

# 객체 지향 프로그래밍 응용

## Chap 4. 대화상자와 컨트롤 (#1)



2012.09.27.

오 병 우

컴퓨터공학과  
금오공과대학교

# Dialog 개요

## Control들을 가진 윈도우

### ◆ 사용자의 입력을 받기 위한 Object의 집합

- 프로그램 수행 도중 사용자의 입력이 필요할 때 다이얼로그 박스 출력
- 다이얼로그 박스는 사용자로부터 입력 받은 데이터를 메인 루틴에 넘기고 소멸

## 종류

### ◆ Modal Dialog

- Parent window 비활성화 (열기)

### ◆ Modeless Dialog

- Parent windows 활성화 유지 (찾기)

### ◆ Common Dialog

- 운영체제에 내장 (글꼴)

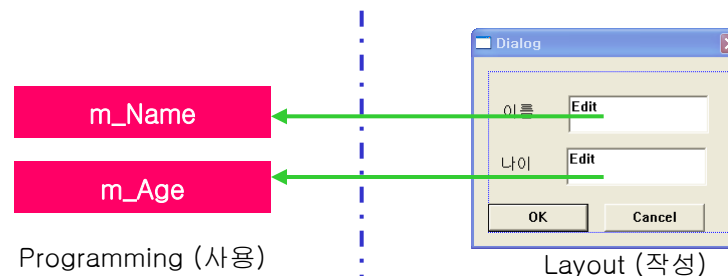
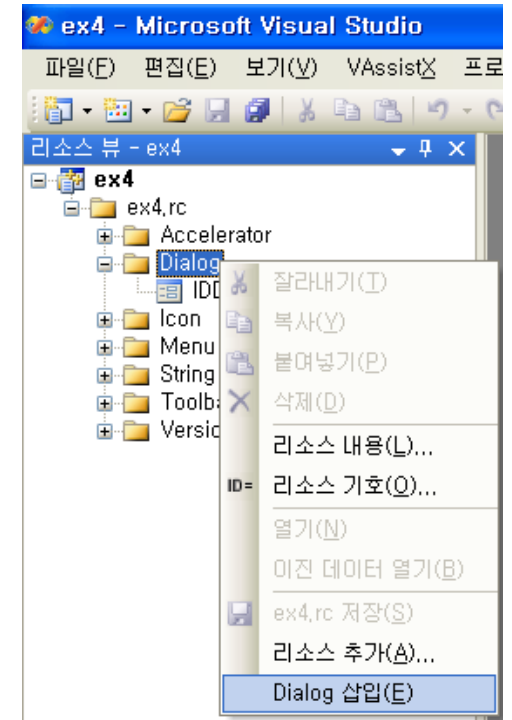
# Dialog 작성 및 사용 방법

## ■ 작성

- ◆ VC++의 리소스 편집기를 사용하여 layout 구성
  - \*.rc 파일에 저장

## ■ 사용

- ◆ Control Notification
  - ON\_CONTROL, ON\_BN\_CLICKED, ON\_EN\_UPDATE() 등의 Message Map Entry Macro 사용
- ◆ 컨트롤에 해당하는 멤버 변수(Value Type or Control Type)를 만들고 DDX(DoDataExchange)를 통해 동기화



# 실 습

## □ 목표

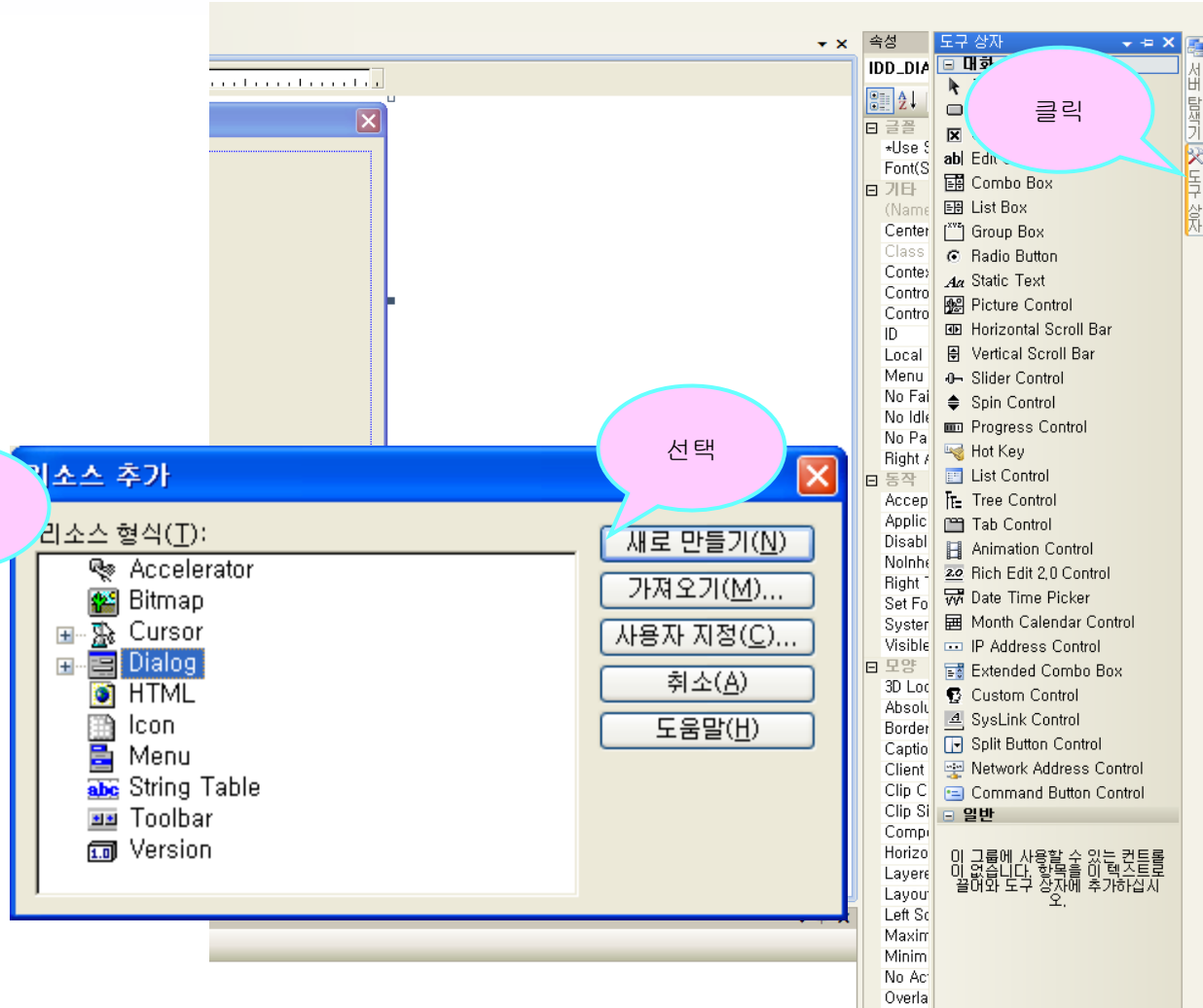
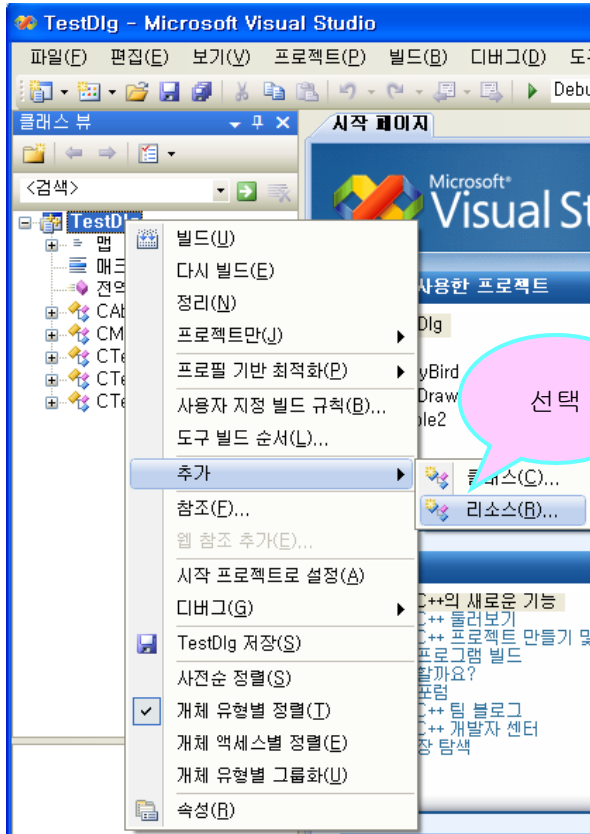
- ◆ MFC-SDI(Single Document Interface) 사용
- ◆ Dialog를 통해 이름과 Font를 입력 받아서 화면에 출력
- ◆ 윈도우가 가렸다가 나타나도 내용이 지워지지 않도록 Document 클래스 사용

## □ 과정

- ◆ Dialog 생성
  - 이름을 입력 받기 위해 Control 들을 화면에 배치하여 **Layout 구성**
  - 작성한 Dialog를 Control 하기 위한 **CDialog의 파생 Class 생성 및 연결**
- ◆ Edit Control의 **Value Type Variable 생성**
  - CDialog의 파생 Class의 Member Variable과 연결
- ◆ 메뉴에서 작성한 Dialog를 호출할 수 있도록 메시지 핸들러 작성
  - Document Class에 입력 받은 이름을 저장할 수 있는 Member Variable 생성 및 초기화
  - 작성한 Dialog의 header file을 include
  - 메뉴 처리 메시지 핸들러에서 **Dialog 변수 생성**
  - **DoModal() 함수 호출**
- ◆ Font를 위한 Common Dialog를 호출할 수 있도록 메시지 핸들러 작성
  - Document Class에 입력 받은 Font와 Color를 저장할 수 있는 Member Variable 생성 및 초기화

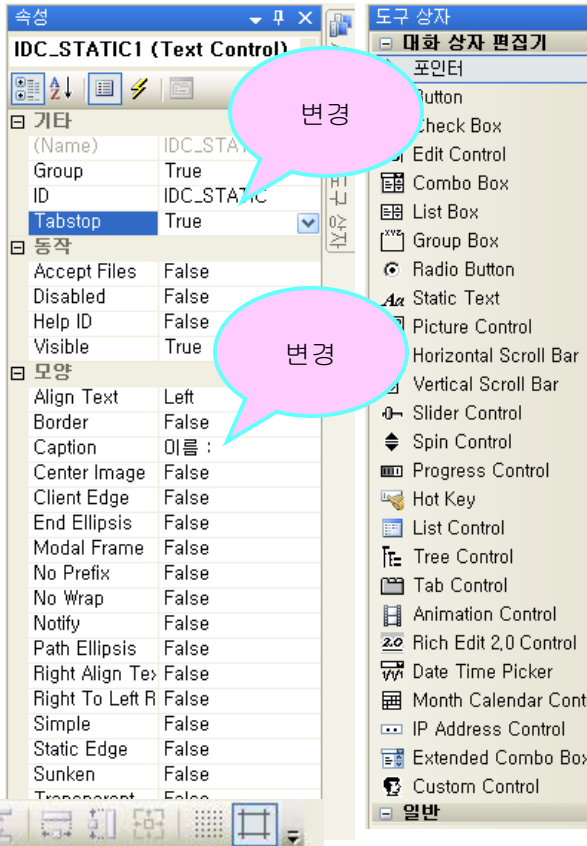
# 실습

## 리소스 추가



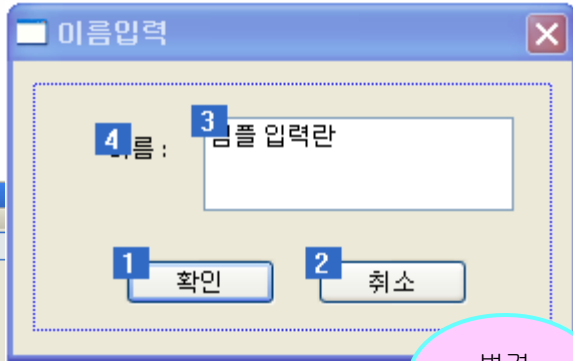
# 실습

- Dialog 박스 Layout 구성
  - ◆ 서식 | 탭 순서 (ctrl + D)



변경

변경



변경  
Resource.h

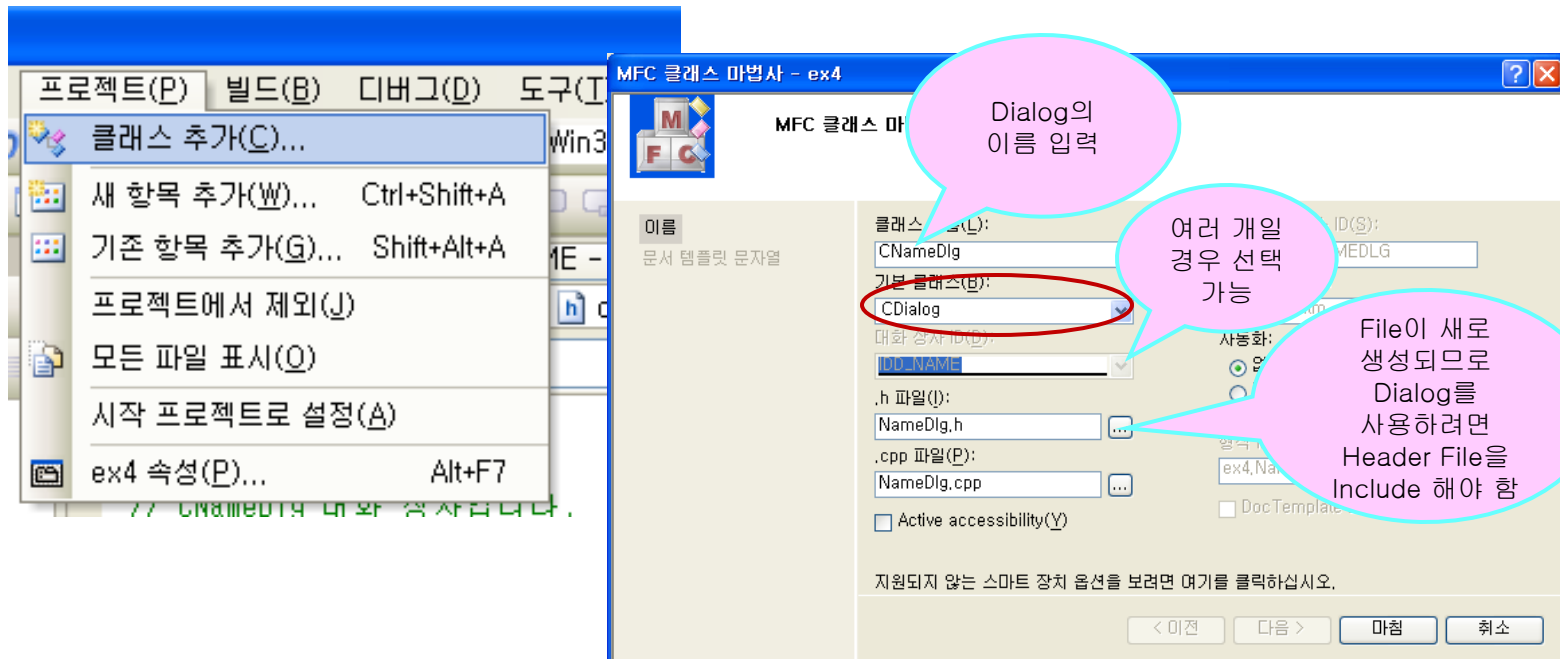
변경

테스트

# 실습

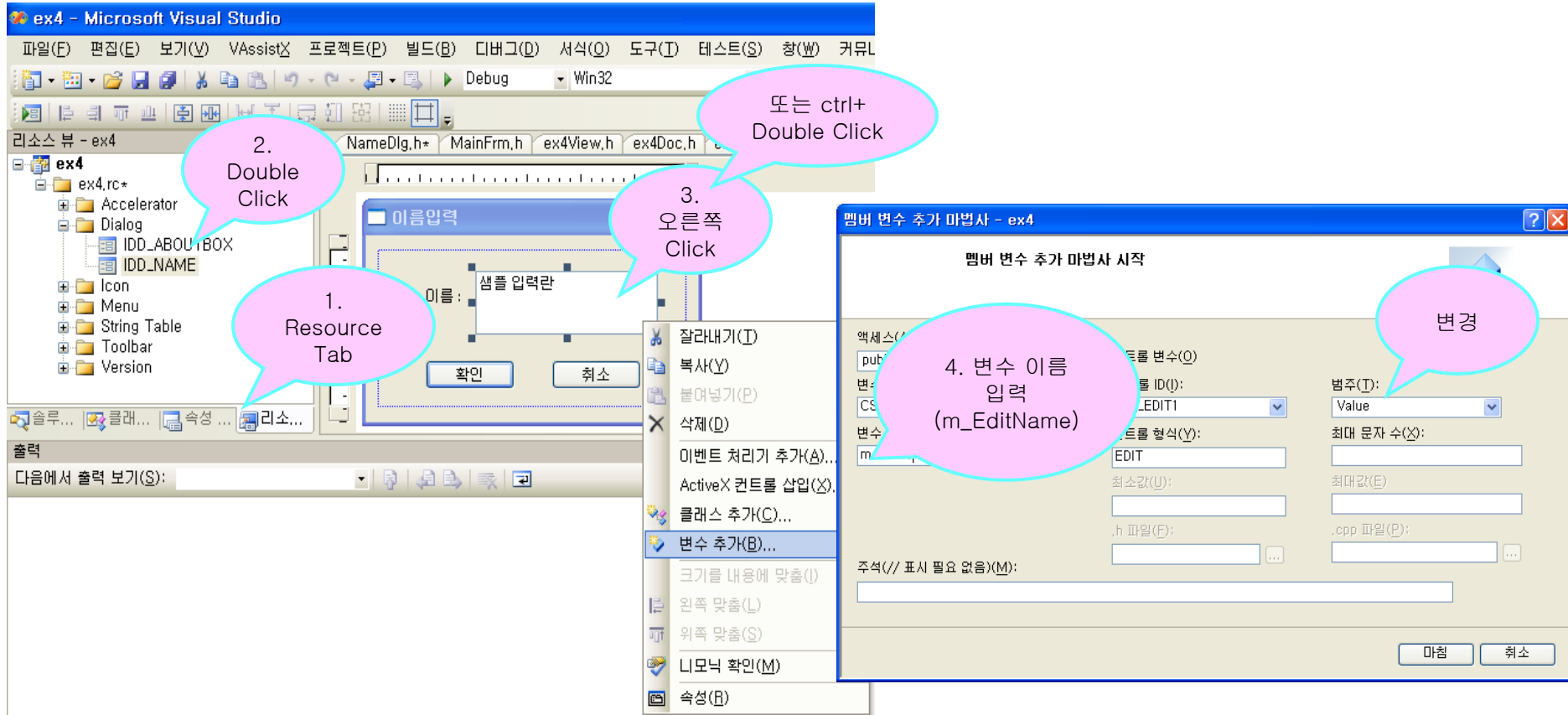
## □ Dialog 클래스 생성

- ◆ 다이얼로그를 선택한 후에 프로젝트|클래스 추가
- ◆ 바로 나타나지 않는다면 왼쪽에 있는 범주를 MFC로 선택하고 오른쪽 템플릿에서 MFC 클래스 선택 후에 추가 버튼 클릭
  - 기본 클래스를 CDialog로 선택하고 대화상자 ID 선택



# 실습

## Value Type 변수 연결



The screenshot illustrates the steps to connect a value type variable in Visual Studio:

- 1. Resource Tab**: The Resource View on the left shows the project structure.
- 2. Double Click**: A callout points to the 'IDD\_ABOUTBOX' resource in the Resource View.
- 3. 오른쪽 Click**: A callout points to a right-click action on the dialog box in the Design view.
- 또는 ctrl+ Double Click**: A callout points to the 'Add Member Variable' option in the context menu.
- 4. 변수 이름 입력 (m\_EditName)**: A callout points to the 'Variable Name' field in the 'Add Member Variable' wizard.
- 변경**: A callout points to the 'Value' dropdown in the 'Add Member Variable' wizard.



# 실습

## ■ 생성 결과

### Header File (NameDlg.h)

```
#pragma once

// CNameDlg 대화 상자입니다.

class CNameDlg : public CDialog
{
    DECLARE_DYNAMIC(CNameDlg)

public:
    CNameDlg(CWnd* pParent = NULL); // 표준 생성자입니다.
    virtual ~CNameDlg();

    // 대화 상자 데이터입니다.
    enum { IDD = IDD_NAME };
    CString m_EditName;

protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV 지원입니다.

    DECLARE_MESSAGE_MAP()

};
```

### Source File (NameDlg.cpp)

```
// NameDlg.cpp : 구현 파일입니다.
//

#include "stdafx.h"
#include "ex4.h"
#include "NameDlg.h"

// CNameDlg 대화 상자입니다.
IMPLEMENT_DYNAMIC(CNameDlg, CDialog)

CNameDlg::CNameDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CNameDlg::IDD, pParent)
{
    // ...
    m_EditName = _T("");
}

CNameDlg::~CNameDlg()
{
}

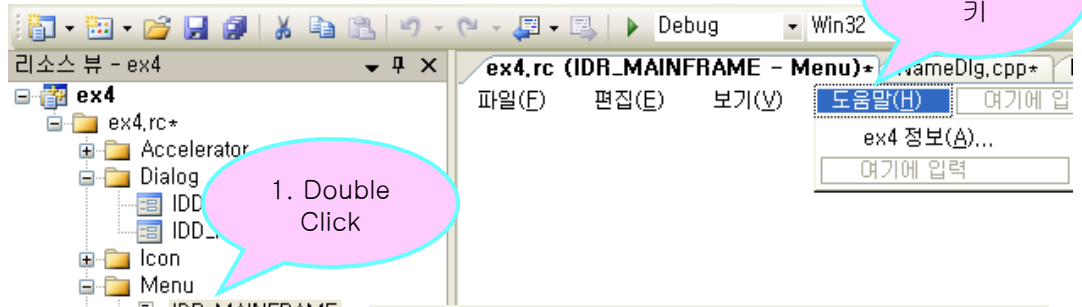
void CNameDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_EDIT1, m_EditName);
}

BEGIN_MESSAGE_MAP(CNameDlg, CDialog)
    END_MESSAGE_MAP()

// CNameDlg 메시지 처리기입니다.
```

# 실습

## 메뉴에 추가



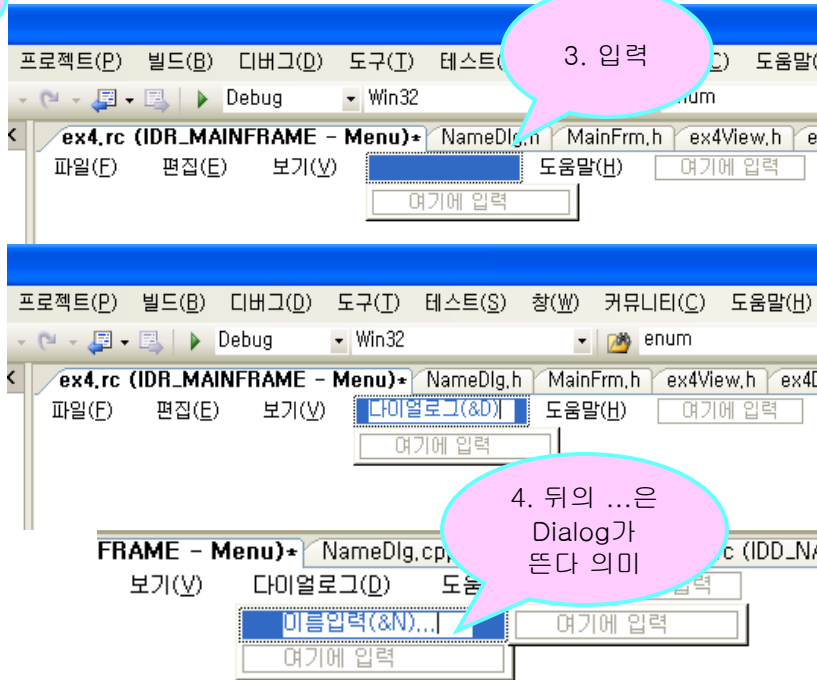
1. Double Click

2. Insert 키

3. 입력

속성	
메뉴 편집기 IMenuEd	
기타	
(Name)	메뉴 편집기
Help	False
ID	IDM_NAME
Prompt	
Separator	False
동작	
Break	None
Right Justify	False
Right Order	False
모양	
Caption	이름 입력(&N) ...
Checked	False
Enabled	True
Grayed	False
Popup	False

5. 메뉴는 IDM\_으로 시작하는 아이디 부여



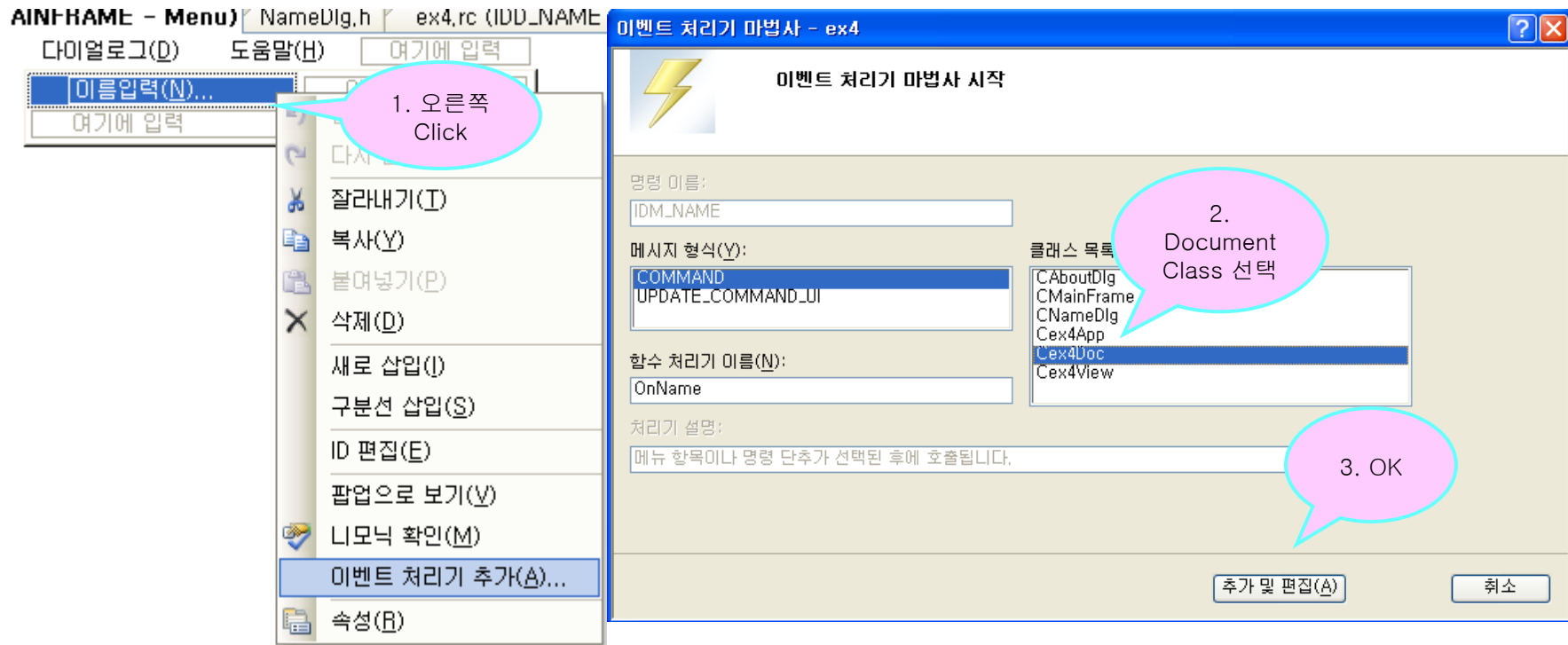
3. 입력

4. 위의 ...은 Dialog가 뜬다 의미

# 실습

## □ 메뉴 처리 메시지 핸들러 생성

### ◆ 마우스 오른쪽 클릭



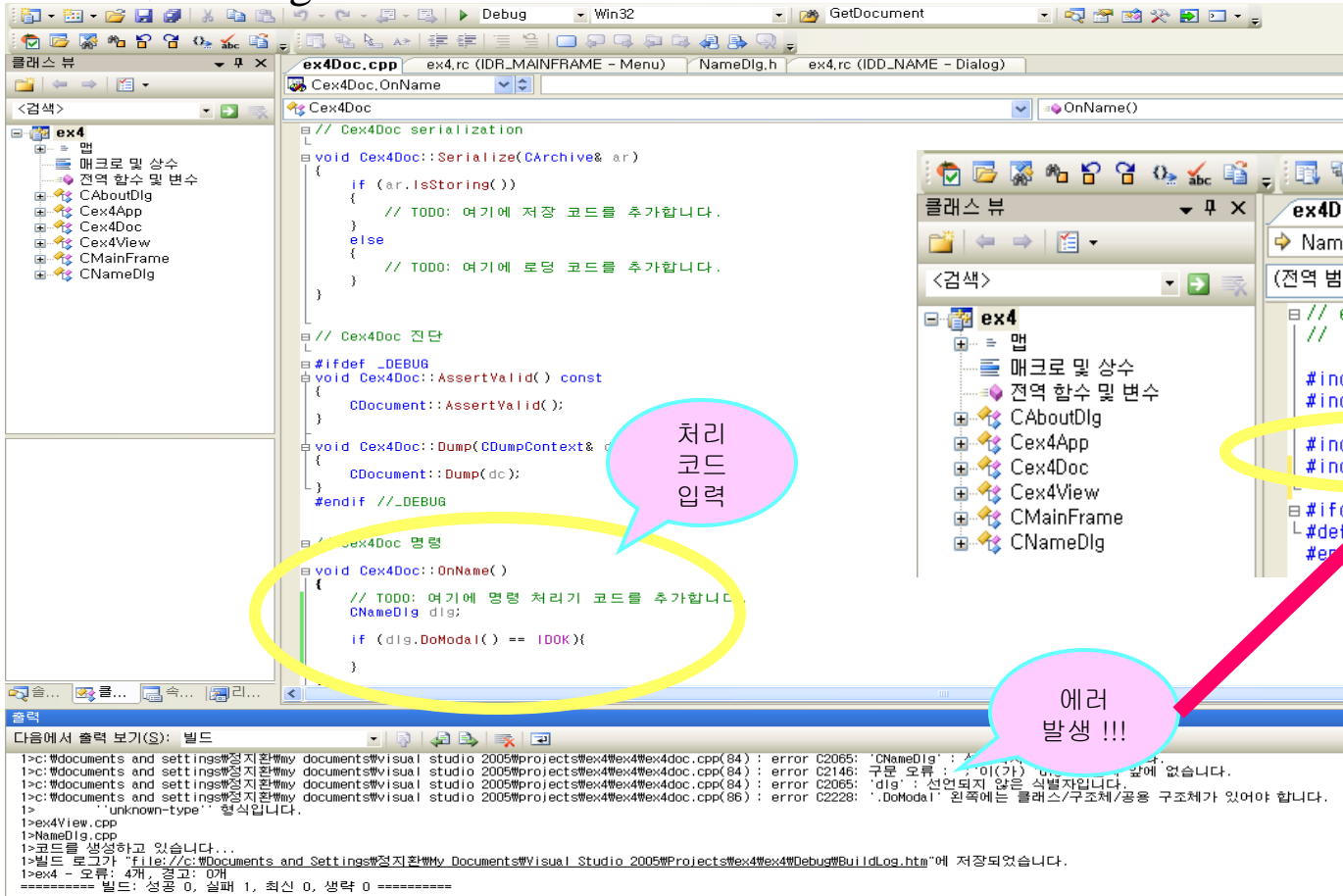
The screenshot illustrates the steps to create a menu handler for a right-click event in a Visual Studio project:

- 1. 오른쪽 Click**: A context menu is opened over the '여기에 입력' (Enter text here) text box. The '이벤트 처리기 추가(A)...' (Add Event Handler...) option is highlighted.
- 2. Document Class 선택**: The '이벤트 처리기 마법사 - ex4' (Add Event Handler Wizard) dialog is shown. The 'MESSAGE\_TYPE' is set to 'COMMAND', and 'Cex4Uoc' is selected in the 'CLASS\_LIST'.
- 3. OK**: The 'OK' button is clicked to complete the wizard.

# 실습

## 메뉴 처리 메시지 핸들러 구현

### ◆ Dialog의 Header File을 Include 해야 함



The screenshot shows the Visual Studio IDE with the following components:

- Left Panel (Solution Explorer):** Shows the project structure for 'ex4', including files like CAboutDlg, Cex4App, Cex4Doc, Cex4View, CMainFrame, and CNameDlg.
- Editor (ex4Doc.cpp):** Contains the implementation of the `Cex4Doc::OnName()` method. A yellow circle highlights the call to `CNameDlg dlg;` and `dlg.DoModal() == IDOK`. A pink callout bubble points to this area with the text "처리 코드 입력" (Input processing code).
- Editor (NameDlg.h):** Shows the header file for `NameDlg`. A yellow circle highlights the `#include "ex4Doc.h"` and `#include "NameDlg.h"` lines. A pink callout bubble points to these lines with the text "Header File을 Include" (Include Header File).
- Bottom Panel (Output Window):** Displays compilation errors. A pink callout bubble points to the error messages with the text "에러 발생!!!" (Error!!!). The errors include:
  - error C2065: 'CNameDlg' : 선언되지 않은 식별자입니다.
  - error C2146: 구분 오류: 이(가) 줄의 끝 부분에 닫는 괄호가 없습니다.
  - error C2065: 'dlg' : 선언되지 않은 식별자입니다.
  - error C2228: '.DoModal()' 항목에는 클래스/구조체/공용 구조체가 있어야 합니다.

# 실습

## Dialog 사용하기

```

CEx4Doc
// Ex4Doc.h : CEx4Doc 클래스의 인터페이스
//

#pragma once

class CEx4Doc : public CDocument
{
protected:
    CString m_Name;
public:
    CString GetName() { return m_Name; };

    // CEx4Doc 생성/소멸
    CEx4Doc()
    {
        // TODO: 여기에 일회성 생성 코드를 추가합니다.
        m_Name = "";
    }

    CEx4Doc::~CEx4Doc()
    {
    }

    BOOL CEx4Doc::OnNewDocument()
    {
        if (!CDocument::OnNewDocument)
            return FALSE;

        // TODO: 여기에 객체 초기화 코드를 추가합니다.
        // SDI 문서는 이 문서를 다시 사용합니다.
        m_Name = "";

        return TRUE;
    }
}
    
```

이름  
저장을  
위한  
변수

이름  
반환  
함수

초기화  
1

초기화  
2

```

// Cex4Doc 명령
void Cex4Doc::OnName()
{
    // TODO: 여기에
    CNameDlg dlg;

    if (dlg.DoModal() == IDOK){
        m_Name = dlg.m_EditName;
        UpdateAllViews(NULL);
    }
}
    
```

Dialog  
Box  
표시

Document에서  
데이터가  
변경되었으니  
View를 다시  
그리시오.

주석으로  
되어  
있으면  
폴 것

```

// CEx4View 그리기
void CEx4View::OnDraw(CDC* pDC)
{
    CEx4Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;

    // TODO: 여기에 윈드 데이터에 대한 그리기 코드를 추가합니다.
    pDC->TextOutA(10, 10, pDoc->GetName());
}
    
```

**F5 눌러서 실행**

# 실습

## Dialog의 Value Type 멤버 변수 세팅

```

// Cex4Doc 명령
void Cex4Doc::OnName()
{
    // TODO: 여기에 명령 처리기 코드
    CNameDlg dlg;

    dlg.m_EditName = m_Name;

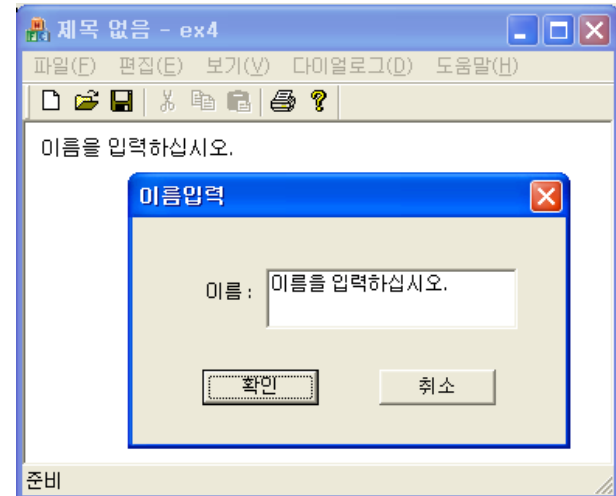
    if (dlg.DoModal() == IDOK){
        m_Name = dlg.m_EditName + "님 참 잘했어요";
        UpdateAllViews(NULL);
    }
}
    
```

현재의 이름으로 초기화

```

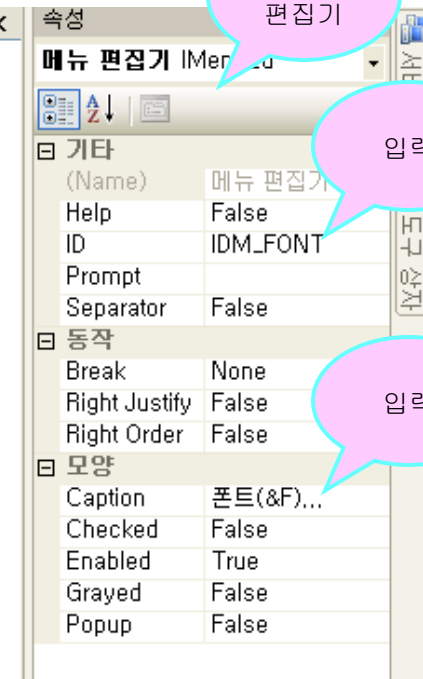
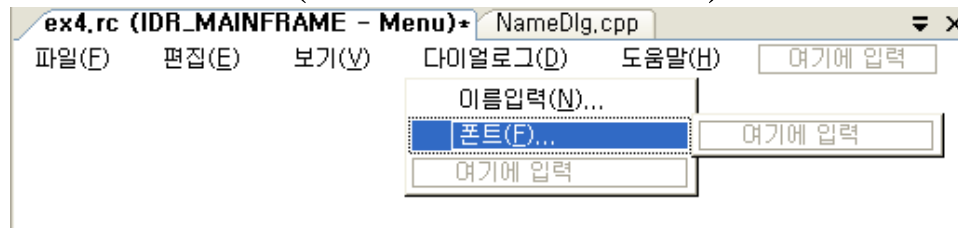
// Cex4Doc 생성/소멸
Cex4Doc::Cex4Doc()
{
    // TODO: 여기에 일회성 생성 코드를 추가합니다.
    m_Name = "이름을 입력하십시오.";
}
    
```

초기화 부분 변경, OnNewDocument()도 같이 변경



# 실습

## ☐ 폰트를 위한 메뉴 추가 (속성 편집창 이용)



# 실습

## View 클래스에서 Font 처리

```

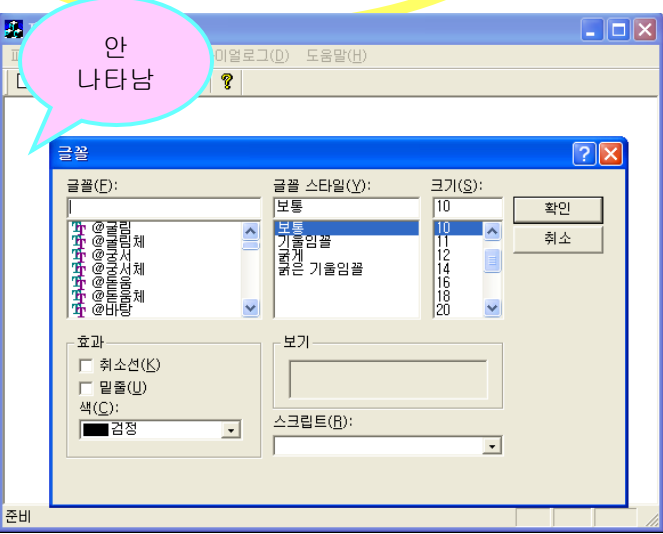
class CEx4View
{
// CEx4View 메시지 처리기
void CEx4View::OnFont()
{
// TODO: 여기에 명령 처리기 코드를 추가합니다.
CFontDialog dlg;

if (dlg.DoModal() == IDOK) {
    dlg.GetCurrentFont(&m_Font);
    m_Color = dlg.GetColor();
    Invalidate();
}
}
}
    
```

```

class Cex4View : public CView
{
protected: // serialization에서만 만들어집니다.
    Cex4View();
    DECLARE_DYNCREATE(Cex4View)

protected:
    LOGFONT m_Font;
    COLORREF m_Color;
}
    
```



```

class CEx4View
{
void CEx4View::OnDraw(CDC* pDC)
{
    CEx4Doc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;

// TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.
CFont font;
font.CreateFontIndirectA(&m_Font);

CFont *oldFont = pDC->SelectObject(&font);
pDC->SetTextColor(m_Color);

pDC->TextOutA(10, 10, pDoc->GetName());

pDC->SelectObject(&oldFont);
}
}
    
```



# 실습

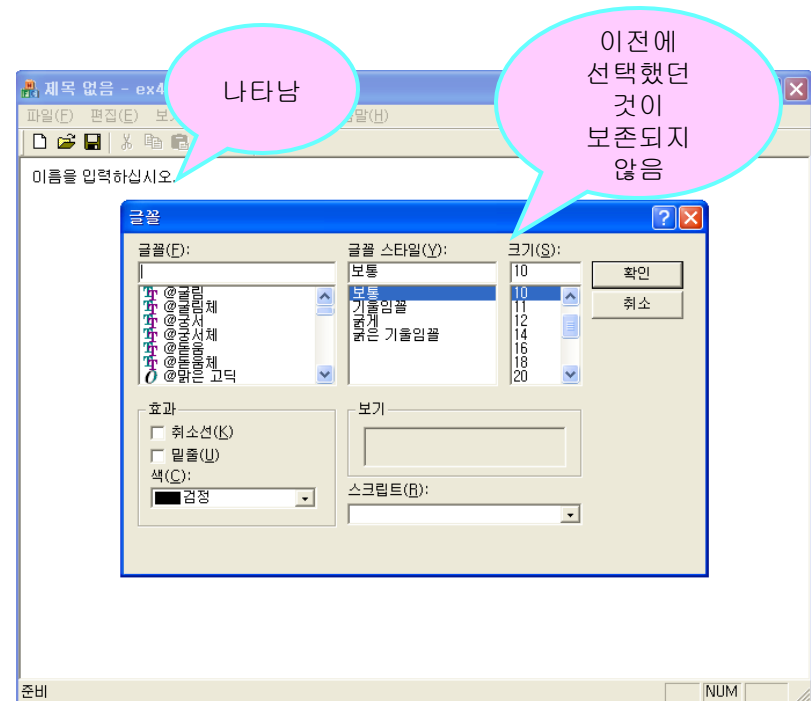
## Font 초기화 처리

```

CEx4View
// CEx4View 생성/소멸

CEx4View::CEx4View()
{
    // TODO: 여기에 생성 코드를 추가합니다.
    CFont font;
    font.CreatePointFont(10 * 10, "굴림");
    font.GetLogFont(&m_Font);

    m_Color = RGB(0, 0, 0);
}
    
```



# 실습

## Font Dialog에서 이전 글꼴 및 색 초기화

```

// Cex4View 메시지 처리기
void Cex4View::OnFont()
{
    // TODO: 여기에 명령 처리기 코드를 추가합니다.
    CFontDialog dlg(&m_Font);
    dlg.m_cf.rgbColors = m_Color;

    if (dlg.DoModal() == IDOK){
        dlg.GetCurrentFont(&m_Font);
        m_Color = dlg.GetColor();
        Invalidate();
    }
}

```

