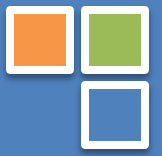
C# 입문 : 이론과 실습



제 10장
메뉴/마우스/키보드 다루기

Programming Language Lab
Dongguk University

목차

- 메뉴 다루기
- 마우스 다루기
- 키보드 다루기

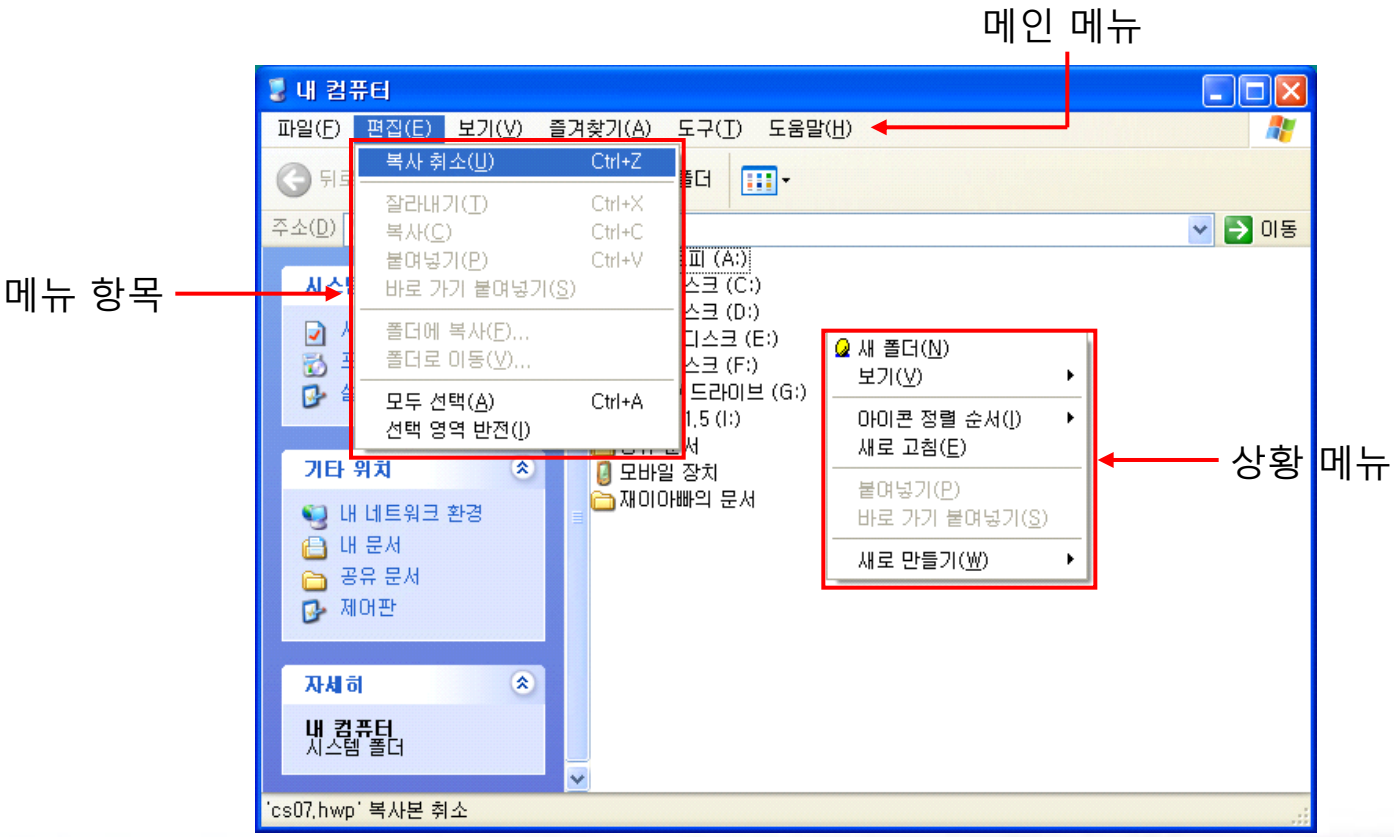
메뉴 다루기 [1/2]

- 메뉴
  - 윈폼 애플리케이션에서 가장 일반적인 사용자 인터페이스
  - 윈폼 애플리케이션이 제공하는 기능을 사용자가 쉽게 이해하고 사용할 수 있도록 도와주는 기능

- 메뉴의 종류
  - 메인메뉴 (main menu)
    - 폼의 상단에 배치되는 주요 메뉴
  - 상황메뉴 (context menu)
    - 마우스 오른쪽 버튼을 클릭했을 때 나타나는 팝업 메뉴

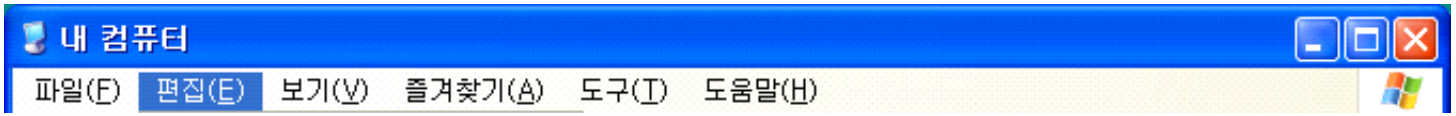
메뉴 다루기 [2/2]

- 메뉴의 구성



메인 메뉴

- 폼의 상단에 배치되는 메뉴

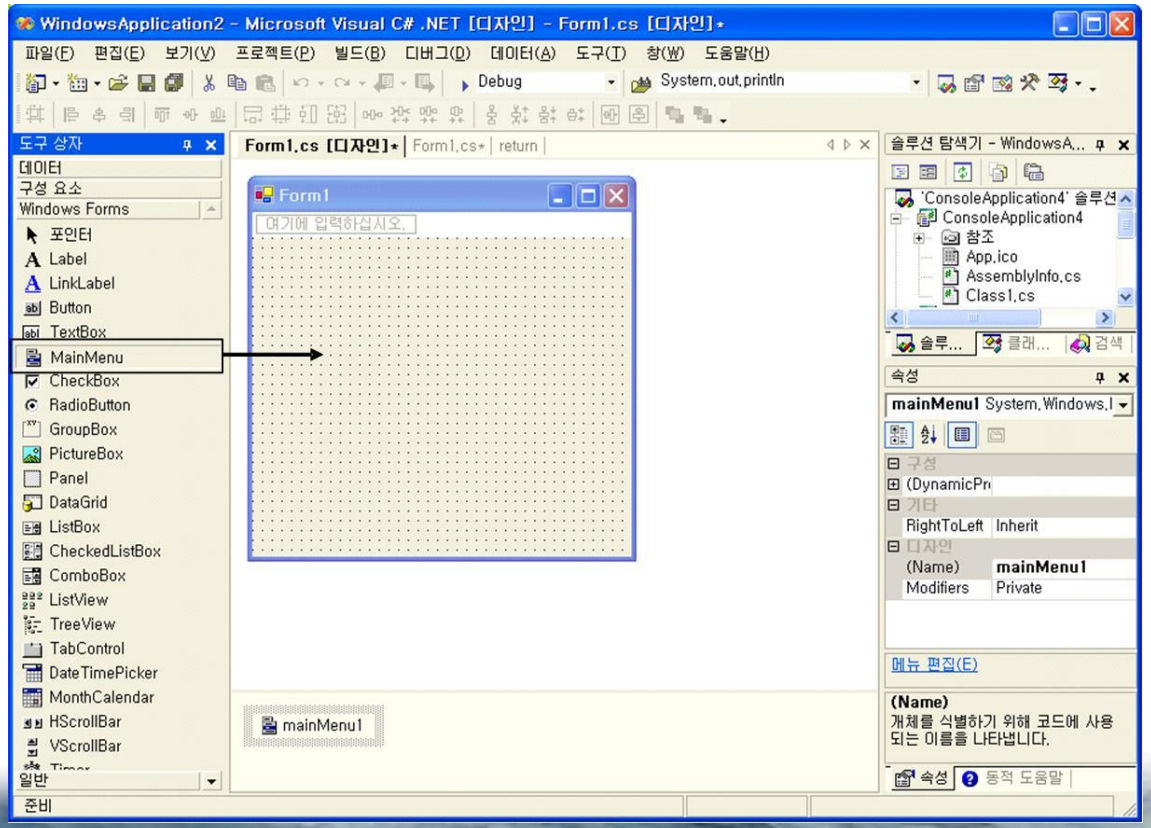


- 마우스 클릭뿐만 아니라 단축키를 통해서는 접근할 수 있는 가장 기본적인 사용자 인터페이스

- 통합 개발 환경의 MainMenu 컴포넌트를 통하여 작성

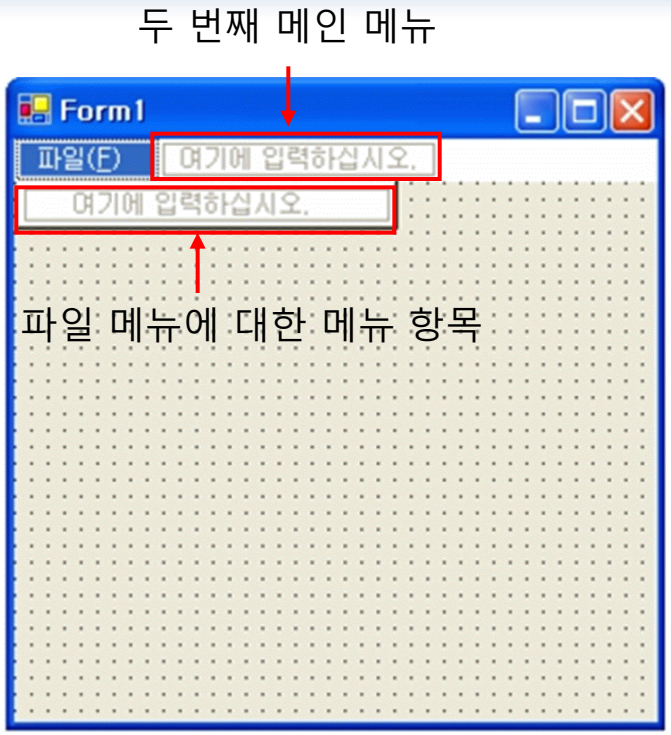
메인 메뉴의 작성 [1/5]

- MainMenu 컴포넌트의 추가
  - 【도구상자】➡【MainMenu】를 선택하여 폼 디자이너로 이동



메인 메뉴의 작성 [2/5]

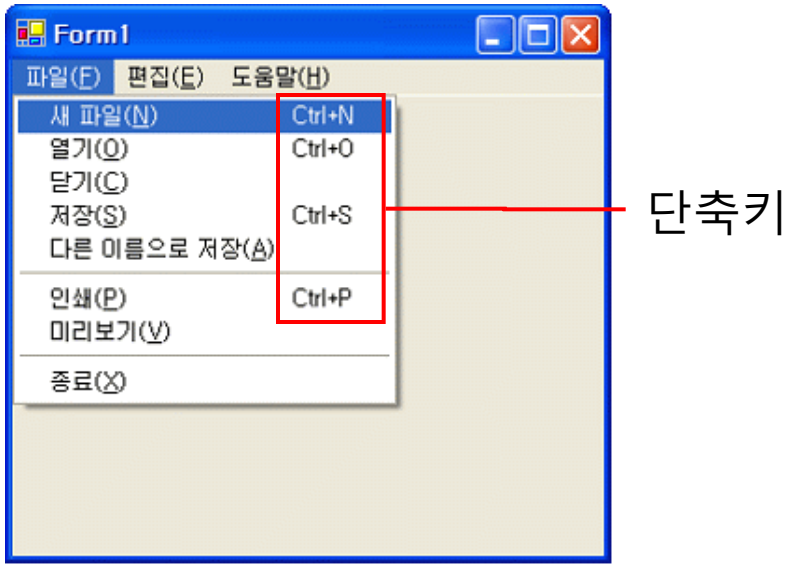
- 메뉴 항목의 추가
  - "여기에 입력하십시오"라는 문구에 "파일(&F)"를 입력
  - 우측의 "여기에 입력하십시오"는 두 번째 메인 메뉴
  - 아래의 "여기에 입력하십시오"는 "파일" 메뉴에 대한 메뉴항목
  - 위의 과정을 반복



파일(&F)의 F 앞에 &를 붙인 이유 !
메뉴에 단축문자를 부여하기 위한 방법
<Alt>키와 단축문자를 눌러서 메뉴의 선택이 가능함

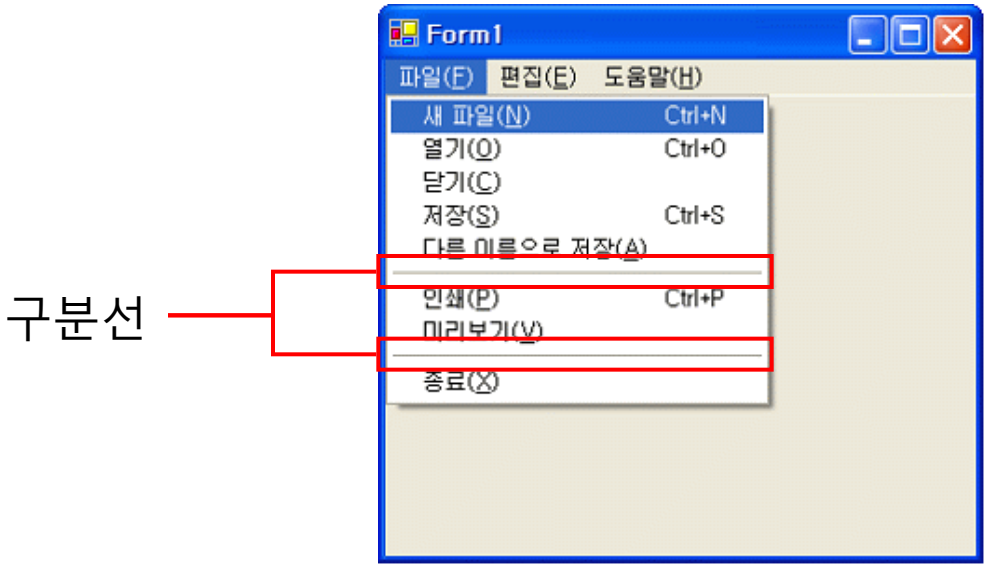
메인 메뉴의 작성 [3/5]

- 메뉴 항목의 단축키 적용
  - 단축키를 적용할 메뉴 항목을 선택
  - 속성 브라우저의 Shortcut 프로퍼티를 통해 설정
    - p.438, 메뉴 항목의 속성 브라우저 참조



메인 메뉴의 작성 [4/5]

- 구분선
  - 메뉴 항목을 그룹화하기 위하여 구분선을 사용
  - "여기에 입력하십시오"라는 문구에 '-'를 입력하거나 마우스 오른쪽 버튼을 선택하여【구분선 삽입】메뉴를 선택



메인 메뉴의 작성 [5/5]

[예제 10.1 - MainMenuApp.cs]

메뉴 항목의 프로퍼티 정보를 참고하여 MainMenu를 작성한다.

Text 프로퍼티		(Name)	Shortcut 프로퍼티
파일(&F)			
	새 파일(&N)	mnuFileNew	CtrlN(Control N)
	열기(&O)...	mnuFileOpen	CtrlO(Control O)
	닫기(&C)	mnuFileClose	
	저장(&S)	mnuFileSave	CtrlS(Control S)
	다른 이름으로 저장(&A)...	mnuFileSaveAs	
	-	mnuSep1	
	인쇄(&P)...	mnuFilePrint	CtrlP(Control P)
	미리 보기(&V)	mnuFileView	
	-	mnuSep2	
	종료(&X)	mnuFileExit	
편집(&E)		mnuEdit	
	잘라내기(&T)	mnuEditCut	CtrlX(Control X)
	복사(&C)	mnuEditCopy	CtrlC(Control C)
	붙여넣기(&P)	mnuEditPaste	CtrlV(Control V)
도움말(&H)		mnuHelp	
	프로그램 정보(&A)...	mnuHelpAbout	

메뉴 항목의 이벤트 [1/4]

- 메뉴 항목을 클릭하면 발생하는 이벤트
  - Popup
    - 메뉴 창이 나타날 때 발생
    - 메뉴 항목의 상태를 동적으로 변화시키는데 사용
  - Select
    - 메뉴항목이 강조되지만 선택된 상태는 아님
    - 메뉴 선택을 돕기 위한 도움말을 표시할 때 유용히 사용
  - Click
    - 메뉴 항목을 클릭했을 때 발생
    - 메뉴와 관련된 이벤트 중에서 가장 많이 사용하는 이벤트

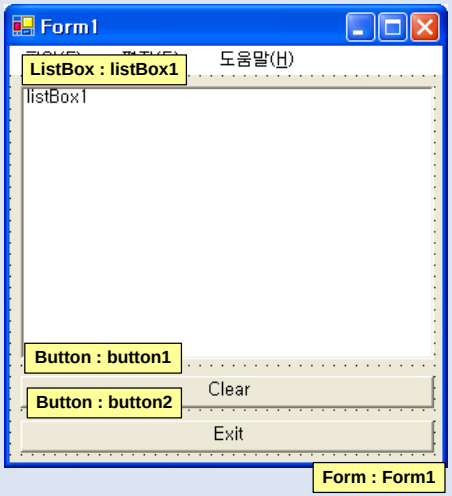
메뉴 항목의 이벤트 [2/4]

- 메뉴 항목과 관련된 Form 객체의 이벤트
  - MenuStart
    - 메뉴가 처음으로 입력 포커스를 얻을 때 발생
    - 폼의 사용자 인터페이스를 관리하기 위해서 사용
  - MenuComplete
    - 메뉴가 입력 포커스를 잃을 때 발생
    - 메뉴가 사라지는 순간을 확인하기 위해서 사용

메뉴 항목의 이벤트 [3/4]

[예제 10.2 – MenuClickApp.cs]

1) 폼 설계



2) 컴포넌트

컴포넌트 : (Name)	프로퍼티	값
MainMenu : mainMenu1		

3) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	MenuClickApp
Button : button1	Text	Clear
Button : button2	Text	Exit
ListBox : listBox1		

4) 이벤트 처리기

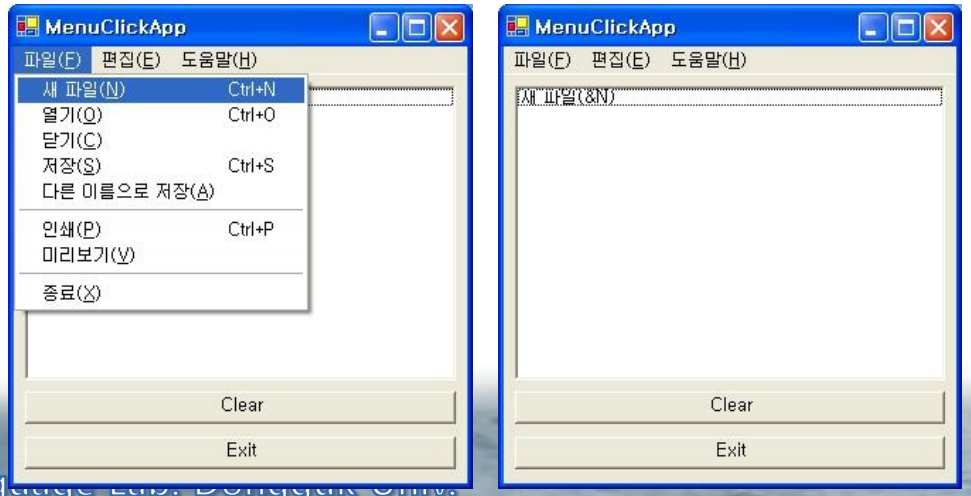
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()
Button : button2	Click	button2_Click()
MenuItem : mnuFileNew	Click	mnuFileNew_Click()
MenuItem : mnuFileOpen	Click	mnuFileOpen_Click()
…		

메뉴 항목의 이벤트 [4/4]

```
private void mnuFileNew_Click(object sender, EventArgs e) {
    listBox1.Items.Add(mnuFileNew.Text);
}
private void mnuFileOpen_Click(object sender, EventArgs e) {
    listBox1.Items.Add(mnuFileOpen.Text);
}
// 나머지 메뉴 항목도 동일하게 작성
private void button1_Click(object sender, EventArgs e) {
    listBox1.Items.Clear();
}
private void button2_Click(object sender, EventArgs e) {
    Application.Exit();
}
```

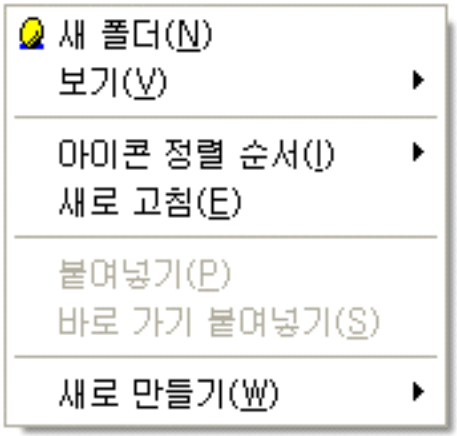
실행 방법 : 각 메뉴 항목을 클릭하여 Click 이벤트를 발생시킨다.

실행 결과 :



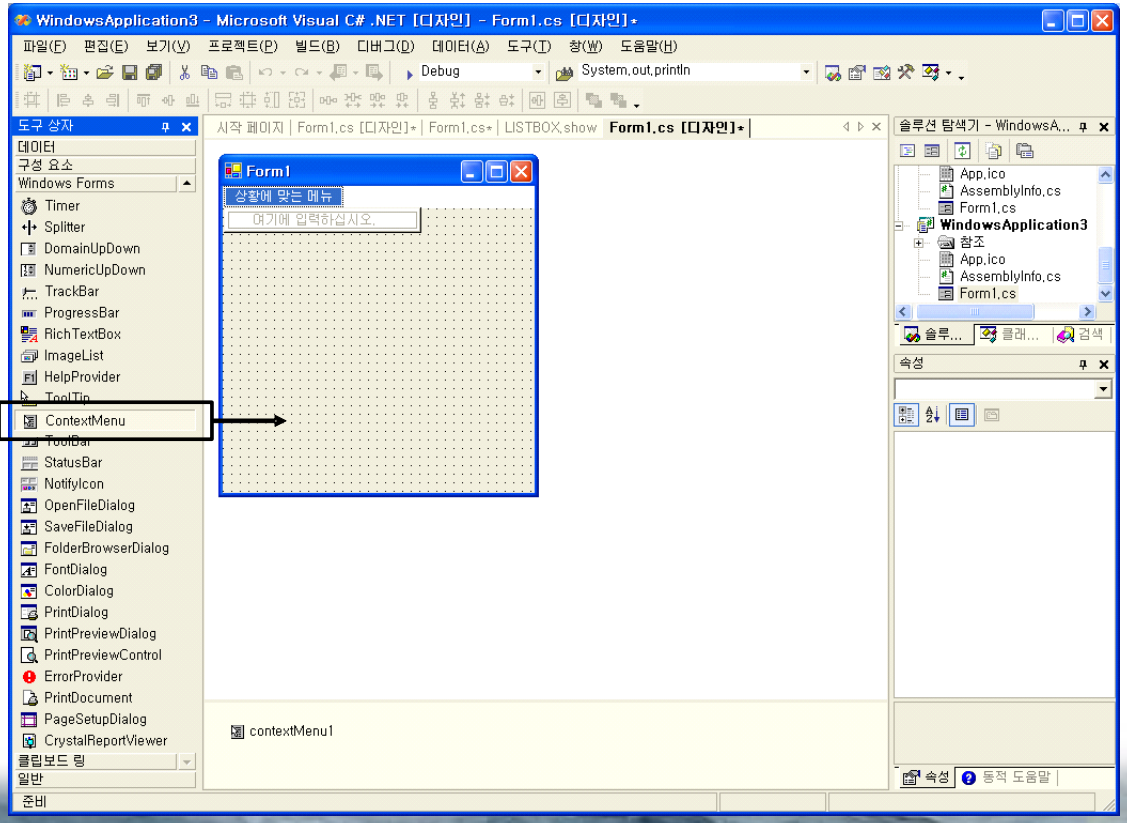
상황 메뉴

- 컨트롤 위에서 마우스의 오른쪽 버튼을 클릭하였을 때 표시되는 팝업 메뉴
  - 현재 애플리케이션의 상태가 반영
  - 상황에 따라 독자적인 메뉴 항목을 가짐



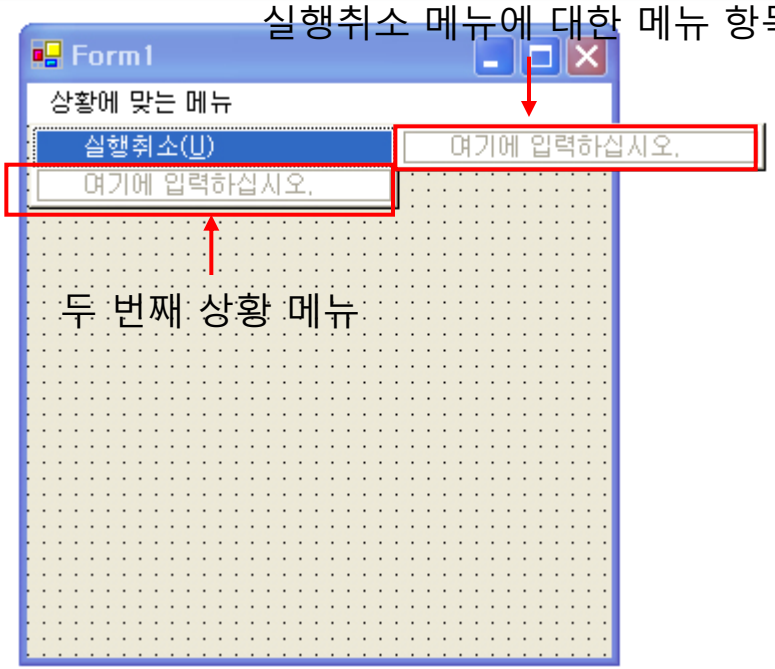
상황 메뉴의 작성 [1/3]

- ContextMenu 컴포넌트의 추가
  - 【도구상자】➡【ContextMenu】를 선택하여 폼 디자이너로 이동



상황 메뉴의 작성 [2/3]

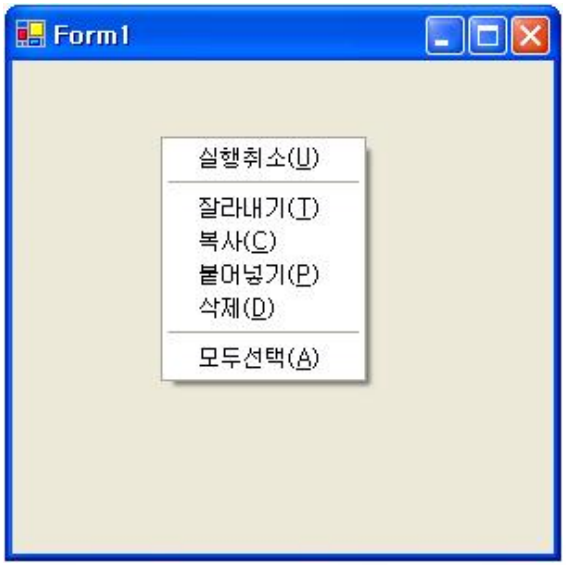
- 메뉴 항목의 추가
  - "여기에 입력하십시오"라는 문구에 "실행취소(&U)"를 입력
  - 우측의 "여기에 입력하십시오"는 "실행취소" 메뉴에 대한 메뉴 항목
  - 아래의 "여기에 입력하십시오"는 두 번째 상황 메뉴
  - 위의 과정을 반복



상황 메뉴는 메인 메뉴와 동일한 프로퍼티와 이벤트를 가진다 !

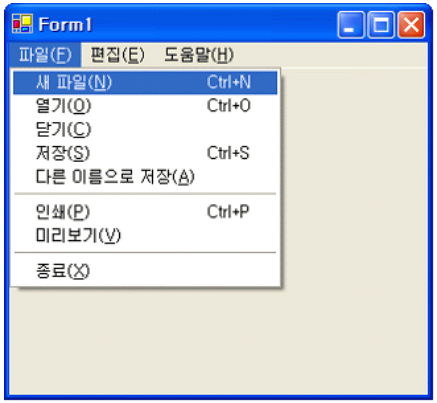
상황 메뉴의 작성 [3/3]

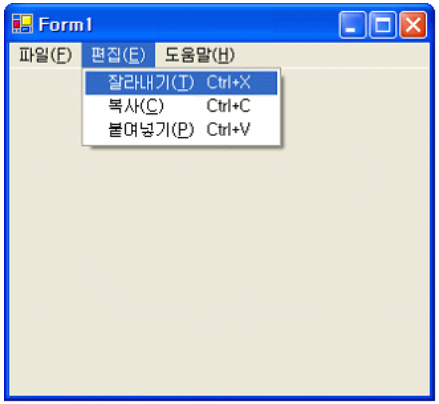
- 완성된 상황 메뉴를 해당 컨트롤의 ContextMenu 프로퍼티에 설정
  - 컨트롤마다 상황 메뉴를 가질 수 있기 때문에 적용하고자 하는 컨트롤의 ContextMenu 프로퍼티에 설정
  - 폼의 ContextMenu 프로퍼티에 작성된 contextMenu1 컴포넌트를 지정한 예



메뉴 – 단축 문자와 단축키 [1/2]

- 단축문자
  - 메뉴항목의 이름에 &를 붙인 형태
    - 파일(&F), 복사(&C)
  - 메뉴 표시줄에 나타나는 메인 메뉴 사이에서는 반드시 유일해야 함
  - 메인 메뉴의 서로 다른 메뉴 항목에 대해서는 중복 사용 가능





메뉴 – 단축 문자와 단축키 [2/2]

- 단축키
  - 메뉴항목의 Shortcut 프로퍼티를 통해 설정
  - 단축키는 하나의 애플리케이션에 포함된 모든 메뉴 항목에 대하여 유일하도록 설정
  - 중복하여 설정할 경우, 두 번째 이후로 설정된 메뉴 항목의 단축키는 반영되지 않음

마우스 다루기

- 마우스
  - 윈도우 사용자에게 가장 편리하고 친숙한 입력장치
  - 윈폼 애플리케이션의 사용자 상호작용은 대부분 마우스를 통해 이루어짐
  - 사용자가 마우스를 이동하거나 클릭하면 이벤트가 발생

- 마우스 이벤트
  - 이동 이벤트
    - 사용자가 마우스의 위치를 이동시킬 경우 발생
  - 선택 이벤트
    - 사용자가 마우스의 버튼을 클릭할 경우 발생

마우스 이동 이벤트

- MouseEnter
  - 마우스 포인터가 컨트롤이나 폼 영역에 들어올 때 발생
- MouseHover
  - 마우스 포인터가 컨트롤이나 폼에서 이동하는 것을 멈출 때 발생
  - 매번 발생하지 않으며 처음 멈출 때만 발생
- MouseLeave
  - 마우스 포인터가 컨트롤이나 폼 영역을 벗어날 때 발생
- MouseMove
  - 마우스 포인터가 새로운 영역으로 이동할 때 발생
- MouseWheel
  - 입력포커스를 가지고 있는 컨트롤이나 폼 위에서 마우스 휠 버튼을 회전시킬 때 발생

마우스 이동 이벤트 처리기

- EventHandler 델리게이트형의 처리기를 사용하는 이벤트
  - MouseEnter, MouseHover, MouseLeave
- MouseEventHandler 델리케이트형의 처리기를 사용하는 이벤트
  - MouseMove, MouseWheel
  - MouseEventArgs 클래스가 제공하는 프로퍼티를 이용하여 마우스의 위치와 상태에 대한 추가적인 정보 사용 가능

```
public delegate void EventHander(object sender, EventArgs e);
public delegate void MouseEventHander(object sender, MouseEventArgs e);
```

MouseEventArgs 클래스의 프로퍼티 [1/2]

- Button
  - 마우스의 상태를 나타내는 MouseButtons 열거형 값
  - MouseButtons 열거형
    - Left : 마우스 왼쪽 버튼을 클릭한 상태
    - Middle : 마우스 중앙 버튼을 클릭한 상태
    - None : 마우스를 누르지 않은 상태
    - Right : 마우스 오른쪽 버튼을 클릭한 상태
    - XButton1 : 첫 번째 X버튼을 클릭한 상태
    - XButton2 : 두 번째 X버튼을 클릭한 상태

IntelliMouse에서 제공하는 버튼

- Clicks
  - 마우스 버튼을 클릭한 횟수

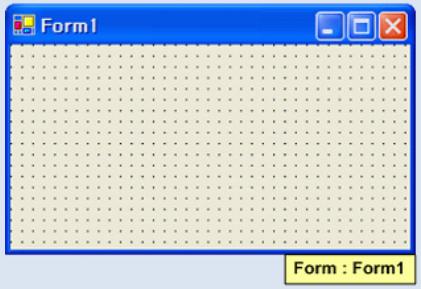
MouseEventArgs 클래스의 프로퍼티 [2/2]

- Delta
  - 마우스 휠의 회전수(휠을 1회 돌리는 것)를 나타내는 값
- X
  - 클라이언트 좌표 내에서, 마우스 위치의 X좌표
- Y
  - 클라이언트 좌표 내에서, 마우스 위치의 Y좌표

마우스 이동 예제 [1/2]

[예제 10.3 – MousePositionApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	MousePositionApp

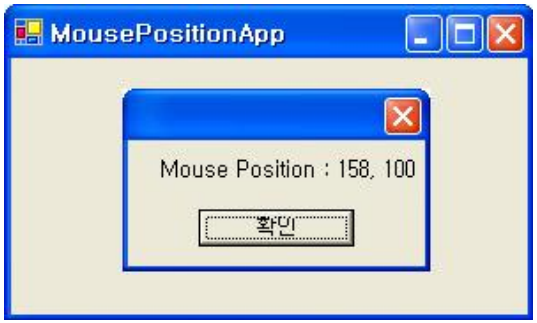
3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
Form : Form1	MouseEnter	Form1_MouseEnter()

마우스 이동 예제 [2/2]

```
private void Form1_MouseEnter(object sender, EventArgs e) {
    Point mousePoint = PointToClient(MousePosition);
    string msg = "Mouse Position : " + mousePoint.X + ", "
                                     + mousePoint.Y;
    MessageBox.Show(msg);
}
```

실행 방법 : 폼 위로 마우스를 이동하여 MouseEnter 이벤트를 발생시킨다.
실행 결과 :



- MousePosition 프로퍼티
  - 마우스의 좌표를 전체화면에 대한 상대좌표로 Point 구조체 형으로 반환
- PointToClient 메소드
  - 전체화면에 대한 상대좌표를 클라이언트 좌표로 변환

마우스 선택 이벤트

- MouseDown
  - 폼이나 컨트롤에서 마우스 버튼을 누를 때 발생
- MouseUp
  - 폼이나 컨트롤에서 마우스 버튼을 누른 후 해제할 때 발생
- Click
  - 폼이나 컨트롤을 클릭할 때 발생
- DoubleClick
  - 폼이나 컨트롤을 더블 클릭할 때 발생

마우스 선택 이벤트 처리기

- EventHandler 델리게이트형의 처리기를 사용하는 이벤트
  - Click, DoubleClick
- MouseEventHandler 델리케이트형의 처리기를 사용하는 이벤트
  - MouseDown, MouseUp

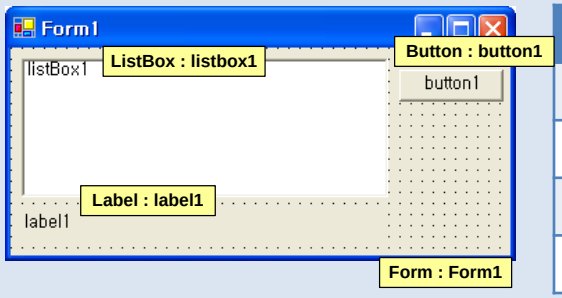
마우스 이벤트의 발생 순서 [1/3]

- 마우스의 이벤트가 비동기적으로 불특정 시간에 발생하더라도 상대적인 순서는 보장됨
  - MouseEnter와 MouseLeave사이에 발생하는 이벤트
    - MouseHover
    - MouseMove
  - Click 이벤트
    - MouseDown과 MouseUp 이벤트 다음에 발생
  - DoubleClick 이벤트
    - Click 이벤트 다음에 발생

마우스 이벤트의 발생 순서 [2/3]

[예제 10.5 – MouseEventApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	MouseEventApp
ListBox : listBox1	Items	
Button : button1	Text	Close
Label : label1	Text	MouseEventArgsLabel

3) 멤버

```
private void UpdateEventLabels(string msg, int x, int y, MouseEventArgs e) {
    string message = string.Format("{0} X:{1}, Y:{2}", msg, x, y);
    string eventMsg = DateTime.Now.ToShortTimeString();
    eventMsg += " " + message;
    listBox1.Items.Insert(0, eventMsg);
    listBox1.TopIndex = 0;
    string mouseInfo;
    if (e != null) {
        mouseInfo = string.Format("Clicks: {0}, Delta: {1}, " +  "Buttons: {2}",
                                  e.Clicks, e.Delta, e.Button.ToString());
    } else { mouseInfo = string.Format("Clicks: {0}", msg); }
    label1.Text = mouseInfo;
}
```

마우스 이벤트의 발생 순서 [3/3]

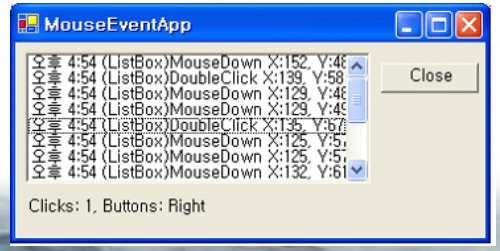
4) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
ListBox : listBox1	MouseDown	listBox1_MouseDown()
	DoubleClick	listBox1_DoubleClick()
Button : button1	Click	Button1_Click()

```
private void listBox1_MouseDown(object sender, MouseEventArgs e) {
    UpdateEventLabels("(ListBox)MouseDown", e.X, e.Y, e);
}
private void listBox1_DoubleClick(object sender, EventArgs e) {
    Point mousePoint = PointToClient(MousePosition);
    UpdateEventLabels("(ListBox)DoubleClick", mousePoint.X, mousePoint.Y, null);
}
private void button1_Click(object sender, EventArgs e) {
    Application.Exit();
}
```

실행 방법 : ListBox 상에서 MouseDown, DoubleClick 이벤트를 발생시킨다.

실행 결과 :

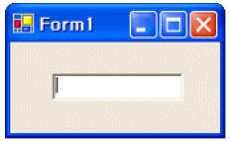


키보드 다루기

- 윈폼 애플리케이션은 사용자로부터 직접 키보드 입력을 받지 않음
  - 텍스트 박스와 같은 컨트롤을 이용하여 키보드 입력이 이루어짐
- C#은 컨트롤을 이용한 키보드 입력 이외에도 사용자 입력을 직접 처리할 수 있는 방법을 제공함

입력 포커스 [1/7]

- 키보드를 통해 입력이 가능한 컨트롤을 표시
  - 키보드를 이용한 사용자의 입력은 여러 개의 컨트롤에서 동시에 사용할 수 없음
  - 입력 포커스를 가지는 컨트롤만이 키보드를 통해 사용자의 입력을 받을 수 있음
  - 입력 포커스를 가지는 컨트롤은 자신의 형태를 변경함
    - 텍스트 상자가 입력 포커스를 가지는 경우



    - 버튼 컨트롤이 입력 포커스를 가지는 경우

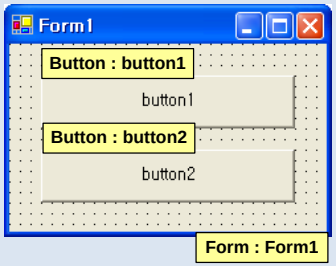
입력 포커스 [2/7]

- Focus() 메소드
  - 특정 컨트롤로 입력 포커스를 이동시키기 위한 메소드
  - Control 클래스로부터 파생된 대부분의 컨트롤들이 가지는 메소드
  - 특정 컨트롤에 대한 포커스가 변경될 경우, 참을 반환
  - 특정 컨트롤에 대한 포커스가 변경되지 못할 경우, 거짓을 반환

입력 포커스 [3/7]

[예제 10.7 – FocusApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	FocusApp
Button : button1	Text	버튼1
	BackColor	ControlDark
Button : button2	Text	버튼2

3) 이벤트 처리기

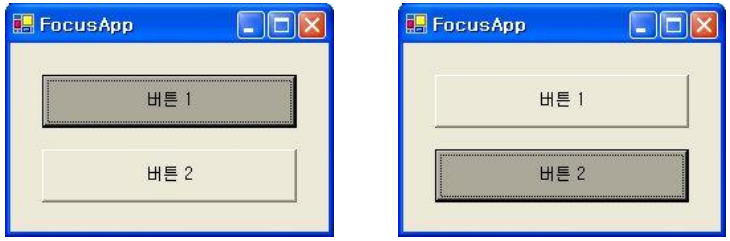
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()
Button : button2	Click	button2_Click()

입력 포커스 [4/7]

```
private void button1_Click(object sender, EventArgs e) {
    this.button1.BackColor = SystemColors.Control;
    this.button2.Focus();
    if (this.button2.Focused)
        this.button2.BackColor = SystemColors.ControlDark;
}
private void button2_Click(object sender, EventArgs e) {
    this.button2.BackColor = SystemColors.Control;
    this.button1.Focus();
    if (this.button1.Focused)
        this.button1.BackColor = SystemColors.ControlDark;
}
```

실행 방법 : 버튼1과 버튼2를 차례대로 선택하여 Click 이벤트를 발생시킨다.

실행 결과 :



- Focused 프로퍼티
  - 해당 컨트롤이 입력 포커스를 가졌는지 확인

입력 포커스 [5/7]

- 입력 포커스와 관련된 프로퍼티
  - 대부분의 컨트롤에서 공통적으로 제공
  - CanFocus
    - 컨트롤이 포커스를 받을 수 있는지 여부를 나타내는 값을 가져옴
  - ContainsFocus
    - 컨트롤이나 해당 컨트롤의 자식 컨트롤이 현재 입력 포커스를 가지고 있는지 여부를 나타내는 값을 가져옴
  - Focused
    - 컨트롤에 입력 포커스가 있는지 여부를 나타내는 값을 가져옴

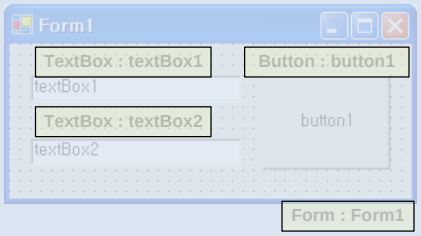
입력 포커스 [6/7]

- 입력 포커스와 관련된 이벤트
  - Enter
    - 자신 또는 자식 컨트롤이 입력 포커스를 가질 때 발생
  - Leave
    - 자신 또는 자식 컨트롤이 입력 포커스를 잃을 때 발생
  - Enter, Leave 이벤트를 이용하면 특정 컨트롤이 입력 포커스를 받았음을 알릴 수 있도록 사용자 인터페이스를 변경할 수 있음

입력 포커스 [7/7]

[예제 10.8 – FocusEventApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	FocusEventApp
Label : label1	Text	비밀번호
TextBox : textBox1	Text	
	PasswordChar	*

3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
TextBox : textBox1	Enter	textBox1_Enter()

실행 방법 : 키보드나 마우스를 이용하여 textBox1에 Enter 이벤트 발생

실행 결과 :

Keys 열거형

- 키보드로 입력된 모든 값이 정의된 열거형
  - System.Windows.Forms 네임스페이스에 포함
  - 183개의 값으로 구성
  - 키보드에 대한 열거형 뿐만 아니라 마우스에 대해서도 정의

기호상수	설 명	기호상수	설 명
A	문자 A	D3	숫자 3
F5	기능키 F5	NumPad3	숫자 패드 3
LShiftKey	왼쪽 쉬프트 키	PageUp	페이지업 키
RControlKey	오른쪽 컨트롤 키	Delete	델 키
Left	왼쪽 화살표 키	Up	위쪽 화살표 키
Divide	나누기 키(/)	Lbutton	마우스 왼쪽 버튼

키보드 이벤트

- KeyDown
  - 사용자가 키를 누를 때 발생
  - 키 상태와 보조키를 위한 Keys 열거형 정보를 사용할 수 있음
- KeyPress
  - 키가 완전히 눌러진 상태에서 발생
  - 키 문자에 대한 정보를 사용할 수 있음
- KeyUp
  - 키를 떼었을 때 발생
  - 키 상태와 보조키를 위한 Keys 열거형 정보를 사용할 수 있음
- 이벤트 발생순서
  - KeyDown ➡ KeyPress ➡ KeyUp

키보드 이벤트 처리하기 [1/4]

- KeyDown, KeyUP 이벤트 처리기
  - KeyEventArgs 클래스의 객체를 매개 변수로 가짐
    - KeyEventArgs 클래스는 키보드 입력을 직접 처리할 수 있는 프로퍼티를 제공

프로퍼티	설 명
Alt	<Alt>키를 눌렀는지 여부를 나타내는 값을 가져옴
Control	<Ctrl>키를 눌렀는지 여부를 나타내는 값을 가져옴
Handled	이벤트가 처리되었는지 여부를 나타내는 값을 가져오거나 설정
KeyCode	KeyDown 또는 KeyUP 이벤트에 대한 키보드 코드를 가져옴
KeyData	Keydown 또는 KeyUp 이벤트에 대한 키 데이터를 가져옴
KeyValue	Keydown 또는 KeyUp 이벤트에 대한 키보드 값을 가져옴
Modifiers	KeyDown 또는 KeyUp 이벤트에 대한 보조 플래그를 가져옴 이는 누른 보조키(<Ctrl>, <Shift> 및 <Alt>)의 조합을 나타냄
Shift	<Shift> 키가 눌렸는지 여부를 나타내는 값을 가져옴

    - 키의 상태와 보조키에 대한 정보를 쉽게 얻을 수 있음

키보드 이벤트 처리하기 [2/4]

- KeyPress 이벤트 처리기
  - KeyPressEventArgs 클래스의 객체를 매개변수로 가짐
    - KeyPressEventArgs 클래스는 키 코드와 보조키에 대한 정보 대신에 눌려진 문자 값을 처리할 수 있는 프로퍼티를 제공

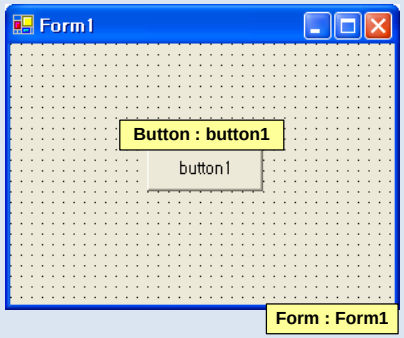
프로퍼티	설 명
Handled	이벤트가 처리되었는지 여부를 나타내는 값을 가져오거나 설정
KeyChar	눌려진 문자값

    - KeyChar 프로퍼티는 사용자가 누른 키의 실제 문자 값을 반환
      - a키가 눌릴 경우 : 'a'를 반환
      - <Shift>+a가 눌릴 경우 : 'A'를 반환

키보드 이벤트 처리하기 [3/4]

[예제 10.9 – KeyEventApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	KeyEventApp
Button : button1	Text	Button

3) 멤버

```
public int xPt, yPt;
public static readonly int MOVE = 10;
```

3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
Button : button1	KeyUp	button1_KeyUp()

키보드 이벤트 처리하기 [4/4]

```
void button1_KeyUp(object sender, KeyEventArgs e) {
    this.xPt = this.button1.Location.X;
    this.yPt = this.button1.Location.Y;
    switch (e.KeyCode) {
        case Keys.Left :
            xPt -= MOVE;        break;
        case Keys.Right :
            xPt += MOVE;        break;
        case Keys.Up :
            yPt -= MOVE;        break;
        case Keys.Down :
            yPt += MOVE;        break;
    }
    this.button1.Text = e.KeyCode.ToString();
    this.button1.Location = new Point(xPt, yPt);
}
```

실행 방법 : 키보드를 눌러서 KeyUp 이벤트를 발생시킨다.
실행 결과 :