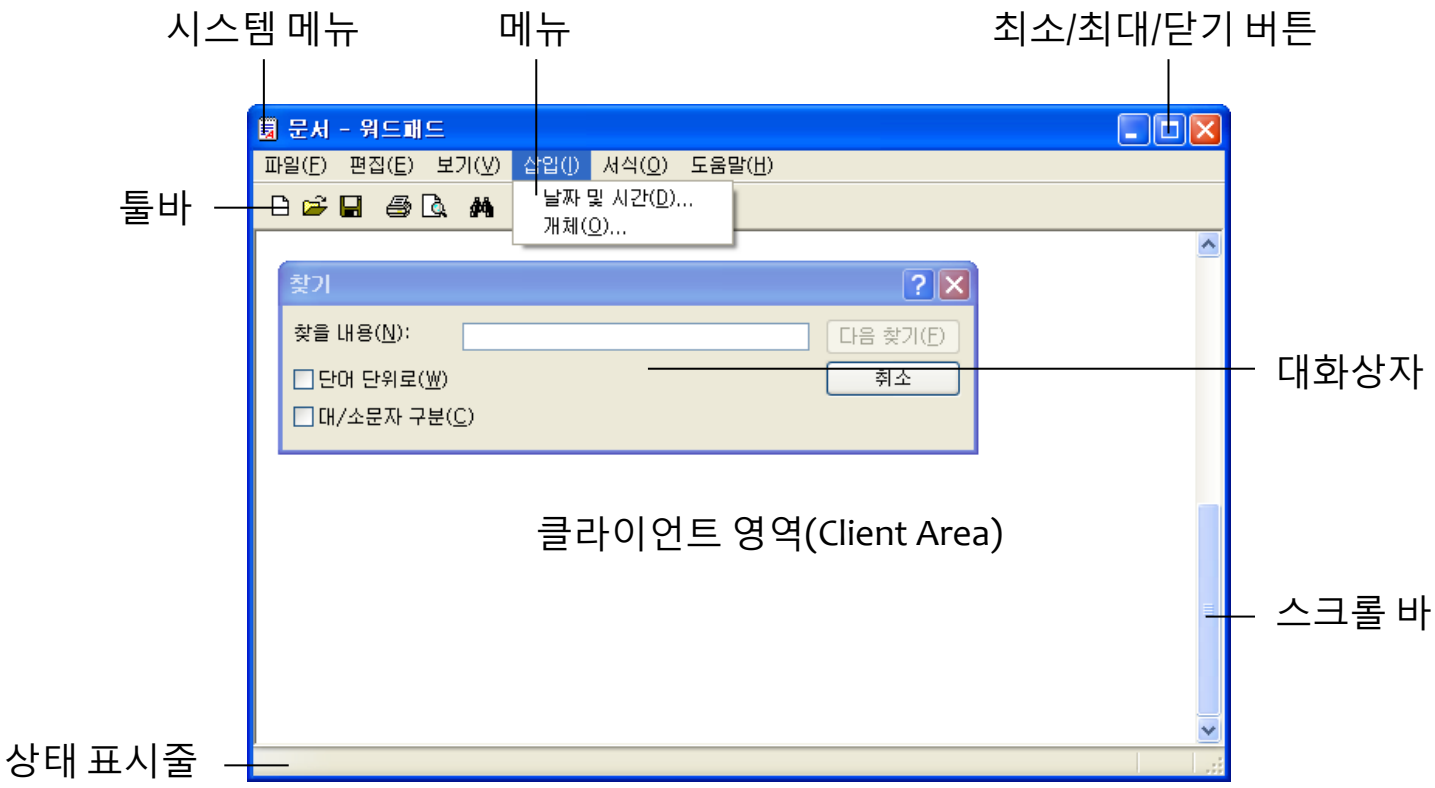
5장
MFC기반 다지기

김성영교수
금오공과대학교
컴퓨터공학부

들어가기 (1)

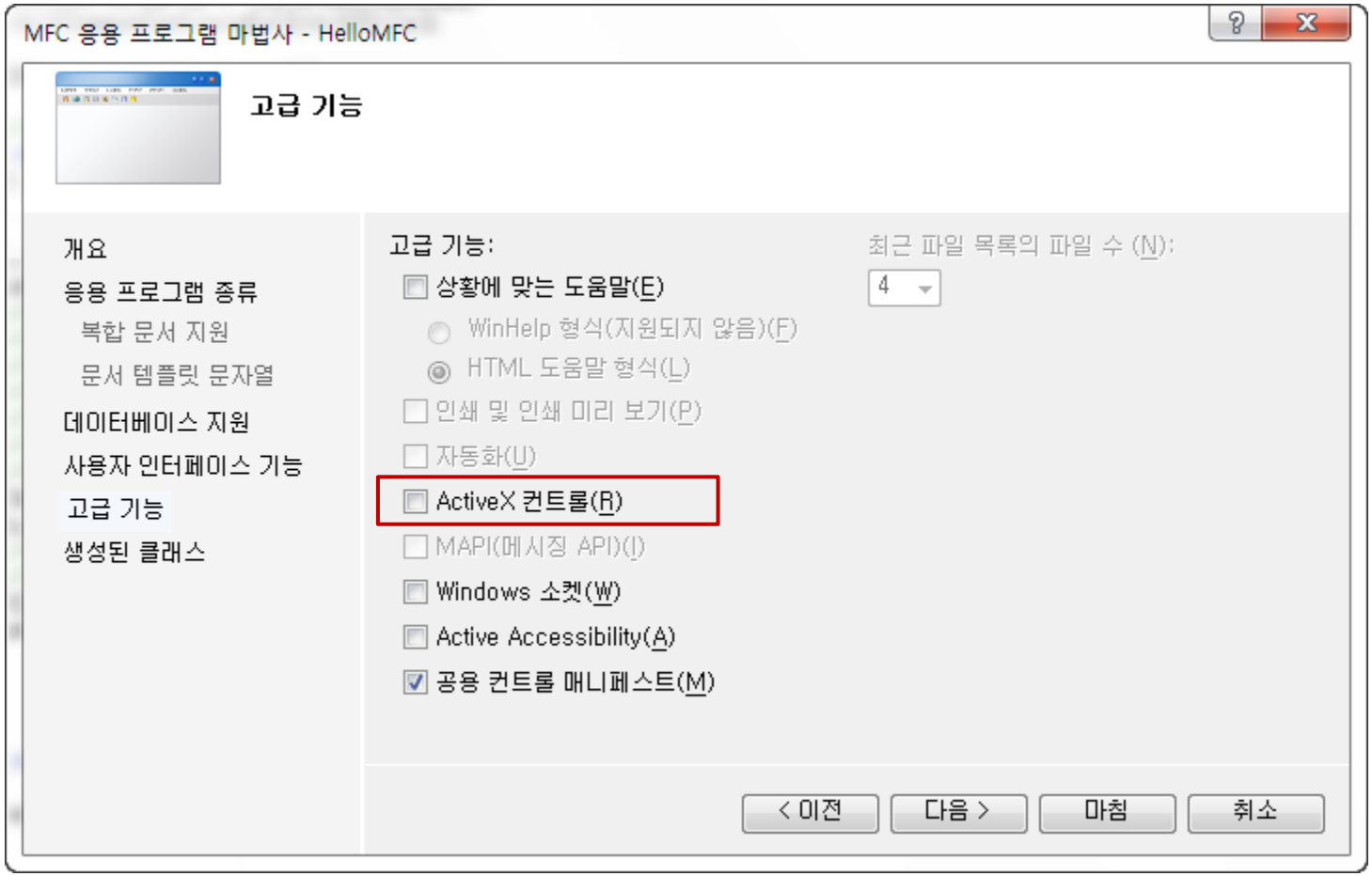
- pp.132 ~ 138
  “비주얼 스튜디오에서 MFC 어플리케이션 작성” 참조



들어가기 (2)


MFC 응용 프로그램 마법사 - HelloMFC
응용 프로그램 종류
개요
응용 프로그램 종류
복합 문서 지원
문서 템플릿 문자열
데이터베이스 지원
사용자 인터페이스 기능
고급 기능
생성된 클래스
응용 프로그램 종류:
단일 문서(S)
다중 문서(M)
대화 상자 기반(D)
[HTML] 대화 상자 사용(I)
다중 최상위 문서(T)
문서/뷰 아키텍처 지원(V)
리소스 언어(L):
한국어(대한민국)
유니코드 라이브러리 사용(N)
프로젝트 스타일:
Windows 탐색기(X)
MFC 표준(A)
MFC 사용:
공유 DLL에서 MFC 사용(U)
정적 라이브러리에서 MFC 사용(E)
< 이전
다음 >
마침
취소

들어가기 (3)


MFC 응용 프로그램 마법사 - HelloMFC
고급 기능
개요
응용 프로그램 종류
복합 문서 지원
문서 템플릿 문자열
데이터베이스 지원
사용자 인터페이스 기능
고급 기능
생성된 클래스
고급 기능:
상황에 맞는 도움말(E)
WinHelp 형식(지원되지 않음)(F)
HTML 도움말 형식(L)
인쇄 및 인쇄 미리 보기(P)
자동화(U)
ActiveX 컨트롤(R)
MAPI(메시징 API)(I)
Windows 소켓(W)
Active Accessibility(A)
공용 컨트롤 매니페스트(M)
최근 파일 목록의 파일 수 (N):
4
< 이전
다음 >
마침
취소

들어가기 (4)

- 솔루션 탐색기 확인
  - WinMain() 함수가 없다?
- MFC 프레임워크에 대한 이해가 필요함
  - 프로젝트를 구성하는 주요 클래스는 4개로 구성
  - 4개의 주요 클래스 및 그 부모 클래스는 무엇인가?

프로젝트에 생성된 클래스	부모 클래스
C___App	CWinApp
CMainFrame	CFrameWnd
C___Doc	CDocument
C___View	CView

MFC 프로그램 시작 (1)

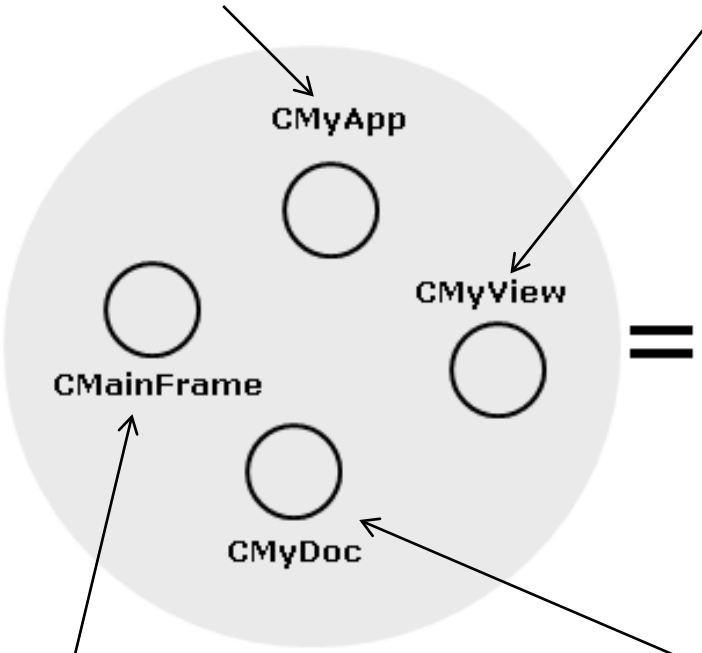
- 프로그래밍
  - 주어진 데이터를 내가 원하는 데이터로 만드는 과정 (함수 사용)

- 객체지향 프로그래밍에서는...
  - 객체 내부에 변수와 그 값을 변경하는 함수가 존재
  - 개념적으로는 객체만 생성하면 프로그래밍 완성
  - 관련 MFC클래스: CWinApp, CFrameWnd, CView, CDocument

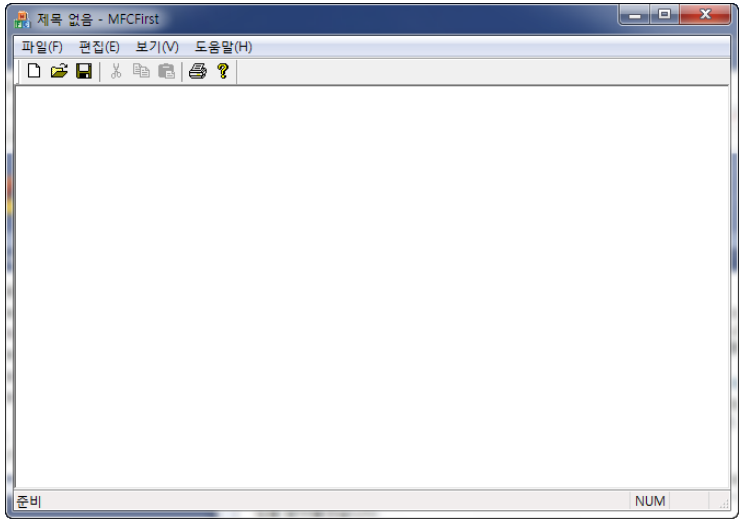
MFC 프로그램 시작 (2)

CWinApp: 프로그램 구동

CView: 클라이언트 윈도우 처리



=



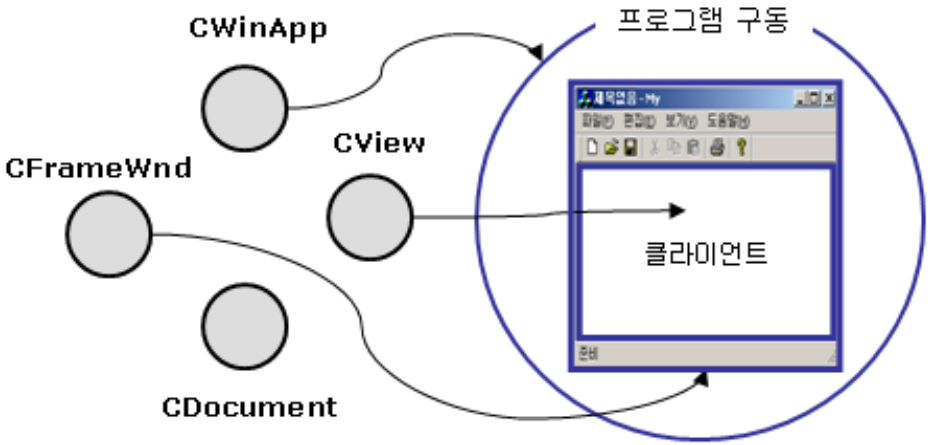
CFrameWnd: 프레임(틀) 윈도우 처리

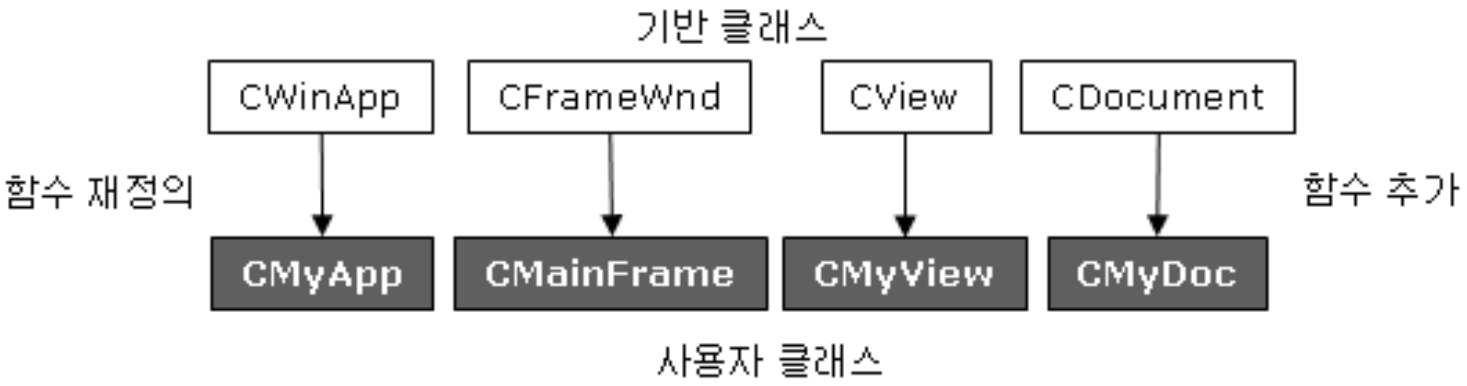
CDocument: 데이터 처리

```
C___App  theApp;  //___.h에 전역변수로 선언
```

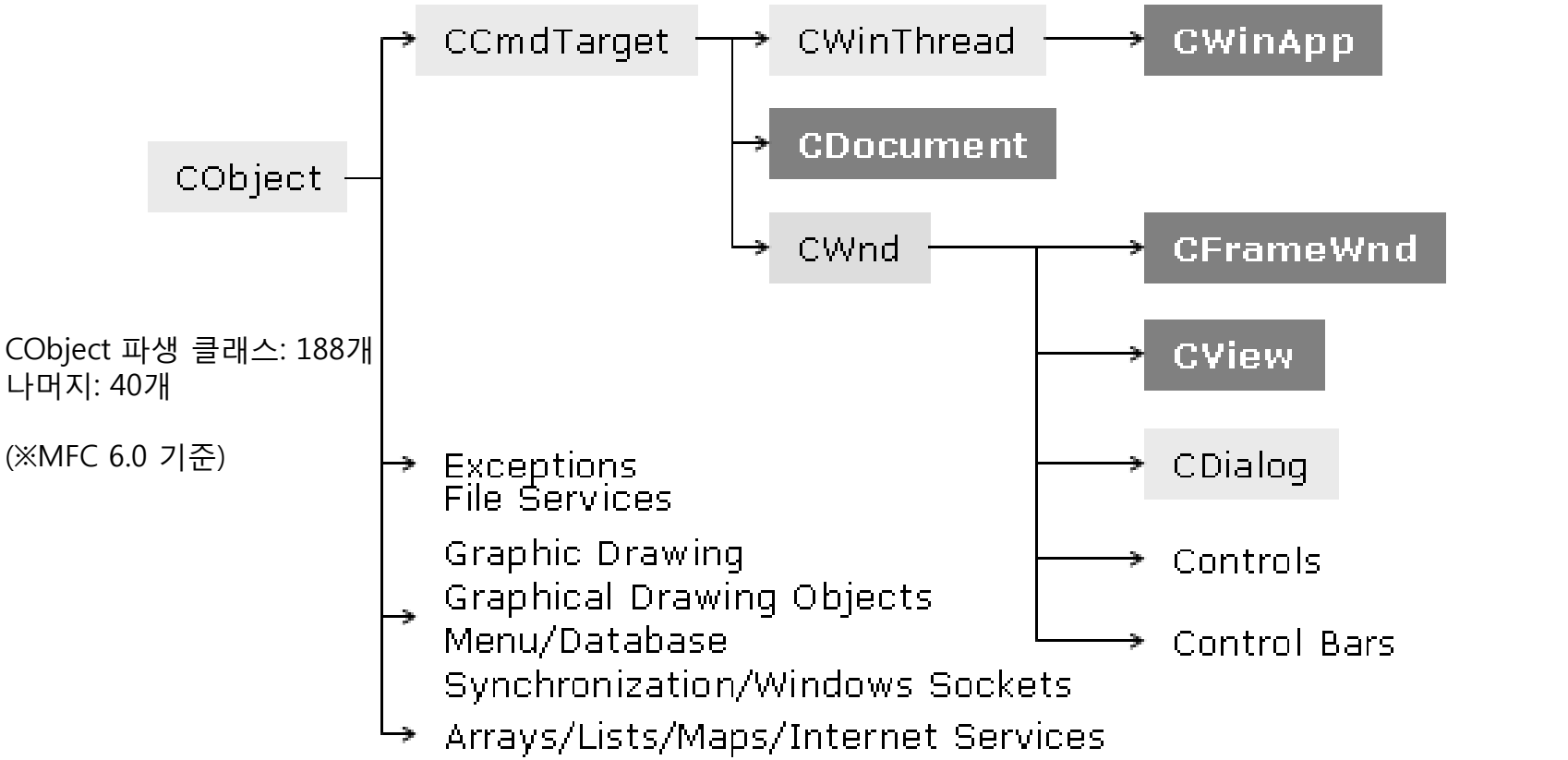
MFC 윈도우 프로그래밍이란?

- MFC 프레임워크에서 생성한 기반 클래스로부터 상속한 후 변수와 함수를 추가 혹은 재정의하여 사용자 클래스를 작성하는 것





기반 클래스



CObject 파생 클래스: 188개
나머지: 40개

(※MFC 6.0 기준)

CWinApp (애플리케이션 클래스) (1)

CObject → CCmdTarget → CWinThread → CWinApp

- 일부 클래스를 제외한 모든 클래스는 CObject 클래스로 시작
- CCmdTarget 클래스에서 WM_COMMAND 메시지 처리
  → CWinApp 클래스에서도 WM_COMMAND 메시지 처리 가능

CWinApp	
멤버변수	멤버함수

HINSTANCE m_hInstance
CWnd* m_pMainWnd
	void InitInstance(); //프로그램 초기화
	void Run(); //프로그램 수행
	void ExitInstance(); //프로그램 종료

CWinApp (애플리케이션 클래스) (2)

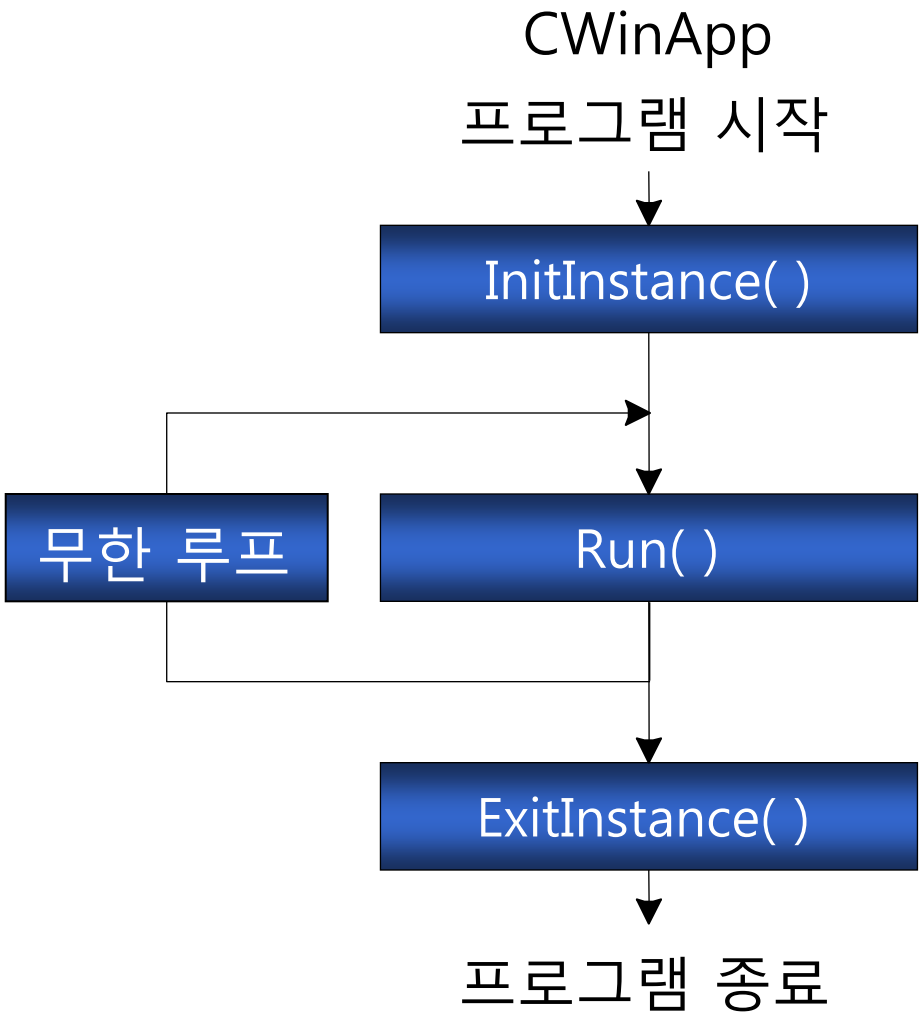
My.cpp

```
CMyApp theApp;

BOOL CMyApp::InitInstance()
{
        ...
}

CMainFrame::CMainFrame()
{
        ...
}

void CMainFrame::OnPaint()
{
        ...
}

void
CMainFrame::OnLButtonDown()
{
        ...
}
```

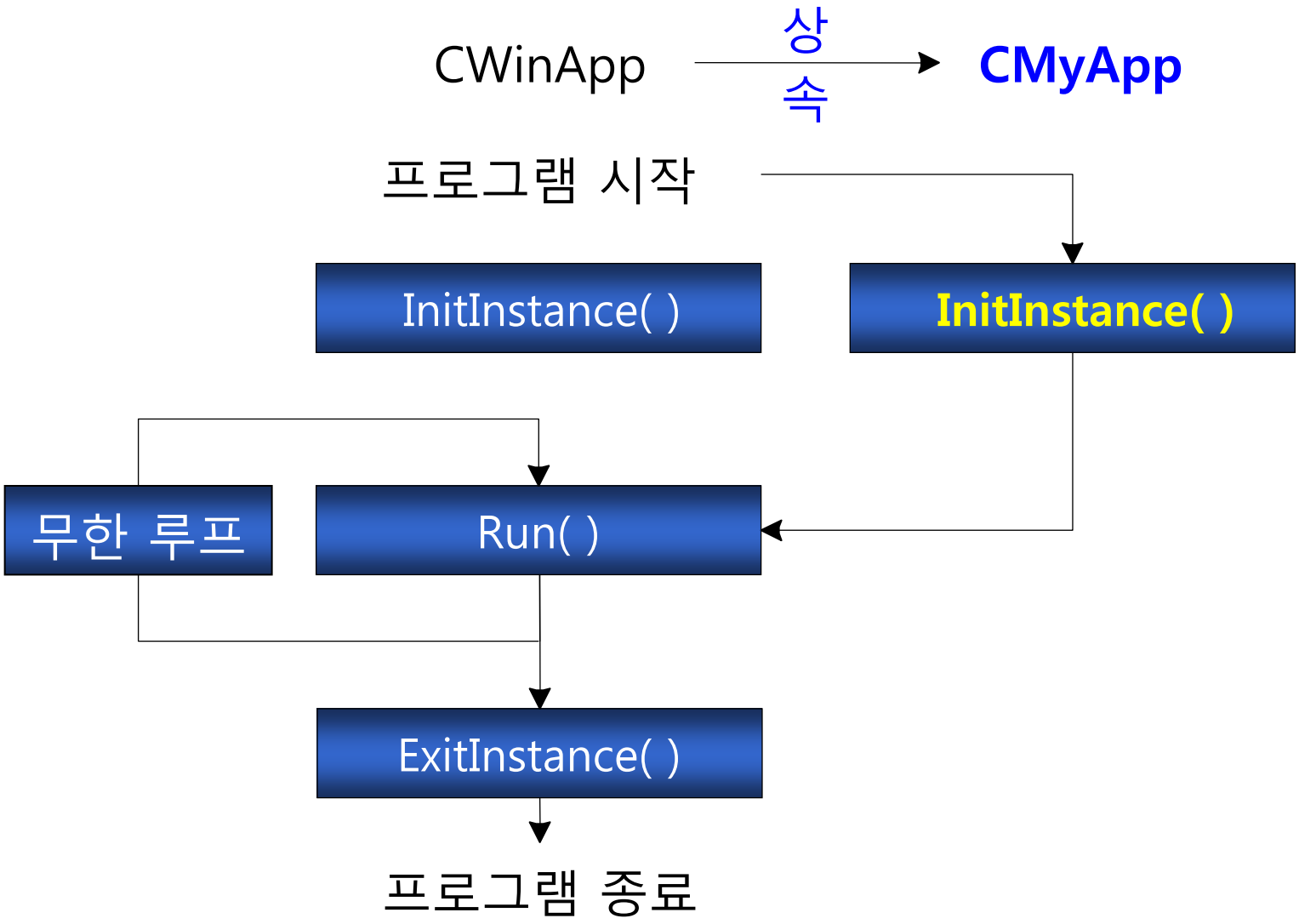
MFC 라이브러리 내부 코드

```
int WINAPI WinMain( . . . ) // 내부에 숨겨진 프로그램 실행 시작점
{
    ptr = ...;  // 응용 프로그램 객체의 주소값으로 변수 ptr 초기화
    ...
    ptr->InitInstance(); // 초기화: 각종 초기화 작업과 더불어
                         // 메인 윈도우 객체 생성
                         // → 메인 윈도우 객체의 생성자에서
                         //   운영체제 수준의 실제 윈도우를 만든다.
    ...
    ptr->Run();    // 메시지 루프: 메시지 큐에서 메시지를 꺼내 처리
                   // → 메인 윈도우가 받은 메시지의 종류에 따라
                   //   해당 메시지 핸들러가 적절히 호출된다.
    ...
    ptr->ExitInstance(); // 종료: 각종 청소 작업 수행
    ...
}
```

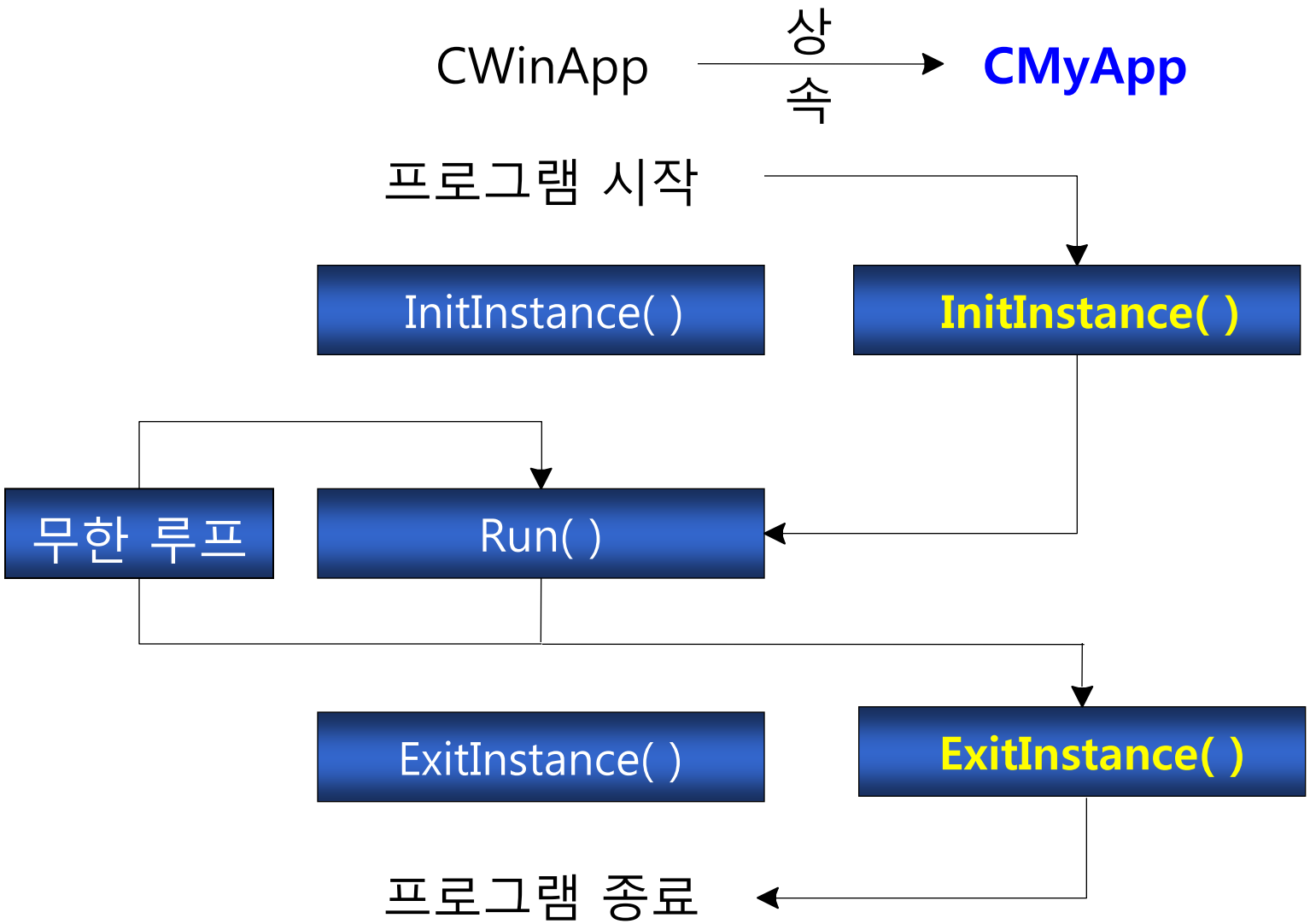
CWinApp (애플리케이션 클래스) (3)



CWinApp (애플리케이션 클래스) (4)



CWinApp (애플리케이션 클래스) (5)



CWnd (윈도우 클래스)

CObject → CCmdTarget → CWnd

- CMainFrame 클래스와 CView 클래스의 부모 클래스
- 윈도우에 대한 클래스
  → 멤버변수로 HWND 타입의 변수를 가짐

CWnd	
멤버변수	멤버함수
HWND m_hWnd	• 첫 번째 인자로 HWND를 갖는 모든 API 함수 • 메시지 핸들러

메시지 핸들러

```
WndProc( . . . )
{
    switch( msg )
    {
    case WM_LBUTTONDOWN:
        MessageBox( hwnd, "안녕하세요", "인사", MB_OK );
        break;
    ...
    }
}
```

- CWnd 클래스에서는 WM_LBUTTONDOWN 등의 메시지에 대한 처리 기능을 하나의 멤버함수로 제공
  → 메시지 핸들러

```
WM_CREATE          →   OnCreate( . . . )
WM_LBUTTONDOWN     →   OnLButtonDown( . . . )
```

메시지 맵 (1)

```
class A
{
    int d;
    int show() { cout << "A"; };
}
class B: public A
{
    int show() { cout << "B"; };
}

void main()
{
    A*  pA;
    B   b;
    pA = &b;          //형-변환은 필요 없음
    pA->show();       //출력결과는?
}
```

메시지 맵 (2)

- B를 출력하려면 ⇒

```
class A
{
     int d;
     int show() { cout << "A"; };
}
class B: public A
{
     int show() { cout << "B"; };
}
```

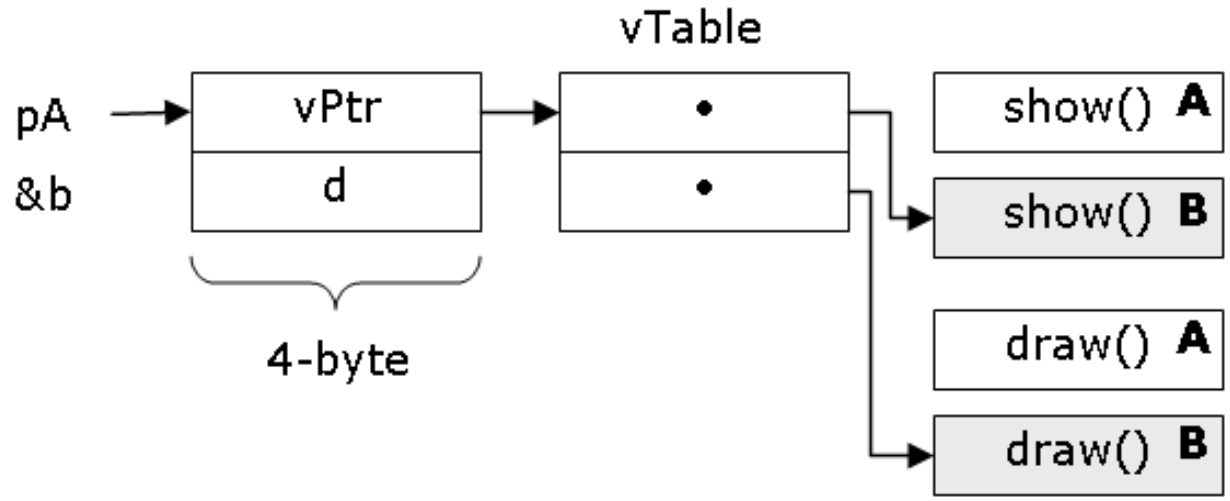
- 클래스 B의 메모리 크기(sizeof(B)) 확인

메시지 맵 (3)

- 클래스 A에 virtual 함수를 하나 더 추가하여 클래스 B의 크기 확인?

```
class A
{
    int d;
    virtual int show() { cout << "A"; };
    virtual int draw() { cout << "A"; };
}
class B: public A
{
    int show() { cout << "B"; };
}
```

메시지 맵 (4)

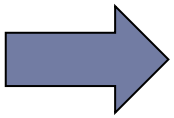


- 변수 b의 메모리 크기(8)
  ⇒ 가상함수 테이블 vTable을 가리키는 포인터 vPtr(4) + 정수변수 d(4)
- 가상함수 테이블의 크기는 virtual 함수의 증가에 따라 4바이트(가상함수의 주소)씩 증가
- 가삼함수 테이블의 함수 주소는 실행시간에 결정
  ⇒ 동적 바인딩(dynamic binding)

메시지 맵 (5)

- 메시지 핸들러는 반드시 가상함수이어야 함
  - CWnd 클래스 상속 받아 메시지 핸들러 재정의
  - 재정의한 핸들러가 호출되도록 하기 위해 가상함수 필요

- **문제점**
  - 가상함수 테이블의 크기가 늘어남
  - 모든 메시지 핸들러를 가상함수로 선언할 경우 메모리 낭비
  - 가상함수 테이블 관리 필요



메시지 맵

```
DECLARE_MESSAGE_MAP()
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
     ON_WM_CREATE()
     ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
```

CFrameWnd (프레임 윈도우 클래스)

CObject → CCmdTarget → CWnd → CFrameWnd

- WM_COMMAND 메시지 처리 가능
- 자식 윈도우를 만들 수 있는 토대 마련

CFrameWnd	
멤버함수	
OnCreate()	// WM_CREATE 메시지 핸들러
PreCreateWindow()	// 윈도우 생성 옵션 지정

```
afx_msg int OnCreate( LPCREATESTRUCT  lpCreateStruct );
```

WM_CREATE 메시지 핸들러에서 자식 윈도우 생성

```
virtual Bool PreCreateWindow( CREATESTRUCT & CS );
```

윈도우 클래스 정의 부분의 일부를 처리

CView (뷰 클래스)

CObject → CCmdTarget → CWnd → CView

- WM_COMMAND 메시지 처리
- 화면출력에 대한 함수 제공
- 데이터를 저장하는 도큐먼트 클래스에 대한 접근 함수 제공
- 뷰 초기화 함수 제공

CView	
멤버함수	
OnDraw()	// 화면 출력
GetDocument()	// 도큐먼트 객체의 주소 획득
OnInitialUpdate()	// 뷰 초기화

CDocument (도큐먼트 클래스)

CObject → CCmdTarget → CDocument

- WM_COMMAND 메시지 처리
- 데이터 처리를 위해 파일 처리와 관련된 함수 제공
- 뷰에게 데이터 변경을 통보하는 함수 제공
- 객체의 재생성 없이 초기화 하는 함수 제공

CDocument	
멤버함수	
OnOpenDocumnet()	// 문서 열기
OnCloseDocument()	// 문서 닫기
OnSaveDocument()	// 문서 저장
UpdateAllViews()	// 화면 갱신 공지
OnNewDocument()	// 문서 초기화
Seriallize()	// 시리얼라이제이션 (serialization)

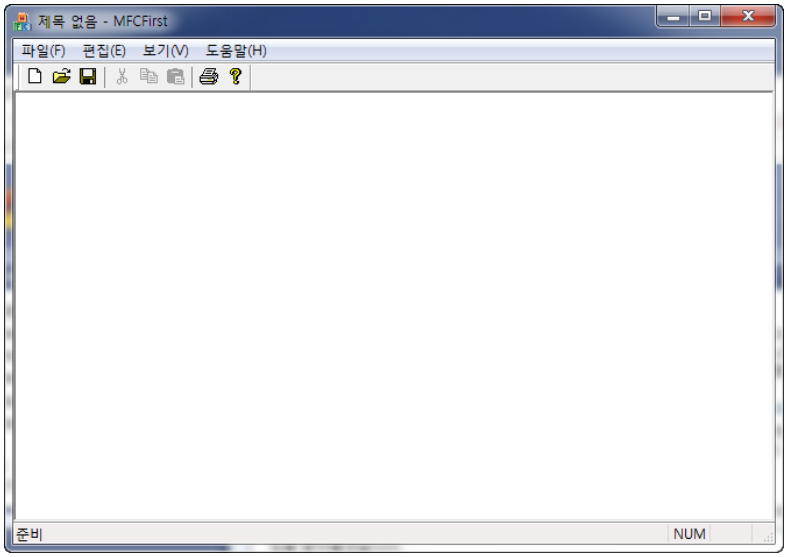
초기화 멤버함수들

클래스	초기화 함수
CWinApp	InitInstance()
CFrameWnd	OnCreate()
CView	OnInitialUpdate()
CDocumnet	OnNewDocument()
CDialog	OnInitDialog()

- 객체가 만들어질 때 생성자 다음으로 자동으로 호출되는 함수
- CFrameWnd 클래스의 OnCreate() 핸들러는 초기화를 위해 정의된 것은 아니지만 초기화 작업에 적합
- CDialog 클래스의 OnInitDialog()는 대화상자 윈도우에서 초기화를 위해 발생시키는 WM_INITDIALOG 메시지에 대한 핸들러

윈도우의 개수는?

- 프로그램 실행 화면의 윈도우 개수는?
  - 윈도우 = HWND

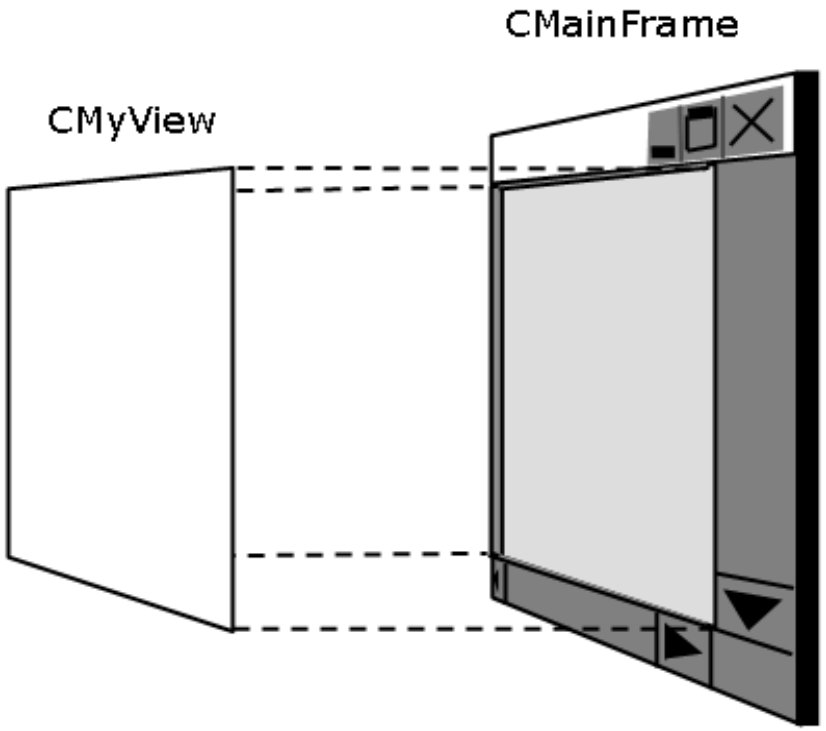


① 1개 ② 2개 ③ 3개 ④ 4개 ⑤ 그 이상

CWnd
HWND m_hWnd

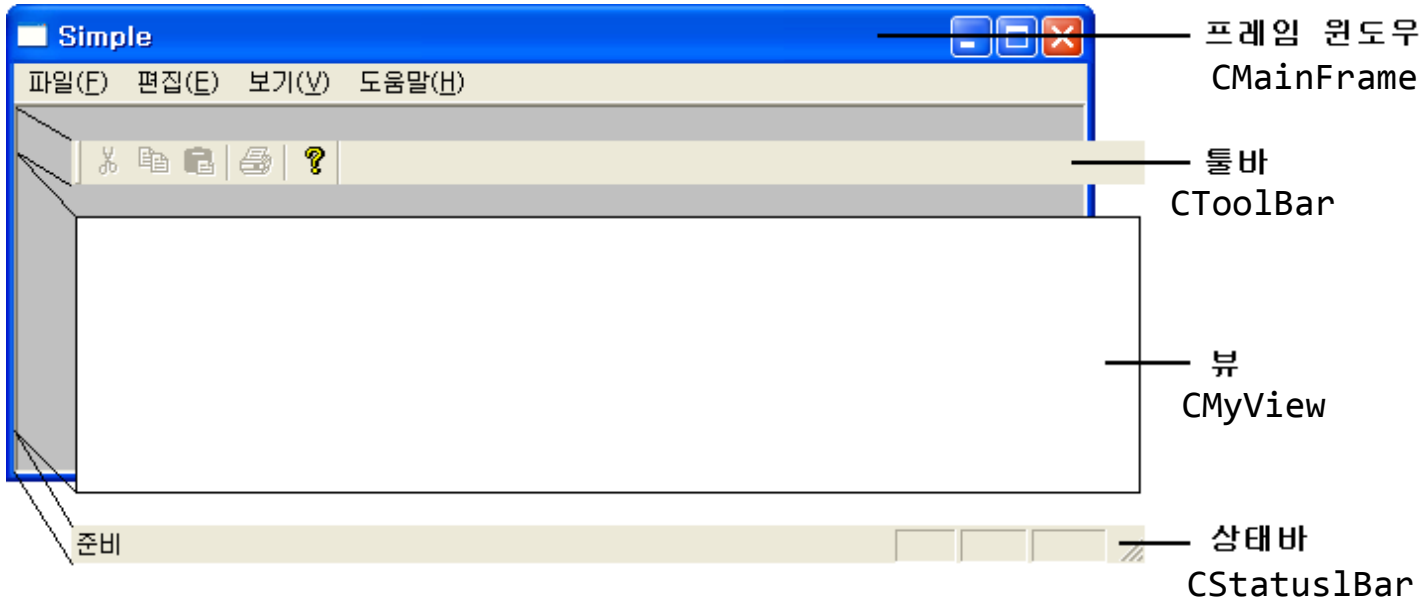
CMainFrame —— 프레임 윈도우

C__View —— 클라이언트 윈도우
(CMainFrame 클래스 윈도우의 자식윈도우
- WS_CHILD속성을 가짐)



Cobject → CCmdTarget → CWnd → CControlBar → CToolBar/CStatusBar

CToolBar와 CStatusBar도 CWnd 클래스를 상속받았으므로 윈도우

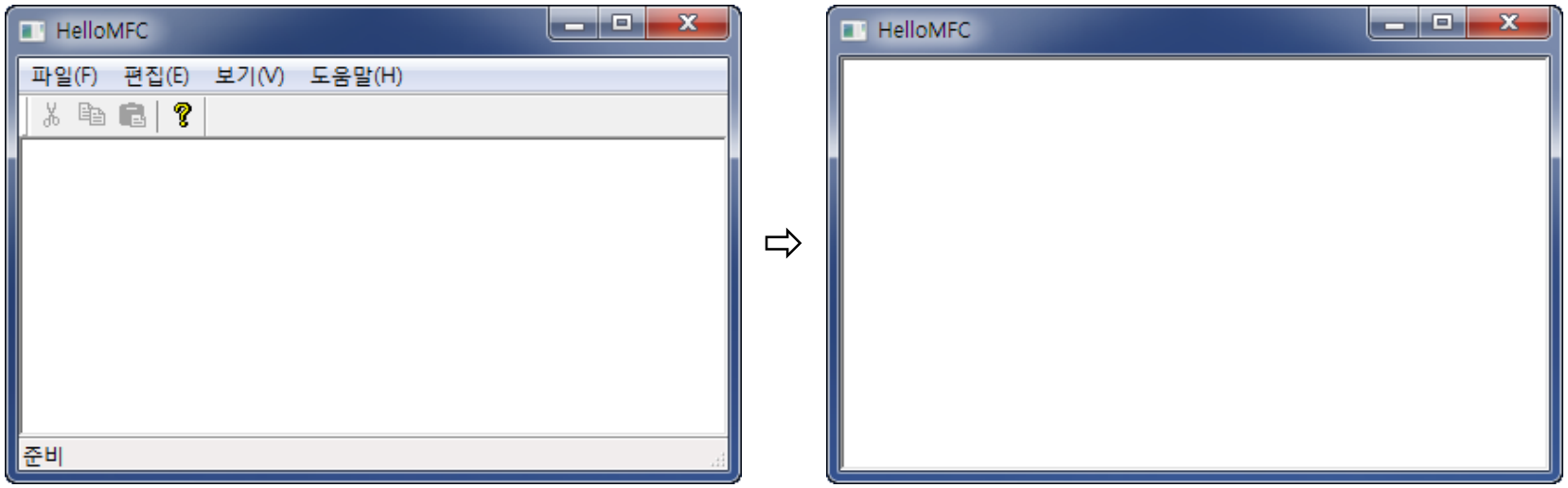


실습 5.1

- 툴바 및 상태바를 제거하자.

실습 5.2

- 메뉴를 제거하자.



윈도우 클래스 정의

```
WNDCLASS WndClass;
char szAppName[] ="This program is to create window";

WndClass.style = NULL;
WndClass.lpfnWndProc = WndProc;
WndClass.cbClsExtra = 0;
WndClass.cbWndExtra = 0;
WndClass.hInstance = hInstance;
WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
WndClass.lpszMenuName = NULL;
WndClass.lpszClassName = szAppName;
```

MFC프로그램 에서도 윈도우가 생성되기 전에 메뉴필드를 NULL로 처리하면 메뉴를 없앨 수 있다!

윈도우 메시지처리는 어디에서?

- 프로젝트를 구성하는 네 개의 클래스 중에서 WM_LBUTTONDOWN 메시지를 처리할 수 있는 클래스는 어느 것인가? 다음 보기에서 모두 고르시오.

① C__App ② CMainFrame ③ C__Doc ④ C__View

Lab.

- 다음의 요구사항을 만족하는 윈도우를 생성하자.
  - 프레임 윈도우의 시스템 메뉴 제거
  - 초기 윈도우의 크기를 500x300으로 지정
  - 클라이언트 영역에서의 커서의 모양이 십자형이 되도록 지정
  - 클라이언트 영역의 색상을 파란색으로 출력
    - 힌트: CreateSolidBrush( ) 함수 사용