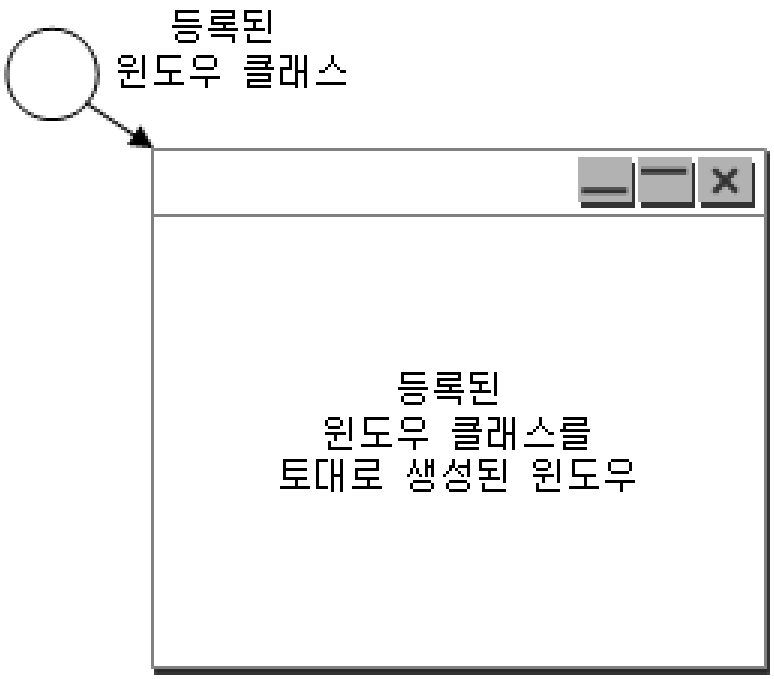
3장
윈도우 생성을 자유롭게

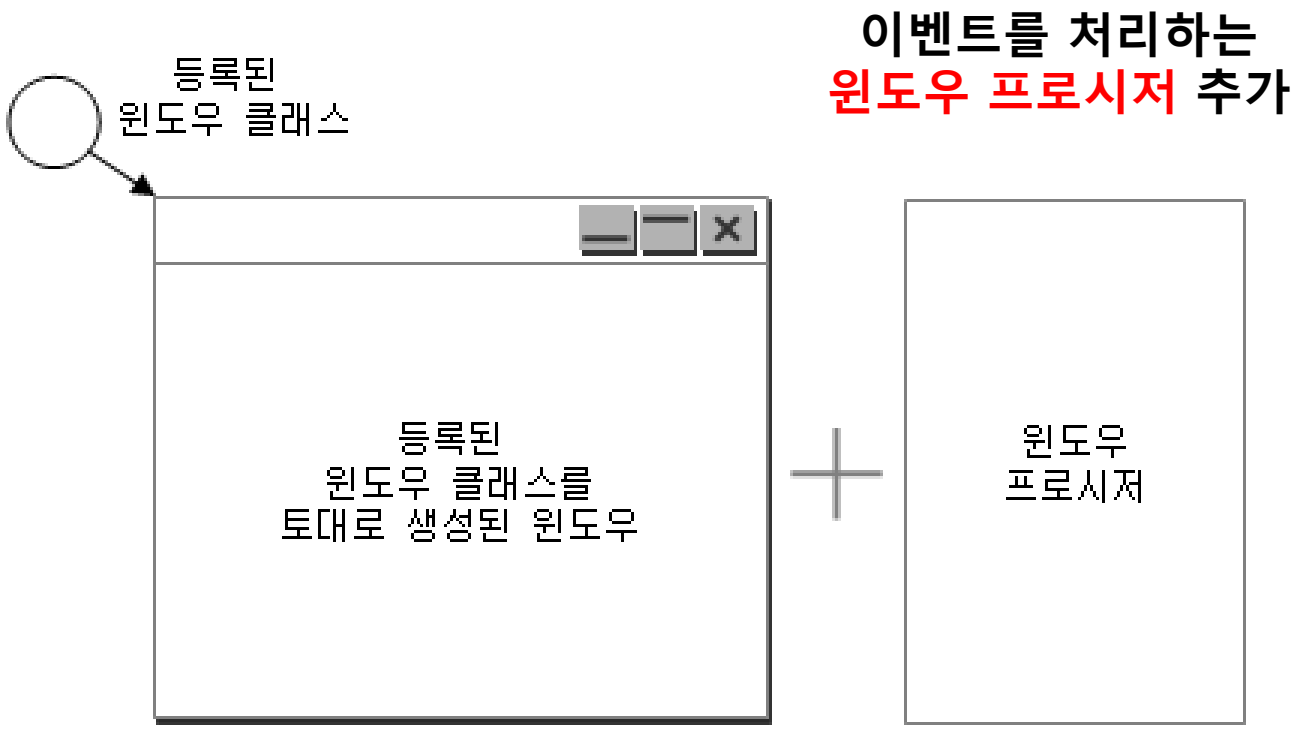
김성영교수
금오공과대학교
컴퓨터공학부

윈도우 프로그램 범주에 대한 이해 (1)



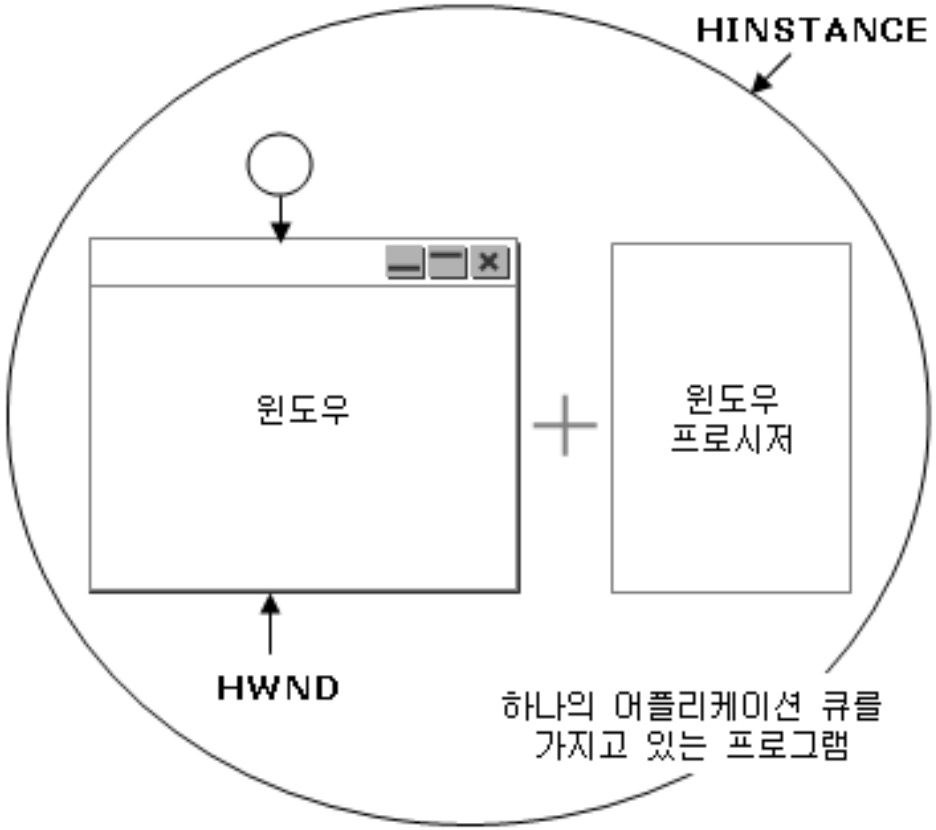
윈도우 클래스와 윈도우

윈도우 프로그램 범주에 대한 이해 (2)

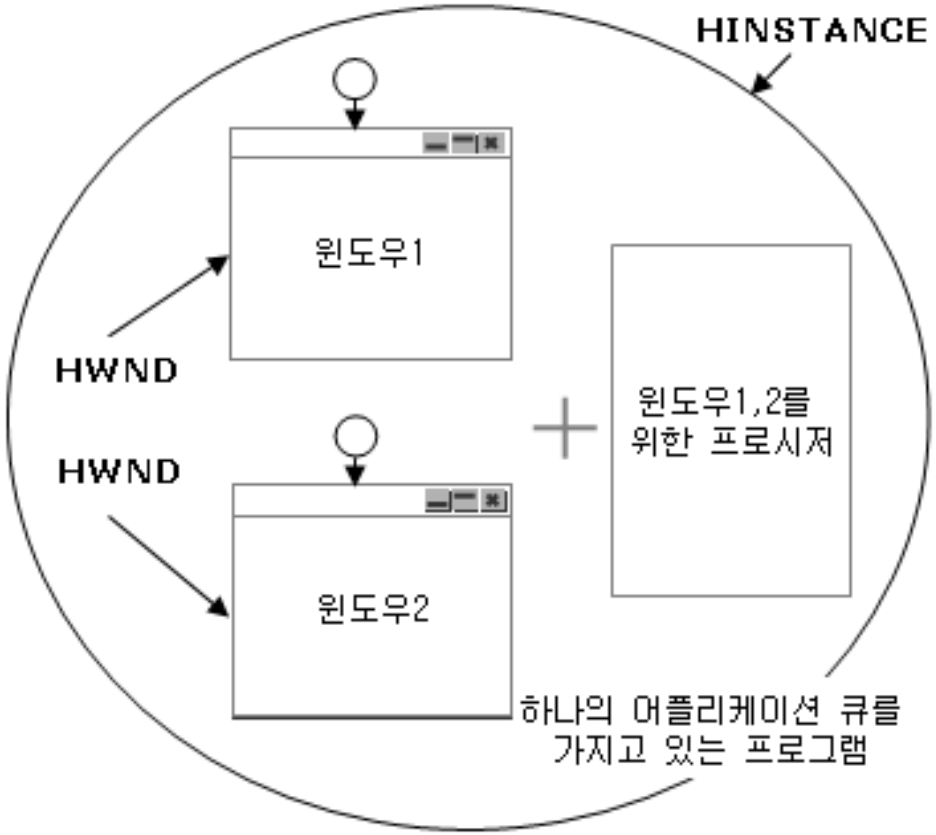


윈도우와 윈도우 프로시저 함수만으로는
윈도우 프로그램이라고 할 수 없음

윈도우 프로그램 범주에 대한 이해 (3)

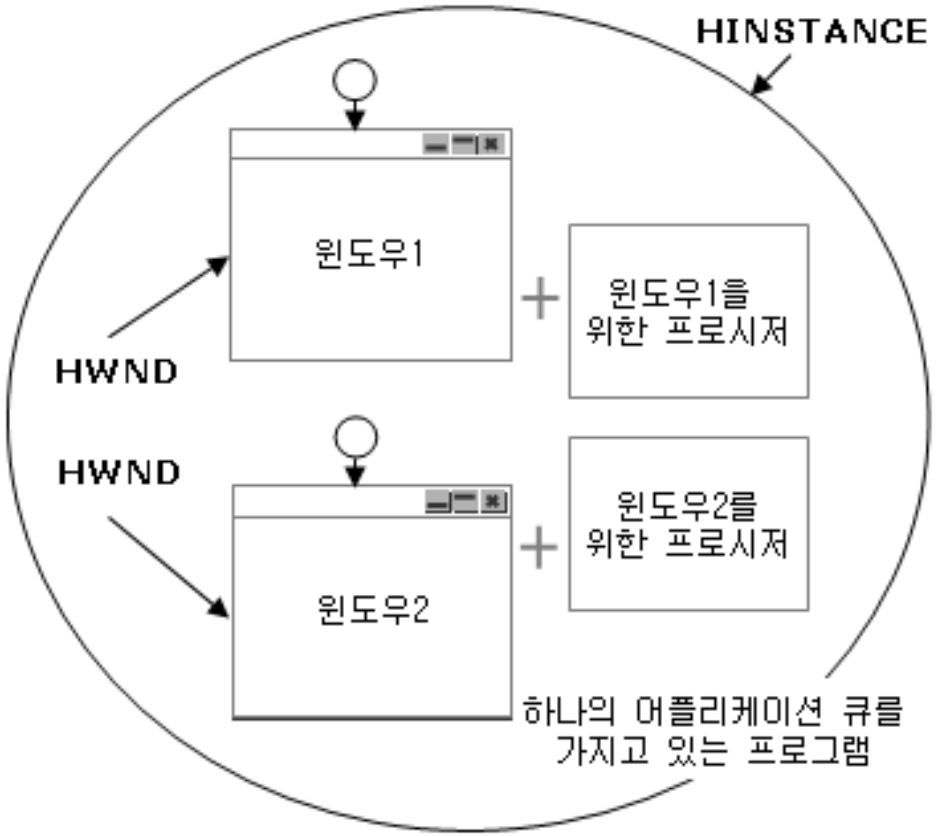


윈도우 프로그램 범주에 대한 이해 (4)



윈도우의 개수를 늘릴 수 있을까?

윈도우 프로그램 범주에 대한 이해 (5)

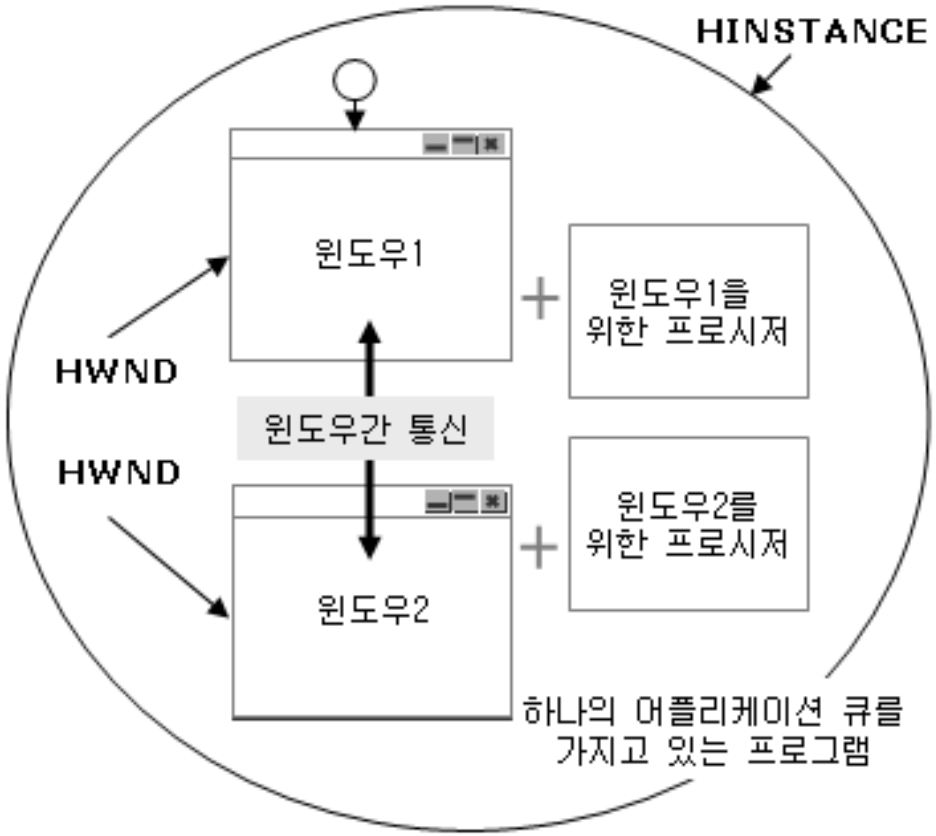


각 윈도우는 발생하는 이벤트 처리하는
독자적인 윈도우 프로시저를 가짐

윈도우 프로그램 범주에 대한 이해 (6)

윈도우가 대략 30개!!
숫자버튼 하나가 하나의
윈도우라고??

윈도우 프로그램 범주에 대한 이해 (7)



윈도우 간의 통신

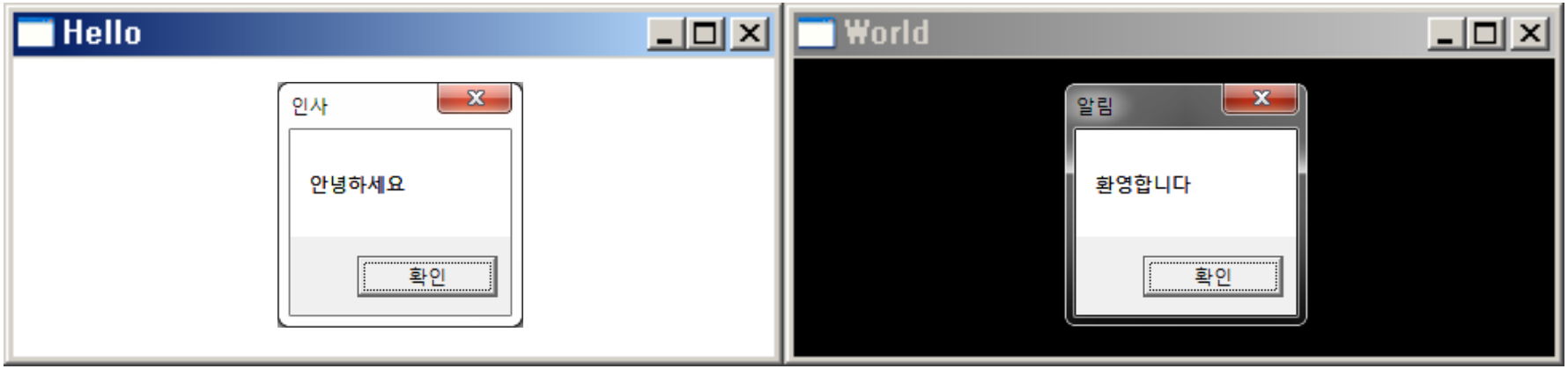
실습 3.1

- 윈도우 두 개를 생성하자.
  - 두 개의 윈도우는 하나의 윈도우 프로시저를 공유하자!!

윈도우를 두 개 만들고
왼쪽버튼 눌림에 대해
메시지 박스 출력

실습 3.2

- 윈도우 두 개를 생성하자.
  - 마우스 왼쪽 버튼 클릭에 대해 각 윈도우는 서로 다른 메시지를 출력하도록 하자!!



World 윈도우의 배경을 검은색으로 변경

실습 3.3

- 윈도우간의 통신을 처리하자 – (1)
  - 흰색 윈도우에서 마우스 왼쪽 버튼을 누르면 검은색 윈도우의 타이틀을 “World”에서 “Black”로 변경하자!!
  - HINT: 검은색 윈도우의 핸들은 전역변수 임

```
BOOL SetWindowText( HWND hWnd, LPCTSTR lpString );

int GetWindowText( HWND hWnd, LPTSTR lpString, int nMaxCount );
```

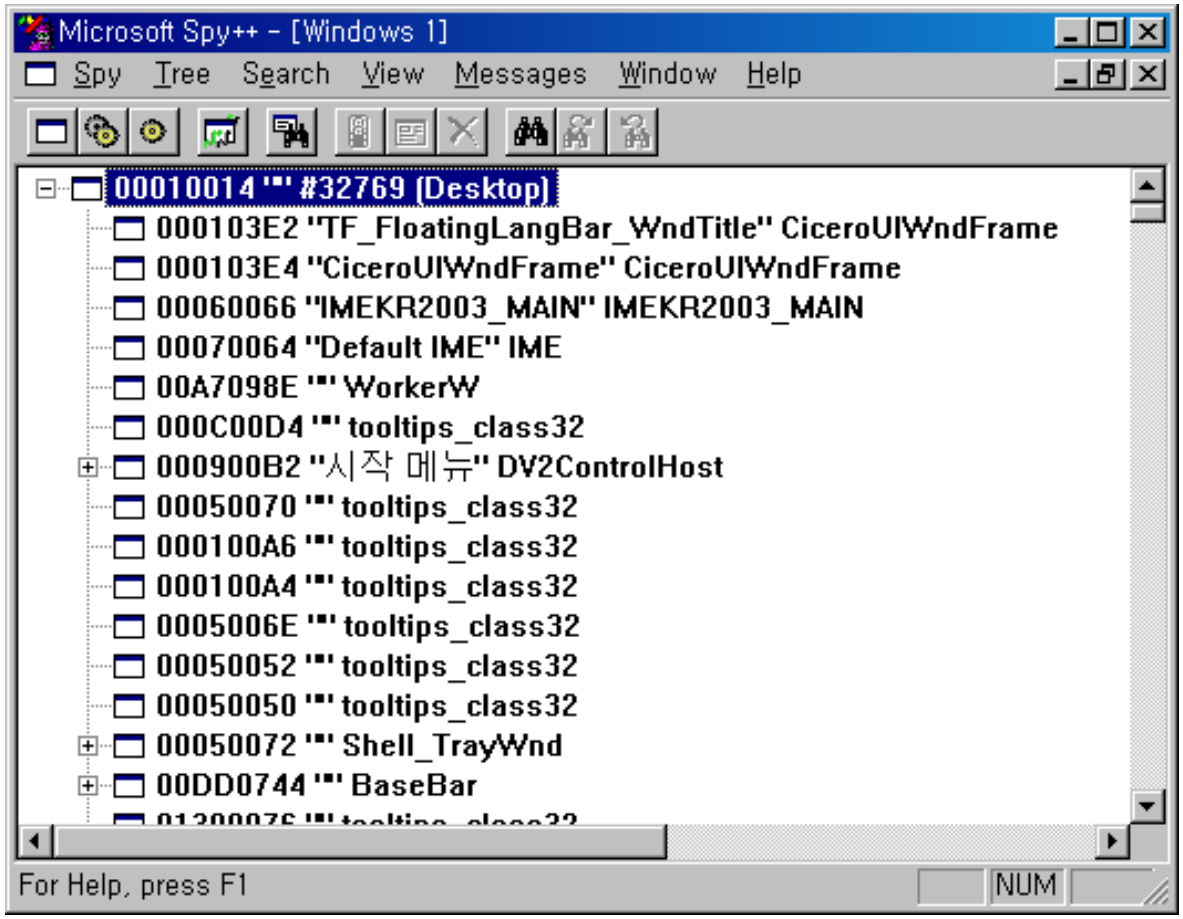
실습 3.4

- 윈도우간의 통신을 처리하자 – (2)
  - 흰색 윈도우에서 마우스 왼쪽 버튼을 누르면 **메모장**에 “Hello”를 출력하자!!
  - HINT: 16진수 사용 및 자료형(HWND) 변환 필요

```
case WM_LBUTTONDOWN:
    HWND hNote;

    hNote = ____________________; //여기에 메모장의 윈도우 핸들 입력

    HDC hdc;
    hdc = GetDC( hNote );
    TextOut( hdc, 0, 0, "Hello", 5 );

    ReleaseDC( hNote, hdc );

    break;
```

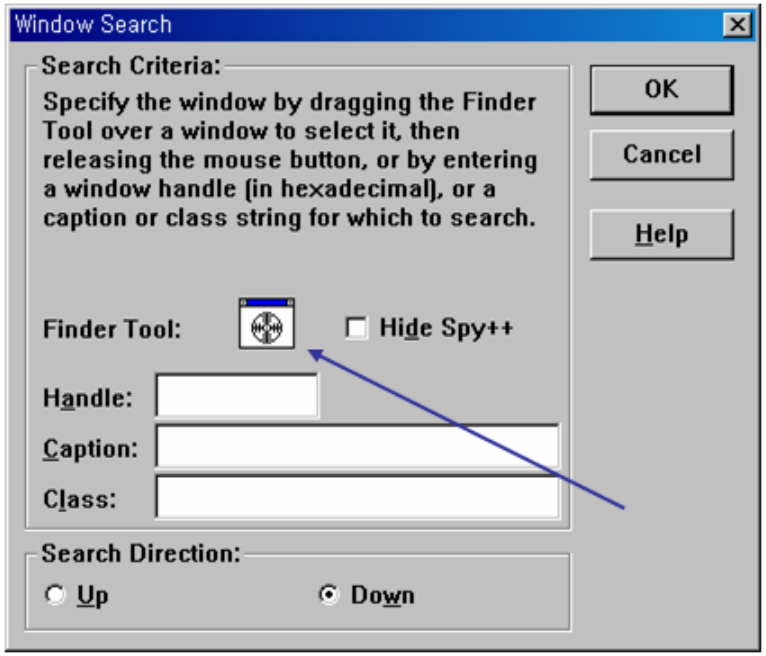
Spy++을 이용한 윈도우 핸들 얻기 (1)

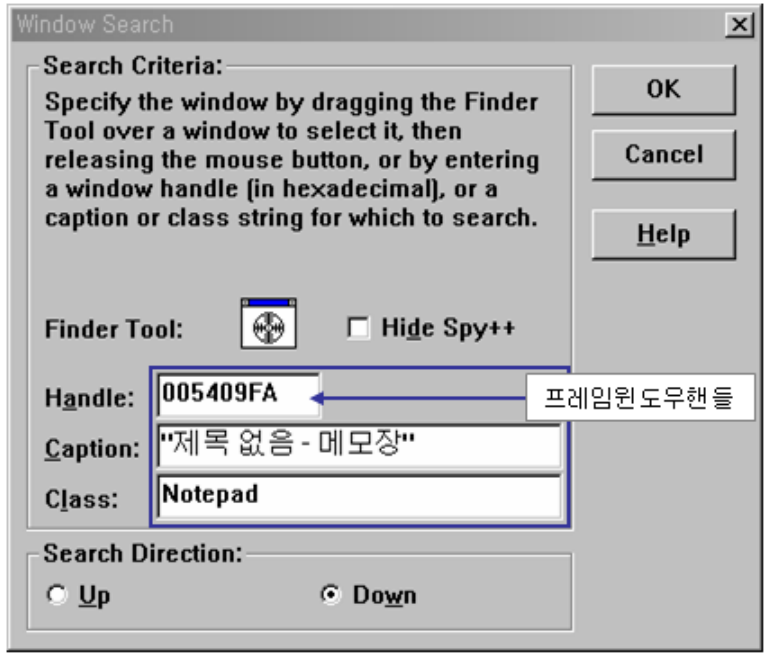
- [검색> 창 찾기...] ([Search>FindWindow...])



Spy++을 이용한 윈도우 핸들 얻기 (2)

- Finder Tool의 아이콘을 원하는 윈도우 내부로 드래그
- 프레임 윈도우와 클라이언트 영역의 구분 필요

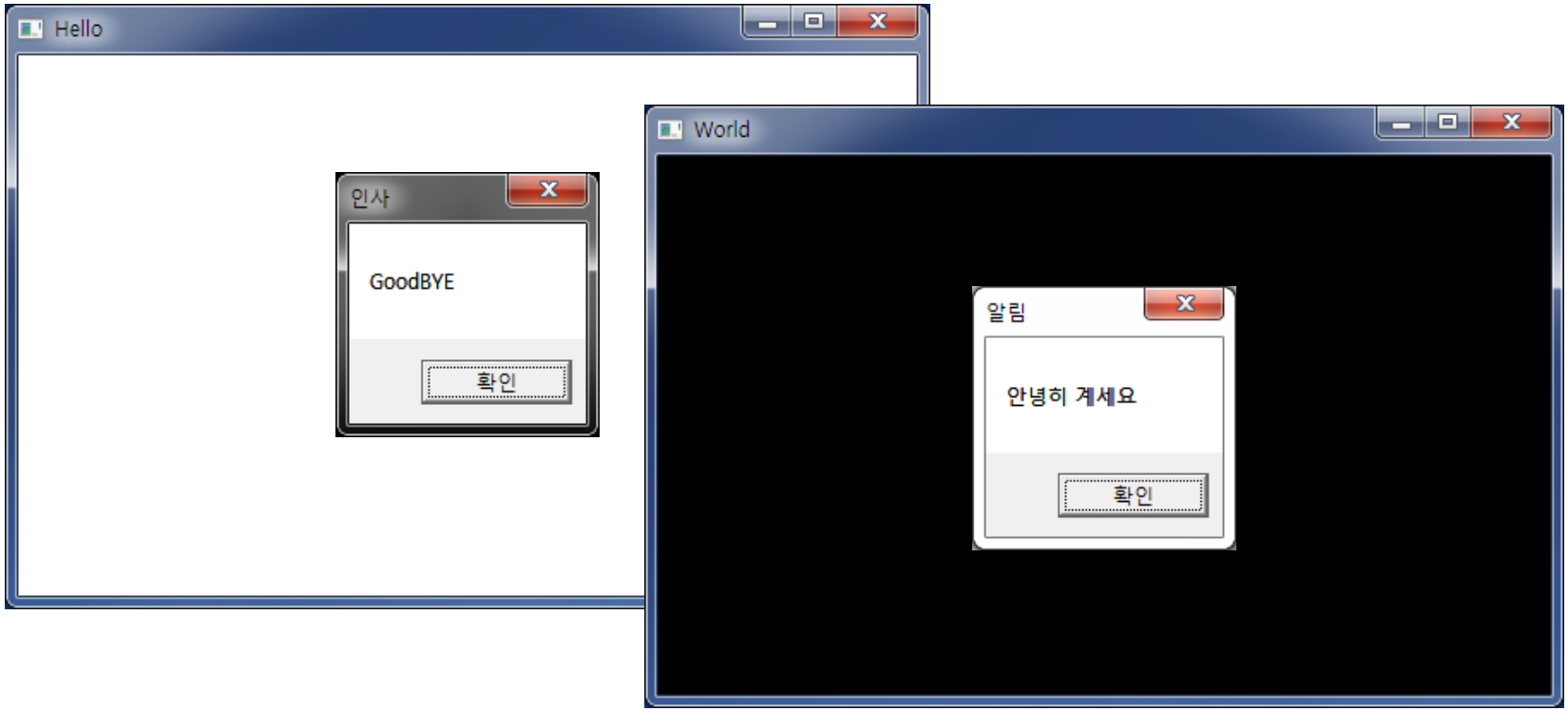




실습 3.5

- 윈도우의 파괴 과정을 살펴보자
  - 윈도우 파괴 단계에서 메시지를 출력하자!!

기대?
결과?



윈도우의 부모-자식 관계 (1)

- 실습 3.5의 문제점
  - 메인 윈도우가 두 개 있는 경우는 드묾
  - 코딩의 편의를 위해 윈도우 핸들을 전역변수로 만들어야 함
  - 부수적으로 사용되는 윈도우의 파괴를 잊어버림

⇒ 윈도우간의 **서열 관계에 대한 설정**이 필요

윈도우의 부모-자식 관계 (2)

- 자식 윈도우를 만들 때 부모 윈도우를 설정

HWND CreateWindow(…, HWND *hParentWnd*, …)
■ 8 번째 인자 (hParentWnd) : 부모 윈도우의 핸들

```
_hwnd2 = CreateWindow(
    "WND2",
    "WORLD",
    WS_OVERLAPPEDWINDOW,
    320, 0, 320, 240,
    hwnd, NULL, hInstance, NULL
);
```

윈도우의 부모-자식 관계 (3)

- 하나의 프로그램에서 두 개 이상의 윈도우를 두는 경우
  - 프로그램 종료처리 윈도우 : 메인 윈도우 (부모 윈도우)
  - 나머지 윈도우 : 자식 윈도우
  - 동일한 부모를 갖는 윈도우: 형제 윈도우

- 자식 윈도우는 **부모 윈도우의 메시지 처리 함수**에서 생성!!
  - 부모 윈도우와 동시에 자식 윈도우를 생성하고자 하는 경우

  ⇒ WM_CREATE 메시지 사용

실습 3.6-1

- 윈도우의 부모-자식 관계를 살펴보자.
  - 자식 윈도우를 부모 윈도우의 메시지 처리 함수에서 생성하자!!
  - 처리 순서
    1. 자식 윈도우 핸들(_hwnd2)은 지역 변수(hwnd2)로 변경함
    2. 인스턴스 핸들을 전역 변수(_hInstance)에 저장함
    3. 자식 윈도우를 부모 윈도우의 메시지 처리 함수에서 생성함
    4. 자식 윈도우 파괴 단계는 생략함
       - DestroyWindow(_hwnd2) 함수를 주석 처리함
    5. 자식 윈도우를 파괴해도 프로그램을 종료하지 않도록 함
       - PostQuitMessage(0) 함수를 주석 처리함

실습 3.6-2

- 윈도우의 부모-자식 관계를 살펴보자.
  - 부모 윈도우에서 자식 윈도우의 타이틀을 “Black”으로 변경하자!!

```
BOOL SetWindowText( HWND hWnd, LPCTSTR lpString );

int GetWindowText( HWND hWnd, LPTSTR lpString, int nMaxCount );
```

윈도우 스타일 변경 (1)

- ShowWindow( ) 함수를 제거

HWND CreateWindow(…, DWORD *dwStyle*, …)
■ 3번째 인자 (dwStyle) : 생성하는 윈도우의 스타일 지정

```
hwnd2 = CreateWindow(
    "WND2",
    "WORLD",
    WS_OVERLAPPEDWINDOW | WS_VISIBLE,
    320, 0, 320, 240,
    hwnd, NULL, _hInstance, NULL
);
```

윈도우 스타일 변경 (2)

- 윈도우 스타일에 옵션 추가 시 OR 연산자 “|” 사용
- 윈도우 스타일에 옵션 제거 시 NEGATION 연산 (~) 과 AND연산 (&)을 함께 사용

```
style = style & (~WS_SYSMENU);
```

윈도우 스타일

- WS_BORDER : 경계를 갖는 윈도우
- WS_CAPTION : 타이틀 바를 갖는 윈도우
- WS_OVERLAPPED : 타이틀 바와 경계를 갖는 윈도우
- WS_CHILD : 자식 윈도우 속성의 윈도우 (WS_POPUP과 함께 사용 불가)
- WS_POPUP : POPUP 윈도우
- WS_MAXIMIZE : 최대화 버튼을 갖는 윈도우
- WS_MINIMIZE : 최소화 버튼을 갖는 윈도우
- WS_SYSMENU : 타이틀 바에 시스템 메뉴를 갖는 윈도우
- WS_THICKFRAME : 크기 조정이 가능한 윈도우
- WS_OVERLAPPEDWINDOW : WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_THICKFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX
- WS_POPUPWINDOW : WS_POPUP | WS_BORDER | WS_SYSMENU
- WS_CHILDWINDOW : WS_CHILD

실습 3.7

- 윈도우 스타일을 변경하자.
  - 자식 윈도우 생성시에 윈도우 스타일로 WS_VISIBLE과 WS_CHILD를 추가하자!!
  - 자식 윈도우의 시스템 메뉴를 제거하자!!