C# Windows Forms

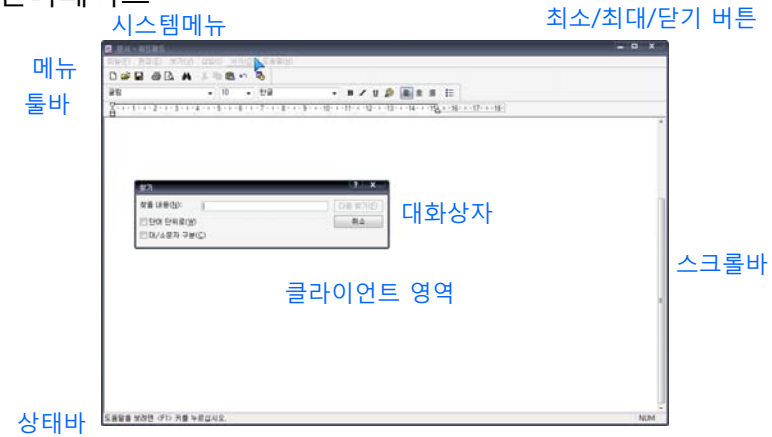
321190
2013년 가을학기
11/05/2013
박경신

Overview

- Windows OS & Windows Applications 특징 이해
- Windows Form
  - Form 클래스
  - InitializeComponent()
  - Application.Run()
  - Form 속성 (Property) - 모양변경, 위치 및 크기, 초기상태
  - Form Modality
  - MDI (Multiple Document Interface)

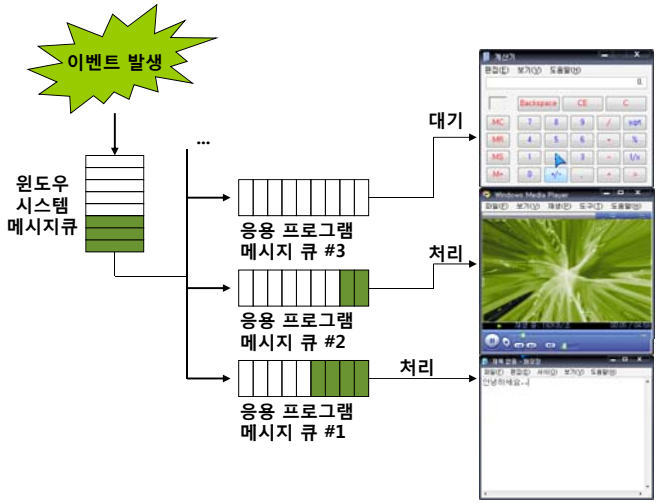
Windows OS 특징

- Graphical User Interface (GUI) – 일관성 있는 사용자 인터페이스



Windows OS 특징

- 메시지기반 구조 (Message-driven Architecture)



Windows OS 특징

- 멀티 태스킹 (Multi-Tasking)
  - 하나의 윈도우즈 시스템에서 여러 개의 응용 프로그램을 수행
  - 응용프로그램 사이의 상호작용 가능
- 멀티 스레딩 (Multi-Threading)
  - 하나의 응용 프로그램에 여러 개의 실행 흐름을 생성
- 장치에 독립적
  - 장치 드라이버 (Device Driver)에 의해 주변 장치들을 제어

Windows Application 특징

- API 호출문 집합
  - API(Application Programming Interface) - 윈도우 운영체제가 응용프로그램을 위해 제공하는 각종 함수 집합

응용 프로그램

```
call API#1
call API#2
...
call API#3
call API#4
...
call API#n
```


Windows Application 특징

- 메시지 핸들러 (Message Handler) 집합
  - 메시지 핸들러 (Message Handler) - 메시지를 받았을 때 동작을 결정하는 코드 - 키보드, 마우스, 메뉴, 등등
  - 윈도우 프로시저 (Window Procedure) – 이러한 메시지 핸들러의 집합

응용 프로그램

```
...
메시지 핸들러 #1
메시지 핸들러 #2
메시지 핸들러 #3
메시지 핸들러 #4
메시지 핸들러 #5
메시지 핸들러 #6
...
```

} 윈도우 프로시저: 메시지 핸들러 집합

Windows Application 특징

- 실행 파일과 DLL 집합
  - DLL(Dynamic-Link Library): 프로그램이 실행 중에 호출할 수 있는 함수(코드)와 리소스

응용 프로그램

```
실행 파일
DLL #1
DLL #2
DLL #3
DLL #4
DLL #5
...
```

Windows Application 특징

- 장치 독립적
  - 윈도우 시스템의 API를 사용하여 간접적으로 주변장치들을 제어

응용 프로그램 ↔ API ↔ 장치 드라이버 ↔ 주변 장치

Windows Forms

- Windows Forms
  - 응용프로그램에 공통적으로 들어가는 기능을 사용자 인터페이스 컴포넌트 형태로 미리 구현해 놓은 .NET 클래스
  - Windows 응용 프로그램의 사용자 인터페이스를 구성하는 창 또는 대화 상자(응용 프로그램에 표시되는 모든 창), 다중 문서 인터페이스 등
  - VC++ MFC와 비슷함
  - .NET 프레임워크에서 지원되므로 .NET 언어에서 모두 사용 가능

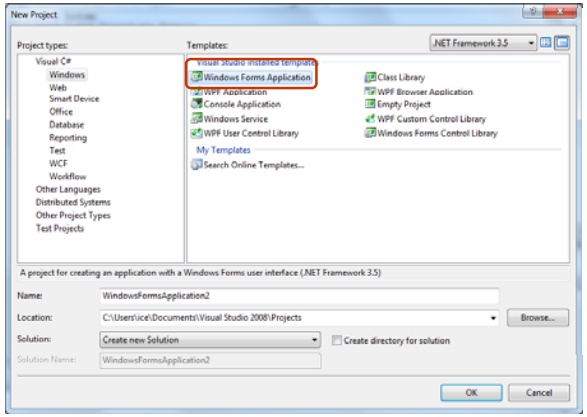
Windows Forms

- 윈도우 폼 응용프로그램 생성시 대부분의 기능 System.Windows.Forms 네임스페이스에 포함
  - 기본적인 윈도우 폼 응용 프로그램 (Windows Forms Application)

```
using System;
using System.Windows.Forms;
static class Program  {
    // 프로그램 시작점
    static void Main() {
        Application.EnableVisualStyle();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());      // 폼 객체생성 및 메시지처리
    }
}
```

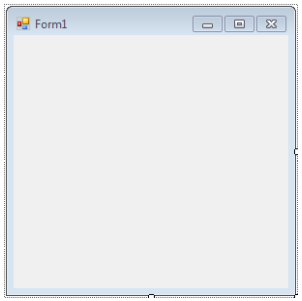
Windows Forms Application

- IDE의 시작페이지에서 Windows Form Application 새 프로젝트를 만들면 자동 생성
  - 새 프로젝트 대화상자



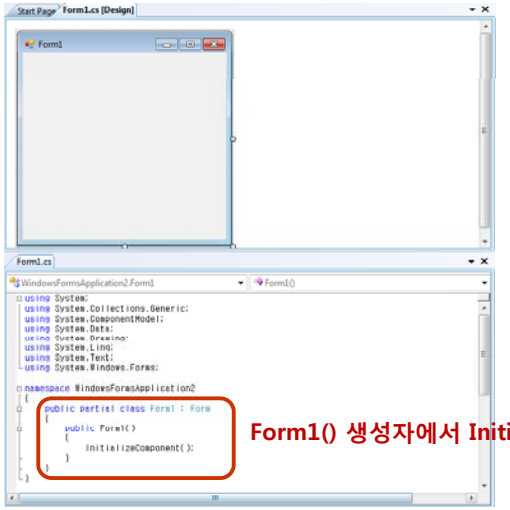
Windows Forms Application

- 기본 폼
  - 폼은 CLR의 **System.Windows.Forms.Form** 클래스에서 지원
  - 프로젝트 생성시 자동으로 작성된 **Form1.cs** 파일 생성



Windows Forms Application

- Form1.cs의 디자인 템플릿과 코드보기



Form1() 생성자에서 InitializeComponent() 호출

Windows Forms Application

- **Partial 클래스**
  - partial 키워드를 사용하여 **클래스**나 **구조체**의 정의 또는 **인터페이스**를 두 개 이상의 소스 파일로 분할가능
  - 대규모 프로젝트를 진행하는 경우 클래스를 개별 파일로 분할하면 여러 프로그래머가 동시에 작업을 수행할 수 있음
  - 자동으로 생성된 소스를 사용하여 작업하는 경우 소스 파일을 다시 만들지 않고도 클래스에 코드를 추가가능

```
public partial class Employee {
        public void DoWork() { .... }
}
public partial class Employee {
        public void GoToLunch() { .... }
}
```

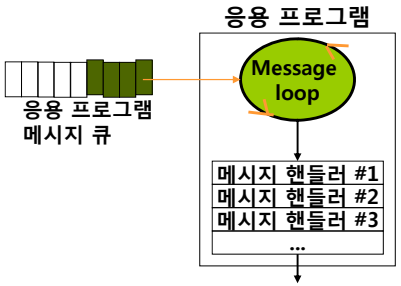
```
partial class Earth : Planet, IRotate {  .... }
partial class Earth : IRevolve { ... }
```

Windows Forms Application

- **InitializeComponent()**
  - 윈도우 폼을 응용프로그램에 연결해주는 코드가 포함
  - 윈도우 폼과 이벤트 핸들러를 연결하는 코드가 포함 – **System.Windows.Forms.Application.Run** 메소드가 이벤트 처리 스레드를 만드는 역할을 함
  - 폼 클래스의 생성자가 딱 한번 호출

Windows Forms Application

- **Application.Run(Form)**
  - 현재 스레드에서 표준 응용프로그램 메시지 루프의 실행을 시작하고 지정된 폼을 표시
  - 폼이 닫힐 때, 메시지 루프에 종료 메시지를 보냄
  - **Message Loop**
    - 메시지 큐에서 메시지를 하나씩 꺼내어 처리하기 위한 반복문
    - 응용프로그램이 종료될 때까지 반복

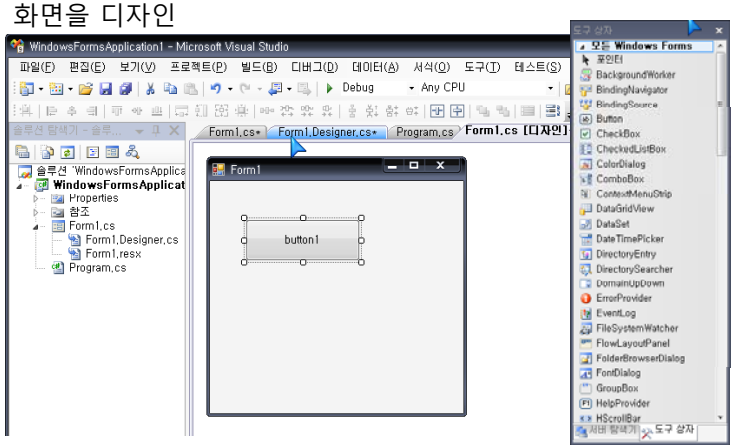


Windows Forms Application

- **Application 클래스**
  - 윈도우 응용프로그램 그 자체를 나타내는 클래스
  - 공개된 생성자가 없기 때문에 객체를 생성할 수 없으며 봉인되어 있어 상속도 할 수 없음
  - **윈도우 응용프로그램을 관리하는 정적 메소드의 집합**
  - Application 클래스는 윈도우 응용프로그램의 시작과 종료, 메시지 처리, 여러 가지 정보조사 등의 작업을 하는 정적 메소드와 속성들을 제공

Windows Forms Application

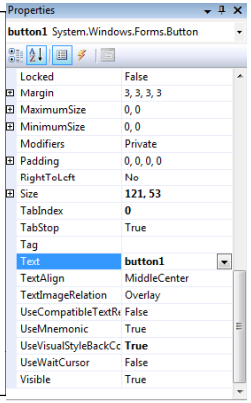
- 윈도우 폼에서 컨트롤 사용
  - 개발자는 도구상자에서 각종 컨트롤 (예: 버튼, 텍스트박스, 리스트뷰 등)을 Drag-and-Drop하여 원하는 곳에 배치하여 화면을 디자인



Windows Forms Application

- 윈도우 폼에서 컨트롤 사용
  - 그리고, 속성 (Property) 창에서 속성을 변경하면 자동으로 **InitializeComponent()** 메소드에 업데이트

```
private void InitializeComponent() {
  this.button1 = new System.Windows.Forms.Button();
  this.label1 = new System.Windows.Forms.Label();
  this.SuspendLayout();
  //
  // button1
  //
  this.button1.Location = new System.Drawing.Point(34, 43);
  this.button1.Name = "button1";
  this.button1.Size = new System.Drawing.Size(121, 53);
  this.button1.TabIndex = 0;
  this.button1.Text = "button1";
  this.button1.UseVisualStyleBackColor = true;
   // 중간생략..
}
```



Form Class

- 폼 클래스
  - 윈도우 응용프로그램의 움직임을 제어하는 멤버들을 정의해 놓은 하위 수준의 클래스
  - 응용프로그램을 종료하거나 휴식(Idle)상태를 처리하는 등의 응용프로그램의 행동을 처리하는 많은 이벤트를 정의
  - 응용프로그램 클래스의 정적 메소드, 속성, 이벤트
  - 응용프로그램 클래스로 할 수 있는 기능
    - 응용 프로그램의 정보를 제공, 응용 프로그램이 종료할 때의 이벤트를 처리, 메시지 처리

Form Class

- 폼 클래스 상속계층 구조

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Windows.Forms.Control
        System.Windows.Forms.ScrollableControl
          System.Windows.Forms.ContainerControl
            System.Windows.Forms.Form

  - 폼 클래스 자체로서의 의미뿐만 아니라, Custom Form 클래스의 직접적인 기본 클래스로서의 의미도 가짐

Form Class

- 폼 클래스는 대부분의 특성을 상속 받아 **재정의하는 메소드** 제공

메소드	설명
Activate()	주어진 폼을 활성화시키시고, 그 포커스를 폼에 맞춤
Close()	폼을 닫고 폼과 관련된 컨트롤과 컴포넌트에 할당되었던 모드 리소스를 해제
CenterToScreen()	폼을 스크린 부동 중심점에 위치
LayoutMDI()	부모 폼 안의 자식 폼들을 모두 정렬
OnResize()	Resize 이벤트에 반응
ShowDialog()	폼을 Modal Dialog로서 나타냄
Show()	숨겨진(Hide) 폼을 화면 상에 표시
Hide()	단지 폼을 화면 상에 보이지 않게 함

Form Class

- 폼 클래스의 **속성 (Property)**
  - **Name** : 폼의 이름
  - BackColor : 배경색 (색상 이름 또는 빨강(Red), 녹색(Green), 파랑(Blue)의 RGB 값으로 지정)
    form1.BackColor = System.Drawing.Color.DarkBlue;
  - BackgroundImage : 폼의 배경그림
  - Icon : 폼 아이콘
  - Opacity : 폼의 투명도
  - 폼의 버튼 사용 여부
    form1.MinimizeBox = false;

속성	값	설명
MinimizeBox	True/False	최소화 버튼 사용여부
MaximizeBox	True/False	최대화 버튼 사용 여부
ControlBox	True/False	최소,최대,닫기 버튼 사용 여부

Form Class

- FormBrderStyle : 폼의 테두리 모양
  form1.FormBorderStyle = FormBorderStyle .FixedSingle;

값	설명
Fixed3D	크기가 고정된 3차원 테두리
FixedDialog	크기가 고정된 대화상자 스타일의 굵은 테두리
FixedSingle	크기가 고정된 단일 선 테두리
Sizable	크기를 조정할 수 있는 테두리
None	테두리가 없는 창
FixedToolWindow	크기가 고정된 도구 창
SizableToolWindow	크기를 조정할 수 있는 도구 창

- Size : 폼의 크기 설정
  form1.Size = new System.Drawing.Size(100, 100);
  폭 : Form1.Width = 200
  높이 : Form1.Height = 200

Form Class

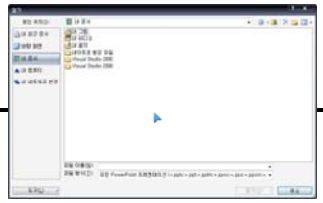
- Location : 폼의 위치설정
  form1.Location = new Point(100, 100);
  X좌표 : Form1.Left = 200;
  Y좌표 : Form1.Top = 200;
- StartPosition : 초기 위치설정
  form1.StartPosition = FormStartPosition.Manual;

값	설명
Manual	디자인 때의 location 속성 값으로 설정된 위치에 표시
CenterScreen	화면 중앙 위치에 표시
WindowsDefaultLocation	Windows 기본 위치인 화면의 좌측 상단에 표시
WindowsDefaultBounds	Windows 기본 테두리 설정 값의 테두리를 포함한 윈도우 기본위치에 표시
CenterParent	부모 폼의 테두리 내에서 중앙에 위치

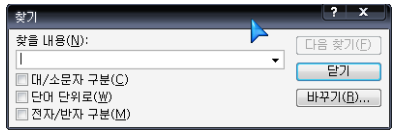
Form Class

- WindowState : 폼의 표시 상태 설정 (Normal : 보통 상태로 변경, Minimized : 최소화 상태로 변경, Maximized : 최대화 상태로 변경)
  form1.WindowState = FormWindowState.Normal;
- ShowInTaskbar : Windows 작업표시줄에 표시여부 설정, 작업표시줄로 프로그램 전환 불가능, <Alt+Tab>을 사용하여 프로그램 전환 가능
  form1.ShowInTaskbar = false;
- TopMost : 최상위 창으로 설정
  form1.TopMost = true;
- Visible : 화면에 폼 표시 여부 설정
  form1.Visible = true;

Form Class



- 모달리티 (Modality)
  - **Modal Form**
    - **모달 폼이 표시되는 동안 제어권을 독점하고 있는 형태**
    - 사용자로부터 입력 받은 정보로 다음 진행상태가 결정되는 경우에 사용
    - **ShowDialog() 메소드를 사용**
    - 예 : 열기, 저장, 옵션 창
  - **Modeless Form**
    - **모달이 아닌 폼이 표시되는 동안 다른 폼으로의 제어이동이 가능한 형태**
    - 서로 다른 창 사이에서의 정보교환을 필요로 하는 경우에 사용
    - **Show() 메소드를 사용**
    - 예 : "찾기 및 바꾸기" 창



MDI(Multiple Document Interface)

- MDI (Multiple Document Interface)
  - 하나의 부모 창(*컨테이너*)과 여러 개의 자식 창으로 구성
- 부모창 (Parent Form)
  - 다른 창을 영역 내에 포함시키는 창
  - IsMdiContainer 속성값을 true로 설정
    form1.IsMdiContainer = true;
  - 활성화된 자식창 확인
    form activeChild = this.ActiveMdiChild;
- 자식창 (Child Form)
  - 부모창의 영역 내에 존재하는 창
  - MdiParent 속성값에 부모 객체 설정
    form2.MdiParent = form1;

MDI(Multiple Document Interface)

- LayoutMdi 속성을 이용하여 자식폼 정렬

Form1.LayoutMdi(System.Windows.Forms.MdiLayout.Cascade);

멤버 이름	설명
ArrangeIcons	모든 MDI 자식 아이콘은 MDI 부모 폼의 클라이언트 영역 내에서 정렬
Cascade	모든 MDI 자식 창은 MDI 부모 폼의 클라이언트 영역 내에서 단식으로 정렬
TileHorizontal	모든 MDI 자식 창은 MDI 부모 폼의 클라이언트 영역 내에서 Tile Horizontal식으로 정렬
TileVertical	모든 MDI 자식 창은 MDI 부모 폼의 클라이언트 영역 내에서 Tile Vertical식으로 정렬

- MdiChildren 속성을 사용하여 모든 자식폼을 배열로 얻기

```
Form[] forms = form1.MdiChildren
foreach (Form f in forms) {
        ...
}
```

Control Class

- Visual Studio.NET의 대부분의 컨트롤들은 **System.Windows.Forms.Control** 클래스로부터 파생
  - 모든 컨트롤들은 컨트롤의 모습이나 행동을 결정하는 여러 속성을 가짐
  - 기반 클래스인 Control 클래스로부터 상속, 재정의

Control Class

- 컨트롤 클래스의 공통 속성(Property)

공통 속성	설명
BackColor	컨트롤의 배경색
BackgroundImage	컨트롤의 배경 이미지
Enabled	컨트롤의 활성화 여부 (비활성화된 컨트롤은 화면엔 표시되지만 동작되지 않음)
Focused	초점을 갖고 있는지 여부 (초점을 갖고 있는 컨트롤만 마우스나 키에 반응함)
Font	컨트롤 텍스트의 폰트
ForeColor	컨트롤의 전경색 (일반적으로 글자의 색깔을 의미)
TabIndex	탭 순서 (탭 키를 누를 때 초점이 이동하는 순서)
TabStop	탭 키를 누를 때 초점을 받을 지를 지정
Text	컨트롤과 관련된 텍스트
Visible	컨트롤의 가시성 지정 (true로 지정된 컨트롤만 화면에 표시)

Control Class

- 컨트롤 클래스의 공통 메소드(Method)

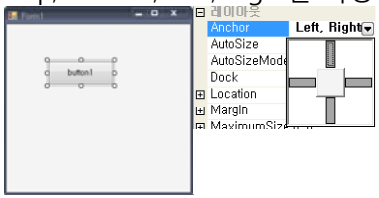
공통 메소드	설명
Focus()	입력 초점을 설정
Hide()	컨트롤을 숨김
Show()	컨트롤을 표시

- 컨트롤 클래스의 GUI관련 디자인 속성

속성	설명
Anchor	폼의 크기 변경 시 고정 거리를 유지해야 할 측면
Dock	컨트롤의 도킹 형태 지정
Padding	측면과 도킹 컨트롤 사이의 공간 지정 (default: 0)
Location	폼의 좌측 상단을 모서리를 기준으로 한 컨트롤의 좌표
Size	컨트롤의 픽셀 단위의 크기
MinimumSize MaximumSize	컨트롤의 최대, 최소 크기

Control Class

- 앵커
  - 폼 안의 컨트롤 들이 그려지는 위치를 고정
  - 예: 앵커를 Top, Bottom, Left, Right를 지정한 경우



- 도킹
  - 컨트롤들을 폼의 치수에 상관없이 폼의 지정된 가장자리를 모두 차지
  - 예: 도킹을 Right를 지정한 경우

Control Class

- 이벤트
  - Key press 또는 mouse click 등 사용자 조작이 발생할 경우의 행동
- 버튼 클릭 이벤트
  - 사용자가 버튼 위에 마우스 포인터를 놓고 왼쪽 마우스 버튼을 눌렀다가, 포인터가 버튼 위에 있는 동안 떼면 발생 (버튼이 포커스를 가진 경우 엔터키를 눌러도 발생)
  - 핸들러 추가 시, InitializeComponent()에 이벤트 핸들러 등록을 위한 대리자 코드 추가

```
this.button1.Click += new System.EventHandler(this.button1_Click);
```

  - 이벤트 핸들러로 사용될 메소드 자체 추가
    - 메소드명은 컨트롤 이름에 밑줄이 붙고, 뒤에 이벤트 명시
    - 버튼을 클릭하였을 경우 폼의 타이틀 변화

```
private void button1_Click(object sender, System.EventArgs e) {
    this.label1.Text = "버튼이 클릭되었습니다."; // 레이블의 텍스트
}
```

Control Class

- 컨트롤의 마우스 이벤트 발생 순서
  1. **MouseEnter : 마우스 포인터가 컨트롤에 들어가면 발생**
  2. **MouseMove : 마우스 포인터를 컨트롤 위로 이동하면 발생**
  3. **MouseHover : 마우스 포인터가 컨트롤 위에 있을 때 발생**
  4. **MouseDown : 마우스 포인터가 컨트롤 위에 있을 때 마우스 단추를 클릭하면 발생**
  5. **MouseWheel : 컨트롤에 포커스가 있을 때 마우스 휠을 움직이면 발생**
  6. **MouseUp : 마우스 포인터가 컨트롤 위에 있을 때 마우스 단추를 눌렀다 놓으면 발생**
  7. **MouseLeave : 마우스 포인터가 컨트롤을 벗어나면 발생**

```
//버튼이 눌렸을 때
private void button1_Click(object sender, EventArgs e){
        Form1 f = new Form1();
        f.ShowDialog();
}
//버튼이 눌렸을 때
private void button2_Click(object sender, EventArgs e){
        Control p = button2.Parent;
        MessageBox.Show("Parent Type = " + p.GetType() + ",
                        Parent Text = " + p.Text, "button2_Click");
 }
//마우스포인터가 컨트롤 내에 들어올 때 발생되는 이벤트 처리
private void button3_MouseEnter(object sender, EventArgs e)   {
        Button b = (Button)sender;
        prevColor = button3.BackColor;
        b.BackColor = System .Drawing .Color .AliceBlue;
}
//마우스포인터가 컨트롤을 벗어날 때 발생되는 이벤트 처리
private void button3_MouseLeave(object sender, EventArgs e)    {
        Button b = (Button)sender;
        b.BackColor = prevColor ;
}
```