# 윈도우 프로그래밍 기초

(5주차)

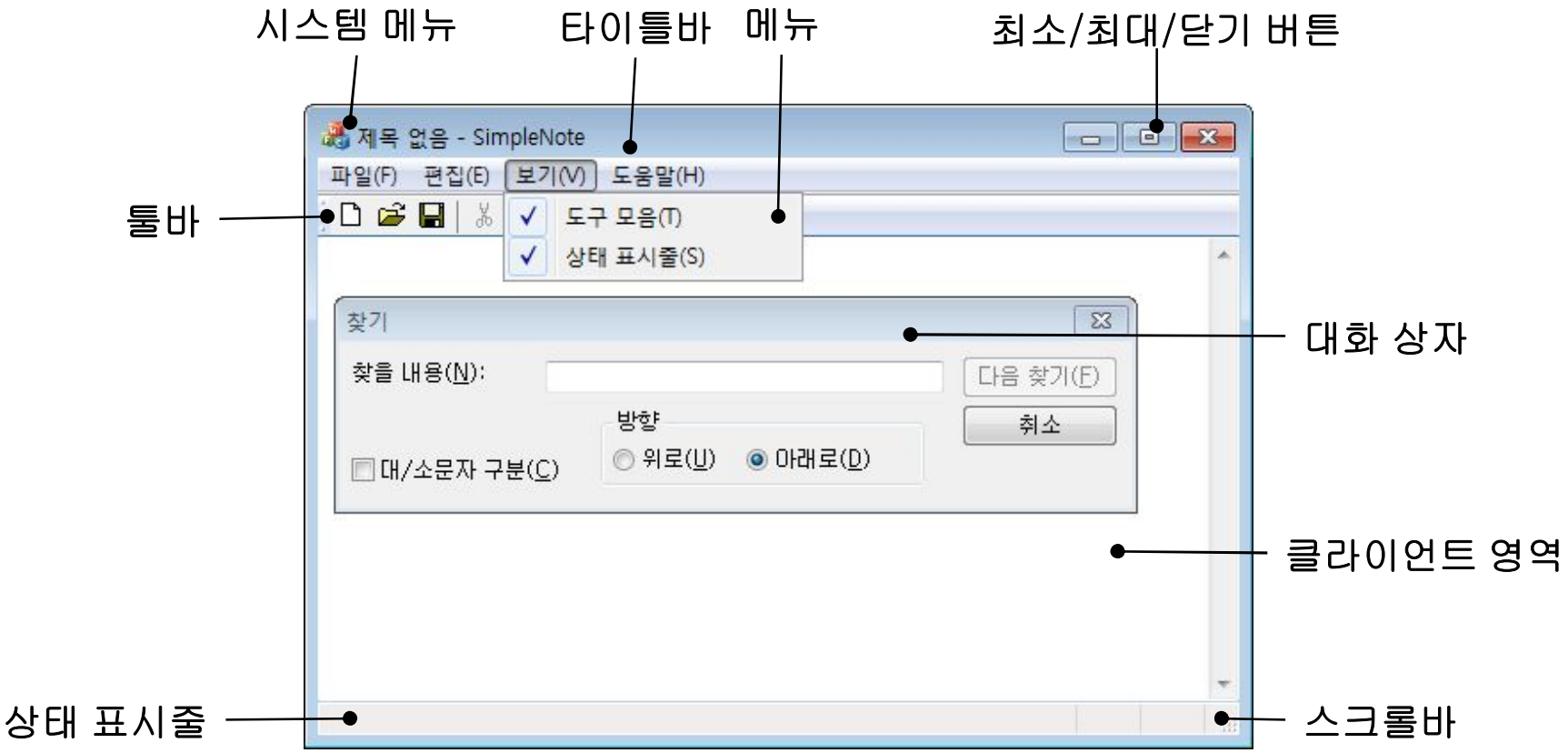
# 학습개요

- 학습 목표
  - 윈도우 운영체제와 윈도우 응용 프로그램의 특징을 이해한다.
  - SDK 응용 프로그램의 작성 과정과 기본 구조 및 동작 원리를 이해한다.
  - MFC 응용 프로그램의 작성 과정과 기본 구조 및 동작 원리를 이해한다.

- 학습 내용
  - 윈도우 프로그래밍 개요
  - SDK 프로그램 기본 구조
  - MFC 프로그램 기본 구조
  - 실습

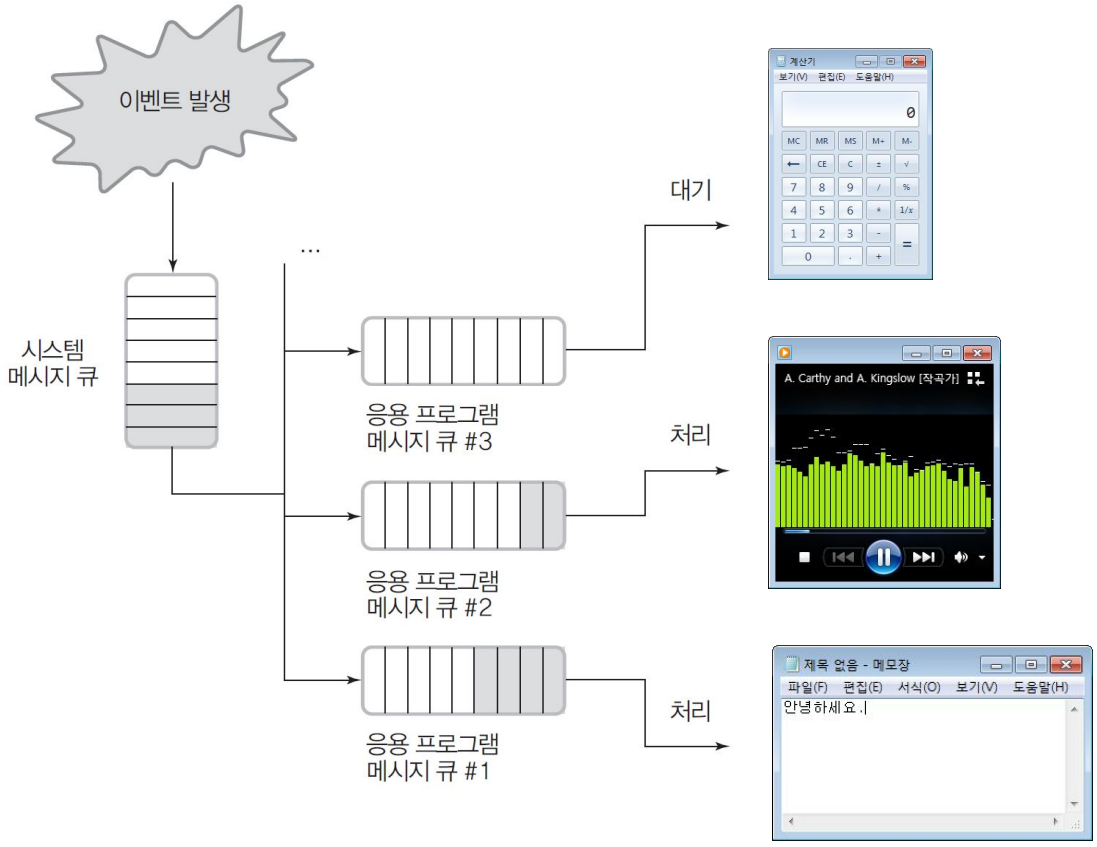
# 윈도우 운영체제의 특징 (1)

- Graphic User Interface



# 윈도우 운영체제의 특징 (2)

- Message Driven Architecture



# 윈도우 운영체제의 특징 (3)

- 멀티태스킹과 멀티스레딩
  - 멀티태스킹(Multitasking)
    - 운영체제가 여러 개의 응용 프로그램을 동시에 실행
  - 멀티쓰레딩(Multithreading)
    - 응용 프로그램 내부에서 여러 개의 실행 흐름(=쓰레드)을 동시에 진행

# 윈도우 응용 프로그램의 특징 (1)

- API 호출문 집합
  - API : Application Programming Interface
  - 윈도우 운영체제가 응용 프로그램을 위해 제공하는 각종 함수의 집합
    - 32 또는 64비트 윈도우 → Win32 API

# 윈도우 응용 프로그램의 특징 (2)

- 메시지 핸들러 집합
  - 메시지 핸들러 : 메시지를 받았을 때 동작을 결정하는 코드

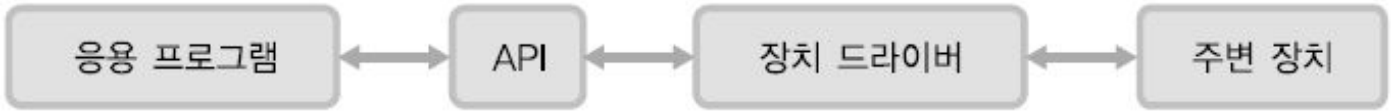

  - 윈도우 프로시저 : 메시지 핸들러의 집합

# 윈도우 응용 프로그램의 특징 (3)

- 실행 파일과 DLL 집합
  - DLL : Dynamic-Link Library
  - 프로그램이 실행 중에 결합하여 사용할 수 있는 코드와 리소스의 집합
  - 윈도우 운영체제가 제공하는 API는 DLL 형태로 제공되며, 응용 프로그래머는 필요한 기능을 DLL로 제작하기도 함

# 윈도우 응용 프로그램의 특징 (4)

- 장치 독립성
  - 주변 장치가 바뀌어도 장치 드라이버(Device Driver)만 설치하면 프로그램을 수정하지 않고 실행할 수 있음



# 윈도우 응용 프로그램의 개발 방식 (1)

- SDK
  - 특징
    - Win32 API ✚ C 언어 기반의 응용 프로그램 코드
  - 장점
    - API를 직접 다루기 때문에 세부 제어가 가능함
    - 윈도우 운영체제가 제공하는 모든 기능을 사용 가능
    - 생성 코드의 크기가 작고 속도도 빠름
    - Visual C++ Express 버전에서도 개발 가능
  - 단점
    - 다른 개발 방식에 비해 생산성이 매우 낮음

# 윈도우 응용 프로그램의 개발 방식 (2)

- RAD: Rapid Application Development
  - 특징
    - 시각적 화면 디자인 ✚ 응용 프로그램 코드 (Visual Basic, Delphi, PowerBuilder 등)
  - 장점
    - 간편하게 직관적으로 프로그래밍할 수 있음
    - 빠른 시간 내에 원하는 기능의 프로그램 개발 가능
  - 단점
    - SDK나 클래스 라이브러리를 이용한 개발 방식보다 생성 코드의 크기가 크고 실행 속도도 떨어지는 편임
    - 윈도우 운영체제가 제공하는 모든 기능을 활용한 세부적인 제어가 어려운 경우가 있음

# 윈도우 응용 프로그램의 개발 방식 (3)

- MFC 클래스 라이브러리
  - 특징
    - MFC 클래스 라이브러리 ✚ C++ 언어 기반의 응용 프로그램 코드(객체지향언어)
  - 장점
    - SDK를 이용한 방식보다 생산성이 높음
    - API를 직접 사용해서 세부적으로 제어할 수 있음
    - RAD 개발 방식보다 코드 크기와 실행 속도 면에서 유리함
  - 단점
    - 객체 지향 프로그래밍에 익숙해야 함
    - 클래스 라이브러리의 구조와 각 클래스의 기능 및 관계를 파악하기 위한 초기 학습 기간이 긴 편임
    - Visual Studio Community Edition 이상이 필요 (Express 버전은 지원 안함)

# 윈도우 응용 프로그램의 개발 방식 (4)

- .NET 프레임워크
  - 윈도우 운영체제에 설치할 수 있는 소프트웨어 개발 및 실행 환경

  - 특징
    - 공용 언어 런타임(CLR, Common Language Runtime)이라는 소프트웨어 가상 머신을 제공하며, 가상 머신의 제어 하에 응용 프로그램이 구동됨(장치 독립성)
    - 윈도우 API에 버금가는 방대한 라이브러리를 제공하며, 언어에 상관없이 라이브러리를 사용 가능(언어 독립성)

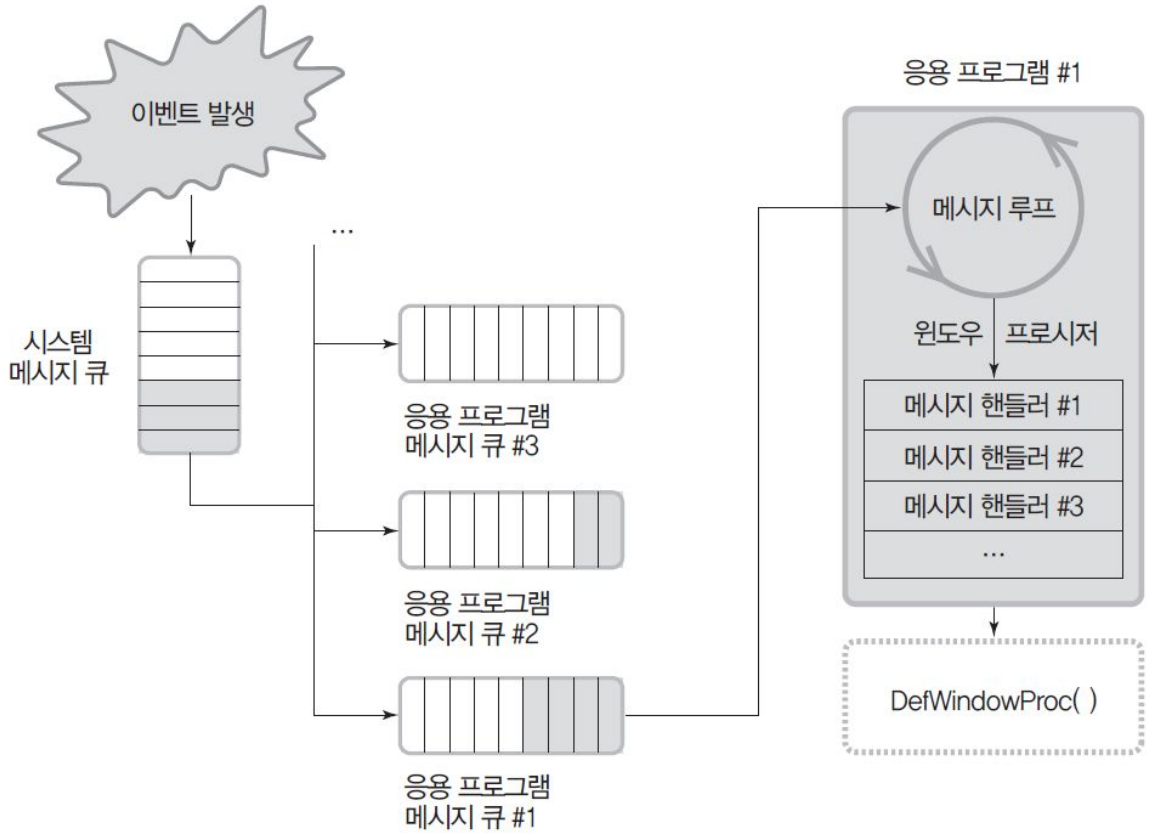
# 실습 – MFC 개발도구 설치 및 환경설정

# SDK 프로그램의 기본 구조 (1)

- SDK 프로그램의 기본 골격
  1. 윈도우 클래스를 정의(초기화)하고 운영체제에 등록함
  2. 윈도우를 생성하고 화면에 보이게 함
  3. 메시지 루프를 구동함
  4. 윈도우 프로시저에서 메시지를 처리함

# SDK 프로그램 기본 구조 (2)

- SDK 프로그램 동작 원리



# HelloSDK 코드 분석 (1)

```
#include <windows.h>

// WinMain 함수에서 참조하므로 함수 원형을 선언한다.
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASS wndclass;
    HWND hwnd;
    MSG msg;

    // 윈도우 클래스를 초기화하고 운영체제에 등록한다.
    wndclass.style = CS_HREDRAW | CS_VREDRAW; // 스타일 지정
    wndclass.lpfnWndProc = WndProc; // 윈도우 프로시저 이름
    wndclass.cbClsExtra = 0; // 여분 메모리(0바이트)
    wndclass.cbWndExtra = 0; // 여분 메모리(0바이트)
    wndclass.hInstance = hInstance; // 인스턴스 핸들
    wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION); // 아이콘 모양
    wndclass.hCursor = LoadCursor(NULL, IDC_ARROW); // 커서 모양
    wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); // 배경(흰색)
    wndclass.lpszMenuName = NULL; // 메뉴(NULL->메뉴 없음)
    wndclass.lpszClassName = TEXT("HelloClass"); // 윈도우 클래스 이름
```

# HelloSDK 코드 분석 (2)

```
    if(!RegisterClass(&wndclass)) return 1;

    // 윈도우를 생성하고 화면에 보이게 한다.
    hwnd = CreateWindow(TEXT("HelloClass"), TEXT("HelloSDK"), WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
    ShowWindow(hwnd, nCmdShow);

    // 메시지 큐에서 메시지를 하나씩 꺼내서 처리한다.
    while(GetMessage(&msg, NULL, 0, 0) > 0){
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return msg.wParam;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam) {
    HDC hdc;
    PAINTSTRUCT ps;
    TCHAR *str = TEXT("Hello, SDK");
```

# HelloSDK 코드 분석 (3)

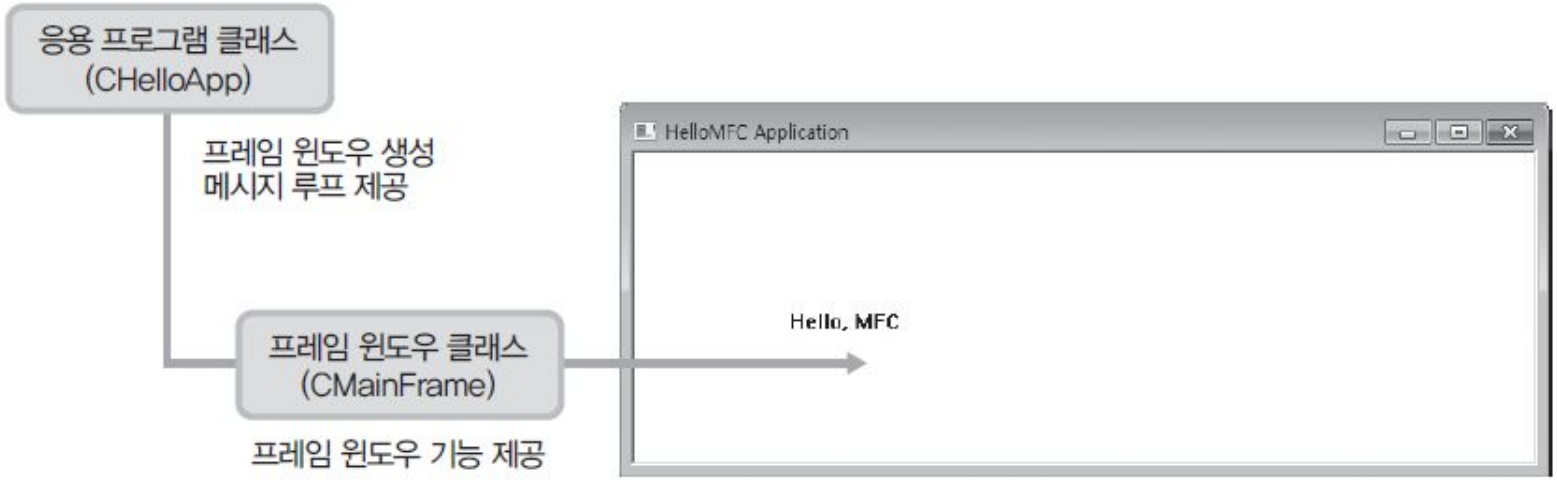
```
    // 발생한 메시지의 종류에 따라 적절히 처리한다.
    switch(message){
    case WM_CREATE:
        return 0;
    case WM_LBUTTONDOWN:
        MessageBox(hwnd, TEXT("마우스 클릭!"), TEXT("마우스 메시지"), MB_OK);
        return 0;
    case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        TextOut(hdc, 100, 100, str, lstrlen(str));
        EndPaint(hwnd, &ps);
        return 0;
    case WM_DESTROY:
        PostQuitMessage(0);
        return 0;
    }

    // 응용 프로그램이 처리하지 않은 메시지는 운영체제가 처리한다.
    return DefWindowProc(hwnd, message, wParam, lParam);
}
```

# 실습 – SDK 프로그램

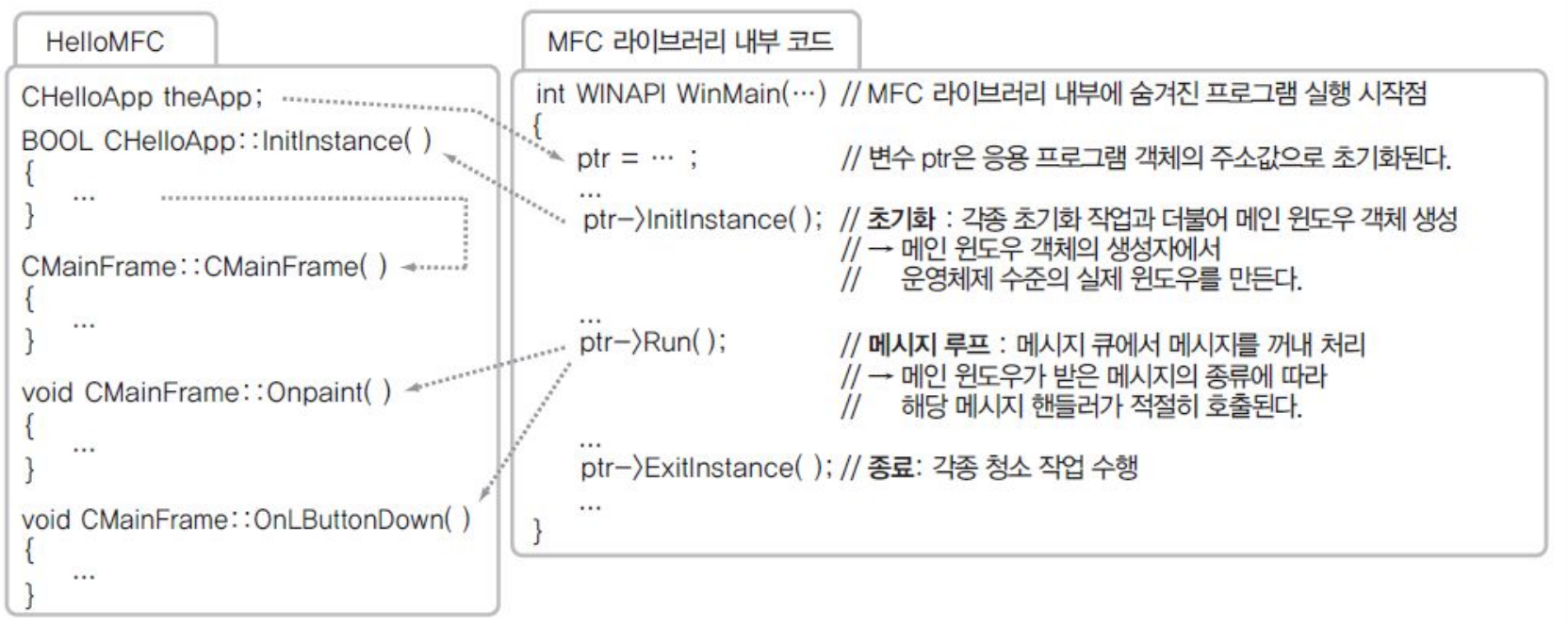
# MFC 프로그램 기본 구조 (1)

- MFC 프로그램의 기본 골격
  1. 응용 프로그램 클래스 정의
  2. 메인(=프레임) 윈도우 클래스 정의
  3. 응용 프로그램 객체 선언
  4. 메시지 맵 선언



# MFC 프로그램 기본 구조 (2)

- MFC 프로그램 동작 원리



# HelloMFC 코드 분석 (1)

```
#include <afxwin.h>

// 응용 프로그램 클래스를 선언한다.
class CHelloApp : public CWinApp
{
public:
    virtual BOOL InitInstance();
};

// 메인 윈도우 클래스를 선언한다.
class CMainFrame : public CFrameWnd
{
public:
    CMainFrame();

protected:
    afx_msg void OnPaint();
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    DECLARE_MESSAGE_MAP()
};
```

# HelloMFC 코드 분석 (2)

```
// 응용 프로그램 객체를 선언한다.
CHelloApp theApp;

// 응용 프로그램 클래스를 정의한다.
BOOL CHelloApp::InitInstance() {
    m_pMainWnd = new CMainFrame;
    m_pMainWnd->ShowWindow(m_nCmdShow);
    return TRUE;
}

// 메인 윈도우 클래스를 정의한다.
CMainFrame::CMainFrame() {
    Create(NULL, _T("HelloMFC"));
}

void CMainFrame::OnPaint() {
    CPaintDC dc(this);
    TCHAR *msg = _T("Hello, MFC");
    dc.TextOut(100, 100, msg, lstrlen(msg));
}
```

# HelloMFC 코드 분석 (3)

```
void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point) {
    MessageBox(_T("마우스 클릭!"), _T("마우스 메시지"));
}

// 메시지 맵을 선언한다.
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_PAINT()
    ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
```

# 실습 – MFC 프로그램

# 학습정리

- 윈도우 운영체제는 그래픽 UI와 Message Driven Architecture를 기반으로 동작하며, 멀티태스킹과 멀티스레딩을 지원합니다.
- 윈도우 운영체제가 응용 프로그램을 위해 제공하는 각종 함수의 집합을 Win32 API라고 합니다.
- SDK 프로그래밍은 Win32 API와 C 언어 기반으로, WinMain 함수와 WndProc 함수로 구성됩니다.
- SDK 프로그래밍은 윈도우 클래스를 정의(초기화)하고 운영체제에 등록 => 윈도우를 생성하고 화면에 보이게 함 => 메시지 루프를 구동 => 윈도우 프로시저에서 메시지를 처리 순으로 기본 코드 골격을 유지합니다.
- MFC 프로그래밍은 MFC 클래스 라이브러리와 C++ 언어 기반의 객체지향 프로그래밍을 합니다.
- MFC 프로그래밍은 응용 프로그램 클래스 정의 => 메인 윈도우 클래스 정의 => 응용 프로그램 객체 선언 => 메시지 맵 선언 순으로 기본 코드 골격을 유지합니다.