객체지향 프로그래밍

고급폼

KYUNGSUNG UNIVERSITY SINCE 1955

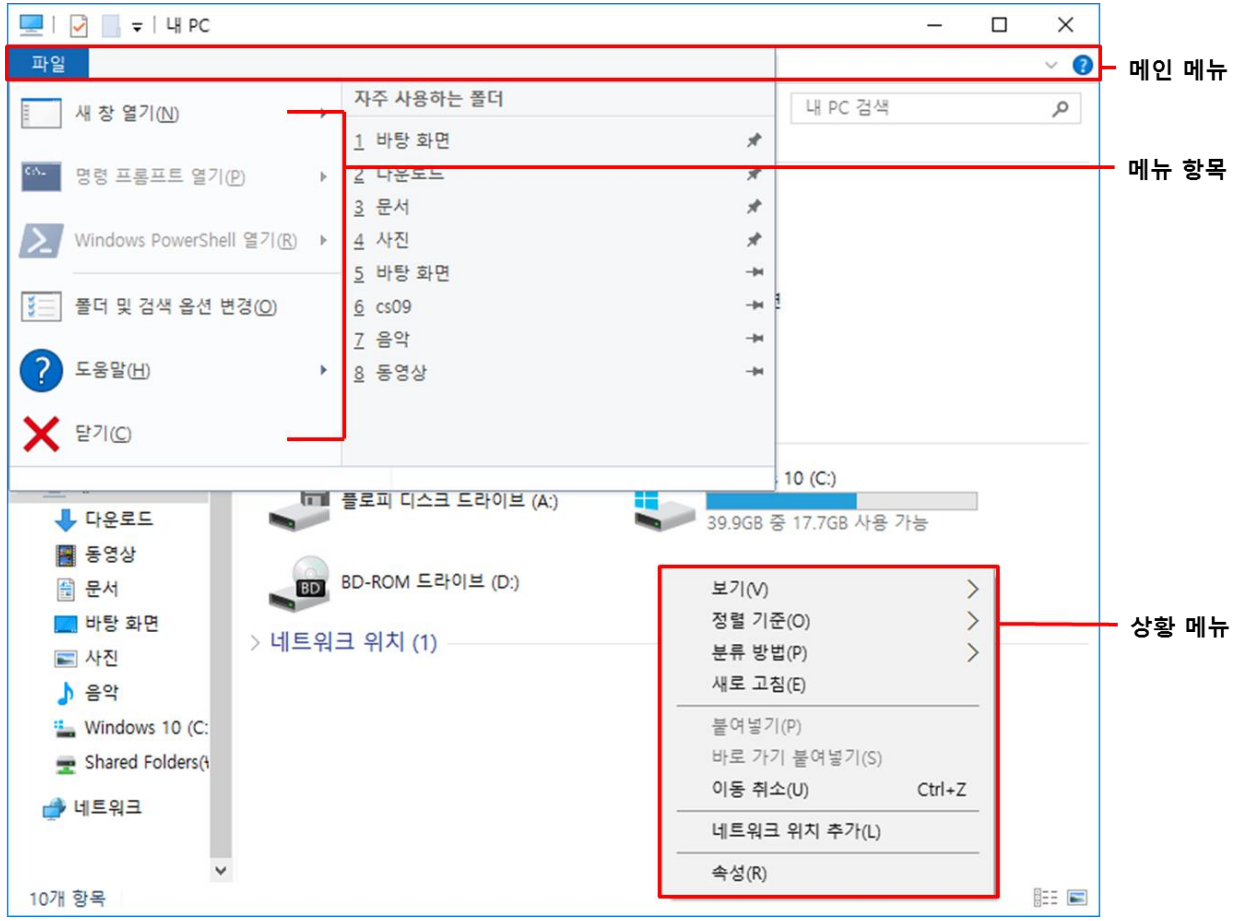
메뉴 다루기

- 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

메뉴의 구성


내 PC
파일
메인 메뉴
메뉴 항목
상황 메뉴
자주 사용하는 폴더
새 창 열기(N)
명령 프롬프트 열기(P)
Windows PowerShell 열기(R)
폴더 및 검색 옵션 변경(O)
도움말(H)
닫기(C)
1 바탕 화면
3 문서
4 사진
5 바탕 화면
6 cs09
7 음악
8 동영상
내 PC 검색
다운로드
동영상
문서
바탕 화면
사진
음악
Windows 10 (C:
Shared Folders(
네트워크
플로피 디스크 드라이브 (A:)
BD-ROM 드라이브 (D:)
네트워크 위치 (1)
39.9GB 중 17.7GB 사용 가능
보기(V)
정렬 기준(O)
분류 방법(P)
새로 고침(E)
붙여넣기(P)
바로 가기 붙여넣기(S)
이동 취소(U) Ctrl+Z
네트워크 위치 추가(L)
속성(R)
10개 항목

메인 메뉴

- 폼의 상단에 배치되는 메뉴
- 마우스 클릭뿐만 아니라 단축키를 통해서는 접근할 수 있는 가장 기본적인 사용자 인터페이스
- 통합 개발 환경의 MenuStrip 컴포넌트를 통하여 작성

메인 메뉴의 작성

- 메뉴 항목의 추가
  - 메뉴에 단축문자를 부여하기 위한 방법
    - 사용할 단축문자 앞에 &를 붙임
    - <Alt>키와 단축문자를 눌러서 메뉴의 선택이 가능
- 메뉴 항목의 단축키 적용
  - 단축키를 적용할 메뉴 항목을 선택
- 구분선
  - 메뉴 항목을 그룹화하기 위하여 구분선을 사용
  - 메뉴 항목에 '-'를 입력

메인 메뉴의 작성

Text 프로퍼티		Shortcut 프로퍼티
파일(&F)		
	새 파일(&N)	CtrlN(Control N)
	열기(&O)...	CtrlO(Control O)
	닫기(&C)	
	저장(&S)	CtrlS(Control S)
	다른 이름으로 저장(&A)...	
	-	
	인쇄(&P)...	CtrlP(Control P)
	미리 보기(&V)	
	-	
	종료(&X)	
편집(&E)		
	잘라내기(&T)	CtrlX(Control X)
	복사(&C)	CtrlC(Control C)
	붙여넣기(&P)	CtrlV(Control V)
도움말(&H)		
	프로그램 정보(&A)...	

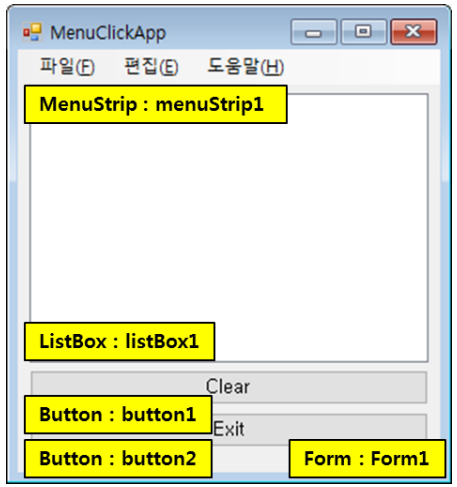
메뉴 항목의 이벤트

- 메뉴 항목을 클릭하면 발생하는 이벤트
  - Click
    - 메뉴 항목을 클릭했을 때 발생
    - 메뉴와 관련된 이벤트 중에서 가장 많이 사용하는 이벤트

메뉴 항목의 이벤트

- 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



컴포넌트 : (Name)	프로퍼티	값
MenuStrip : menuStrip1		

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

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

```
using System;
using System.Windows.Forms;

namespace Ex10_02_MenuClickApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
```

메뉴 항목의 이벤트

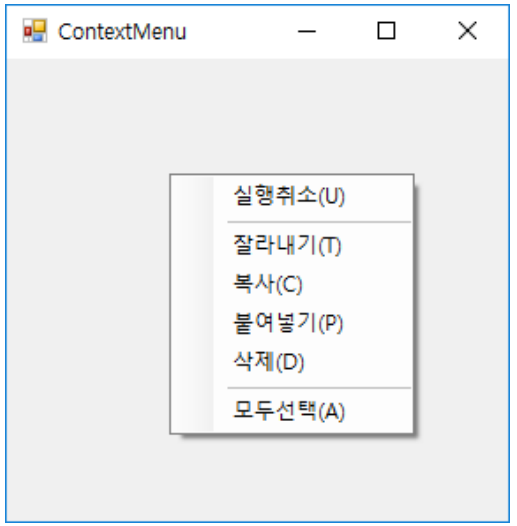
```
        private void 새파일ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            listBox1.Items.Add(((ToolStripMenuItem)sender).Text);
        }

        private void 열기ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            listBox1.Items.Add(((ToolStripMenuItem)sender).Text);
        }

         . . .

        private void 붙여넣기ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            listBox1.Items.Add(((ToolStripMenuItem)sender).Text);
        }

        private void 정보ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            listBox1.Items.Add(((ToolStripMenuItem)sender).Text);
        }
    }
}
```

상황 메뉴

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

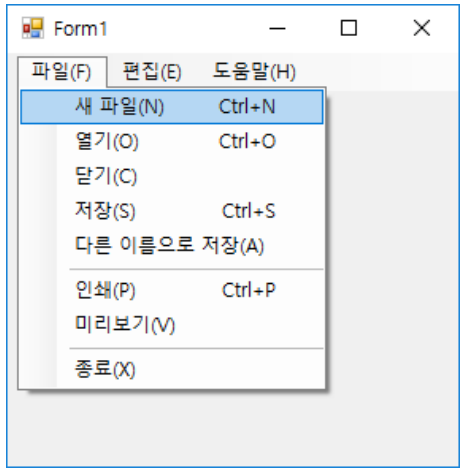
상황 메뉴의 작성

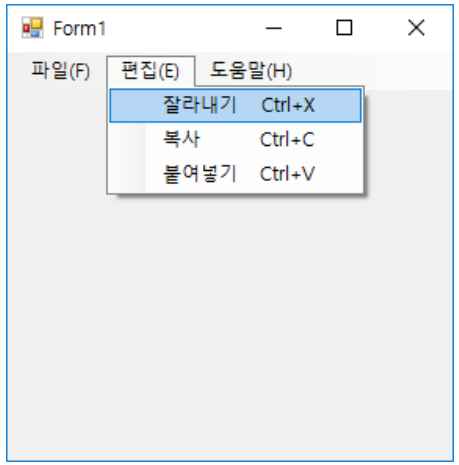
- ContextMenuStrip 컴포넌트의 추가
- 상황 메뉴는 메인 메뉴와 동일한 프로퍼티와 이벤트를 가짐
- 완성된 상황 메뉴를 해당 폼 또는 컨트롤의 ContextMenu 프로퍼티에 설정
  - 컨트롤마다 상황 메뉴를 가질 수 있기 때문에 적용하고자 하는 컨트롤의 ContextMenu 프로퍼티에 설정
  - 폼의 ContextMenu 프로퍼티에 작성된 contextMenu1 컴포넌트를 지정한 예



메뉴 – 단축 문자와 단축키

- 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

- 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 이루어 짐
  - 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

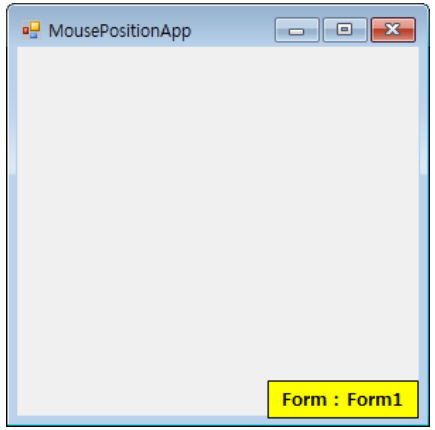
- EventHandler 델리게이트형의 처리기를 사용하는 이벤트
  - MouseEnter, MouseHover, MouseLeave
- MouseEventHandler 델리케이트형의 처리기를 사용하는 이벤트
  - MouseMove, MouseWheel
  - MouseEventArgs 클래스가 제공하는 프로퍼티를 이용하여 마우스의 위치와 상태에 대한 추가적인 정보 사용 가능

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

MouseEventArgs 클래스의 프로퍼티

- 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
- Clicks
  - 마우스 버튼을 클릭한 횟수
- 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

IntelliMouse에서 제공하는 버튼



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

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

```
using System;
using System.Windows.Forms;

namespace Ex10_04_MousePositionApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

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

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

마우스 선택 이벤트

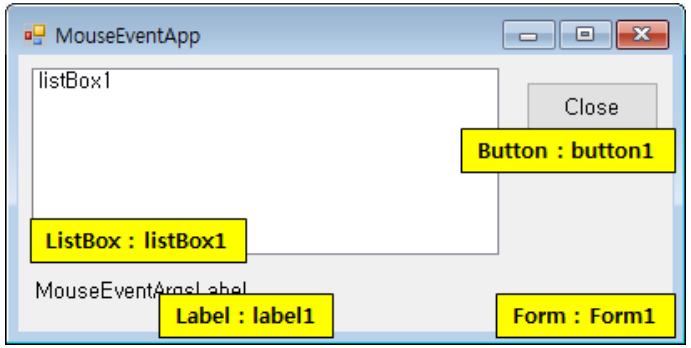
- MouseDown
  - 폼이나 컨트롤에서 마우스 버튼을 누를 때 발생
- MouseUp
  - 폼이나 컨트롤에서 마우스 버튼을 누른 후 해제할 때 발생
- Click
  - 폼이나 컨트롤을 클릭할 때 발생
- DoubleClick
  - 폼이나 컨트롤을 더블 클릭할 때 발생

마우스 선택 이벤트 처리기

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

마우스 이벤트의 발생 순서

- 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



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

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

```
using System.Data;
using System.Windows.Forms;

namespace Ex10_05_MouseEventApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
    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;
    }
```

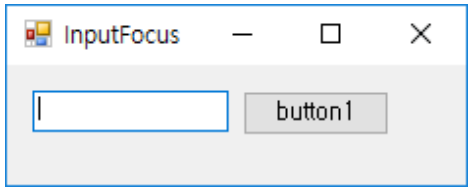
```
        private void listBox1_MouseDown(object sender, MouseEventArgs e)
        {
            UpdateEventLabels("(ListBox)MouseDown", e.X, e.Y, e);
        }

        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();
        }
    }
}
```

키보드 다루기

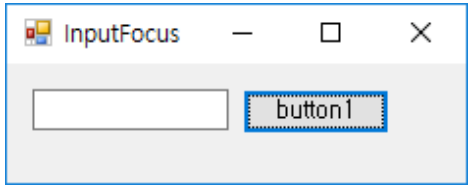
- 윈폼 애플리케이션은 사용자로부터 직접 키보드 입력을 받지 않음
  - 텍스트 박스와 같은 컨트롤을 이용하여 키보드 입력이 이루어짐
- C#은 컨트롤을 이용한 키보드 입력 이외에도 사용자 입력을 직접 처리할 수 있는 방법을 제공함

입력 포커스

- 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

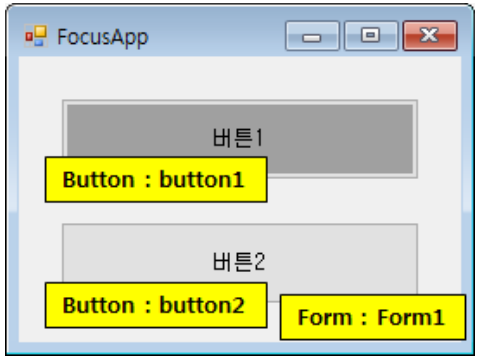


    - 버튼 컨트롤이 입력 포커스를 가지는 경우



입력 포커스

- 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



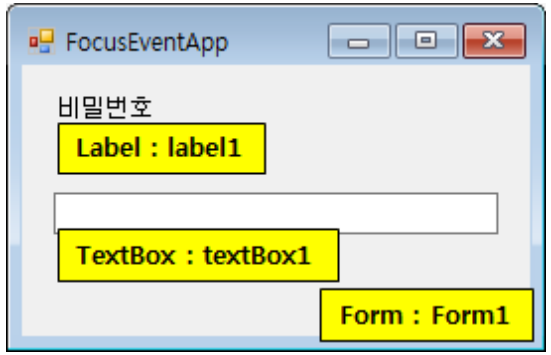
컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	FocusApp
Button : button1	Text	버튼1
	BackColor	ControlDark
Button : button2	Text	버튼2

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

```
using System;
using System.Windows.Forms;

namespace Ex10_07_FocusApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        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;
        }
    }
}
```

- 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 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



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

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

Keys 열거형

- 키보드로 입력된 모든 값이 정의된 열거형
  - System.Windows.Forms 네임스페이스에 포함
  - 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

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

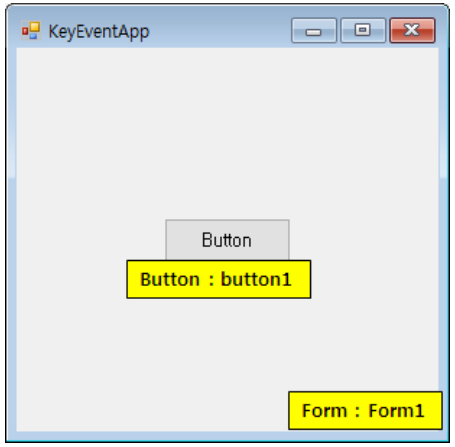
프로퍼티	설 명
Alt	<Alt>키를 눌렀는지 여부를 나타내는 값을 가져옴
Control	<Ctrl>키를 눌렀는지 여부를 나타내는 값을 가져옴
Handled	이벤트가 처리되었는지 여부를 나타내는 값을 가져오거나 설정
KeyCode	KeyDown 또는 KeyUP 이벤트에 대한 키보드 코드를 가져옴
KeyData	Keydown 또는 KeyUp 이벤트에 대한 키 데이터를 가져옴
KeyValue	Keydown 또는 KeyUp 이벤트에 대한 키보드 값을 가져옴
Modifiers	KeyDown 또는 KeyUp 이벤트에 대한 보조 플래그를 가져옴 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

- 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



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

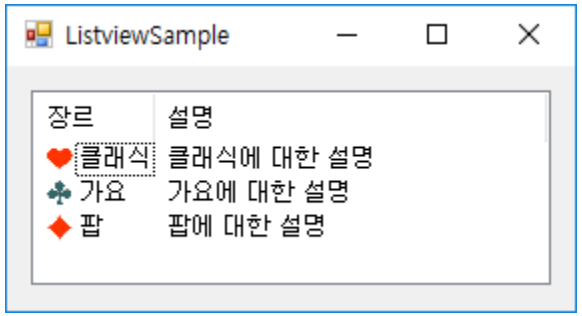
컨트롤 : (Name)	이벤트	메소드명
Button : button1	KeyUp	button1_KeyUp()

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Ex10_09_KeyEventApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

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

리스트뷰

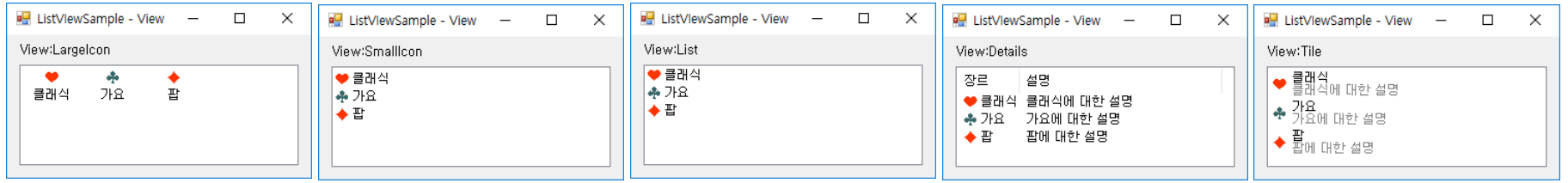
- 리스트 상자와 유사한 형태를 지니며 목록을 구조적으로 장식할 수 있는 컨트롤
  - 리스트 상자 + 추가적인 정보 (아이콘, 설명)



리스트뷰의 형태

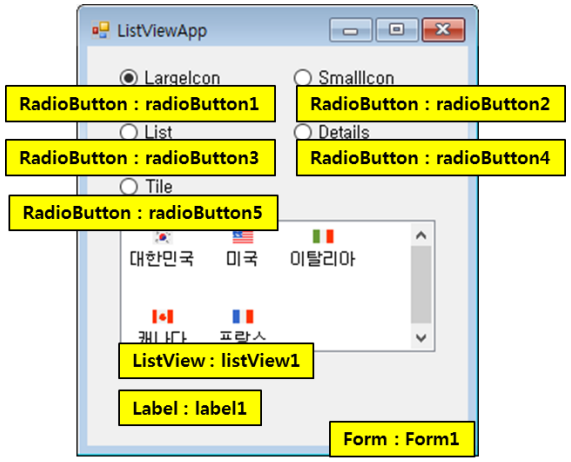
- View 프로퍼티의 값에 따라 다양한 형태를 가짐
  - System.Windows.Forms 네임스페이스에 포함된 View열거형을 값으로 가짐
  - View 열거형

기호상수	설명
LargeIcon	큰 아이콘의 형태 (1)
SmallIcon	작은 아이콘의 형태 (2)
List	간단한 리스트 형태 (3)
Detail	자세한 리스트 형태 (4)
Tile	큰 아이콘이 표시되는 자세한 리스트 형태 (5)



리스트뷰 항목의 선택

- SelectedItems 프로퍼티
  - 리스트 뷰에서 선택된 항목을 저장하는 프로퍼티
  - 반환형
    - ListViewItem 클래스형
      - 리스트 뷰의 MultiSelect 프로퍼티가 거짓일 경우
    - ListViewItem 클래스의 배열형
      - 리스트 뷰의 MultiSelect 프로퍼티가 참일 경우



컴포넌트 : (Name)	프로퍼티	인덱스	값
ImageList : imageList1	Images	0	South Korea.png
		1	USA.png
		2	Italy.png
		3	Canada.png
		4	France.png

https://icons8.com/icons/pack/flags

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ListViewApp
RadioButton : radiobutton1	Text	LargeIcon
	Checked	True
RadioButton : radioButton2	Text	SmallIcon
RadioButton : radioButton3	Text	List
RadioButton : radioButton4	Text	Details
RadioButton : radioButton5	Text	Details
Label : label1	Text	
ListView : listView1	Columns	columnHeader1
		columnHeader2
	LargeImageList	imageList1
	SmallImageList	imageList1
ColumnHeader : columnHeader1	Text	국가
ColumnHeader : columnHeader2	Text	국가번호

Items	프로퍼티	값	프로퍼티	값
ListViewItem0	ImageIndex	0		
	SubItems	ListViewSubItem0	Text	대한민국
		ListViewSubItem1	Text	82
ListViewItem1	ImageIndex	1		
	SubItems	ListViewSubItem0	Text	미국
		ListViewSubItem1	Text	1
ListViewItem2	ImageIndex	2		
	SubItems	ListViewSubItem0	Text	이탈리아
		ListViewSubItem1	Text	39
ListViewItem3	ImageIndex	3		
	SubItems	ListViewSubItem0	Text	캐나다
		ListViewSubItem1	Text	1
ListViewItem4	ImageIndex	4		
	SubItems	ListViewSubItem0	Text	프랑스
		ListViewSubItem1	Text	33

컨트롤 : (Name)	이벤트	메소드명
RadioButton : radioButton1	CheckedChanged	radioButton1_CheckedChanged
RadioButton : radioButton2	CheckedChanged	radioButton2_CheckedChanged
RadioButton : radioButton3	CheckedChanged	radioButton3_CheckedChanged
RadioButton : radioButton4	CheckedChanged	radioButton4_CheckedChanged
RadioButton : radioButton5	CheckedChanged	radioButton5_CheckedChanged
ListView : listView1	Click	listView1_Click()

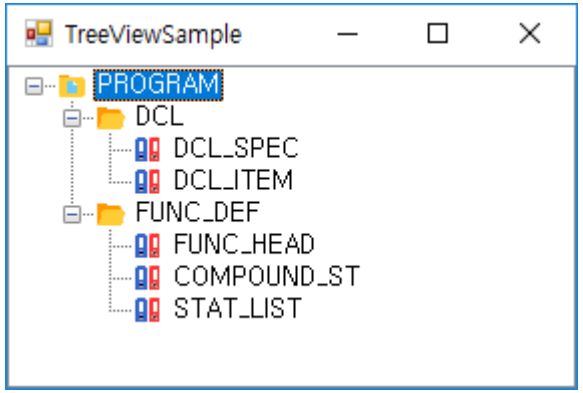
```
using System;
using System.Windows.Forms;

namespace Ex11_01_ListViewApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void listView1_Click(object sender, EventArgs e)
        {
            foreach (ListViewItem item in listView1.SelectedItems)
            {
                ListViewItem.ListViewSubItemCollection subItem = item.SubItems;
                // 각 항목에 대한 부항목을 얻기 위해 SubItems 프로퍼티를 사용
                label1.Text = subItem[0].Text + "의 국가번호는 " +  subItem[1].Text + "입니다.";
            }
        }
```

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton1.Checked)
        // 리스트 뷰의 항목을 큰 아이콘 형태로 보여준다.
        listView1.View = View.LargeIcon;
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton2.Checked)
        // 리스트 뷰의 항목을 작은 아이콘 형태로 보여준다.
        listView1.View = View.SmallIcon;
}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton3.Checked)
        // 리스트 뷰의 항목을 간단한 리스트 형태로 보여준다.
        listView1.View = View.List;
}
```

```
        private void radioButton4_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton4.Checked)
                // 리스트 뷰의 항목을 자세한 리스트 형태로 보여준다.
                listView1.View = View.Details;
        }

        private void radioButton5_CheckedChanged(object sender, EventArgs e)
        {
            if (radioButton5.Checked)
                // 리스트 뷰의 항목을 타일 형태로 보여준다.
                listView1.View = View.Tile;
        }
    }
}
```

트리뷰

- 목록을 계층적으로 보여주기 위한 컨트롤
  - 노드를 계층적으로 표시
  - 노드에 이미지 아이콘을 추가할 수 있음



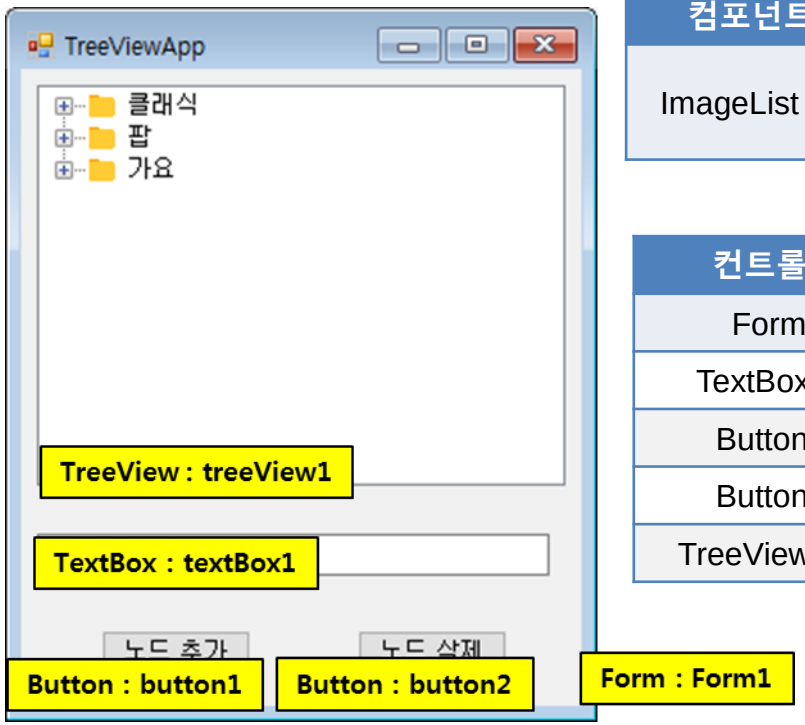
트리뷰의 노드

- 트리 노드 편집기를 통해 생성
- TreeNode 클래스의 객체
- TreeView 컨트롤의 Nodes 프로퍼티에 TreeNodeCollection 형으로 저장
- TreeNodeCollection 클래스의 메소드를 통해 노드의 편집이 가능함
  - TreeNodeCollection 클래스의 메소드

메소드	설명
Add(TreeNode node)	트리 뷰에 새로운 노드를 추가
Clear()	트리 뷰의 모든 노드를 삭제
Insert(int index, TreeNode node)	트리 뷰의 지정된 인덱스에 노드를 삽입
Remove(TreeNode node)	트리 뷰의 노드 중 매개 변수에 해당하는 노드를 삭제

TreeNode 클래스

- 트리 뷰의 노드는 TreeNode 클래스의 객체
  - TreeNodeCollection 클래스의 메소드를 사용할 경우
  - TreeNode 클래스의 객체를 생성해야 함
  - TreeNode 클래스의 생성자
    - public TreeNode(string label);
    - public TreeNode(string label, int idx1, int idx2);
      - label : 노드 이름에 해당하는 문자열
      - idx1 : 노드가 선택되지 않았을 때의 이미지 인덱스
      - idx2 : 노드가 선택되었을 때의 이미지 인덱스



컴포넌트 : (Name)	프로퍼티	인덱스	값
ImageList : imageList1	Images	0	Folder.png
		1	CD.png

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	TreeViewApp
TextBox : textBox1	Text	
Button : button1	Text	노드 추가
Button : button2	Text	노드 삭제
TreeView : treeView1	Nodes	

Node의 레이블	프로퍼티	이미지(인덱스)	선택한 이미지(인덱스)
클래식		0(Folder)	0(Folder)
	베토벤	1(CD)	1(CD)
	슈베르트	1(CD)	1(CD)
	모짜르트	1(CD)	1(CD)
팝		0(Folder)	0(Folder)
	Britney Spears	1(CD)	1(CD)
	Mariah Carey	1(CD)	1(CD)
	Capenters	1(CD)	1(CD)
가요		0(Folder)	0(Folder)
	이승환	1(CD)	1(CD)
	전인권	1(CD)	1(CD)
	이효리	1(CD)	1(CD)

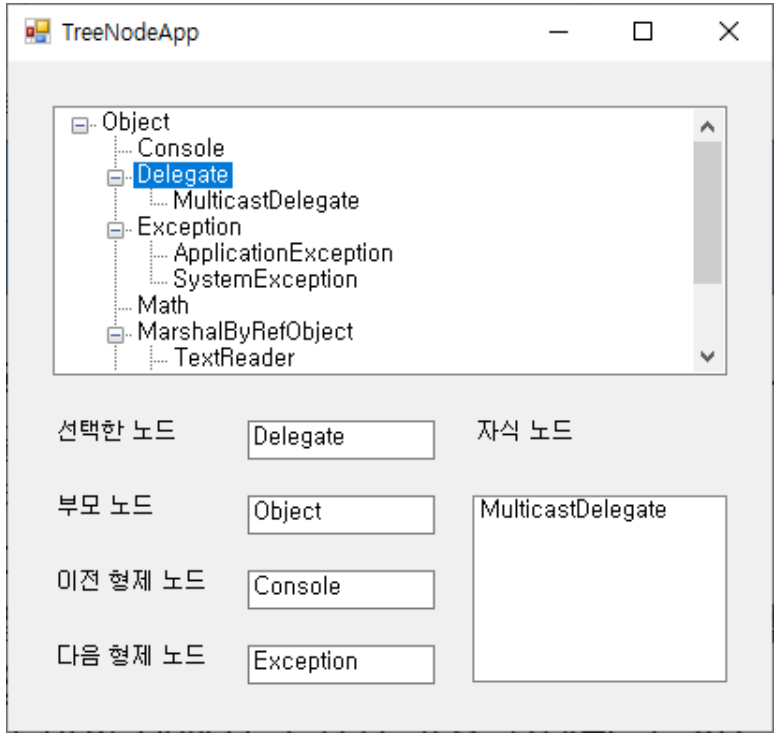
컨트롤 : (Name)	이벤트	메소드명
Form : Form1	Load	Form1_Load()
Button : button1	Click	button1_Click()
Button : button2	Click	button2_Click()

```
using System;
using System.Windows.Forms;

namespace Ex11_02_TreeViewApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "" && treeView1.SelectedNode != null)
            {
                // 선택된 노드가 있으면, 그 노드의 자식 노드로 추가한다.
                treeView1.SelectedNode.Nodes.Add(
                                new TreeNode(textBox1.Text, 1, 1));
                textBox1.Text = "";
                textBox1.Focus();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            treeView1.Nodes.Remove(treeView1.SelectedNode);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            treeView1.ExpandAll(); // 트리 뷰의 모든 노드를 펼침.
        }
    }
}
```

트리뷰 항목의 선택

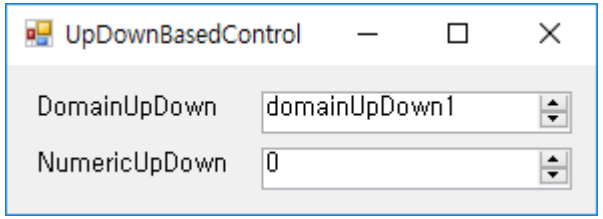
- SelectedNode 프로퍼티
  - 트리 뷰에서 선택된 노드를 저장하는 프로퍼티
  - 반환형
    - TreeNode 클래스형
  - TreeNode 클래스의 프로퍼티를 이용하면 선택된 노드를 기준으로 부모, 이전 형제, 다음 형제, 자식 노드를 참조할 수 있음
    - TreeNode 클래스의 프로퍼티

메소드	설 명
Parent	현재 트리 노드의 부모 노드
PrevNode	현재 트리 노드의 이전 형제 노드
NextNode	현재 트리 노드의 다음 형제 노드
Nodes	현재 트리 노드의 자식 노드들

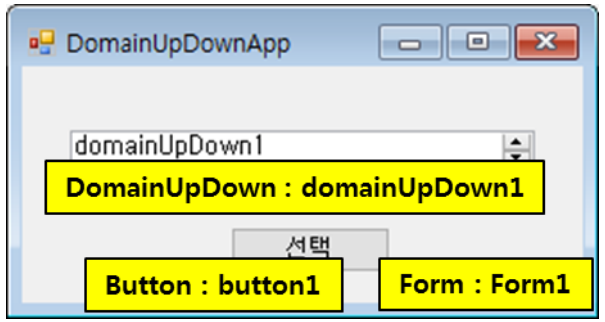


업다운 콘트롤

- 주어진 목록에서 항목을 선택할 수 있는 컨트롤
  - 업다운 버튼을 이용하여 필요한 값을 선택
  - 스핀 컨트롤(spin control)
- 영역 업다운 컨트롤
  - 문자열로 이루어진 항목에서 특정한 항목을 선택할 수 있는 컨트롤
- 수치적 업다운 컨트롤
  - 지정한 범위 내에서 수치적 값을 선택할 수 있는 컨트롤



영역 업다운 콘트롤 작성



컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	DomainUpDownApp
Button : button1	Text	선택
DomainUpDown : domainUpDown1	Items	프로그래밍언어 컴파일러 컴퓨터구성 알고리즘 데이터베이스 운영체제
	Wrap	True

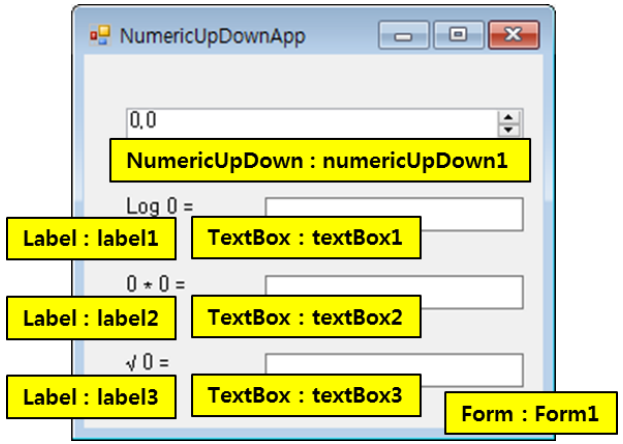
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	Button1_Click()

```
using System;
using System.Windows.Forms;

namespace Ex11_04_DomainUpDownApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(domainUpDown1.SelectedItem.ToString());
        }
    }
}
```

수치적 업다운 콘트롤 작성

- 수치적 업다운 컨트롤의 추가
  - 【도구상자】➡【NumericUpDown】을 선택하여 폼에 추가
- 수치적 업다운 컨트롤의 항목에 대한 범위와 증가/감소량을 설정
  - 수치적 업다운 컨트롤의 프로퍼티를 통해 설정

프로퍼티	설 명
Minimum	수치적 업다운 컨트롤의 최소 값.
Maximum	수치적 업다운 컨트롤의 최대 값.
Increment	수치적 업다운 컨트롤의 증가/감소 양.
Value	수치적 업다운 컨트롤의 현재 값.
DecimalPlaces	수치적 업다운 컨트롤에 표시할 소수 자릿수.
ThousandsSeparator	10진수 3자리마다 구분 기호를 삽입 여부.
Hexadecimal	수치적 업다운 컨트롤의 값을 16진수로 표시.



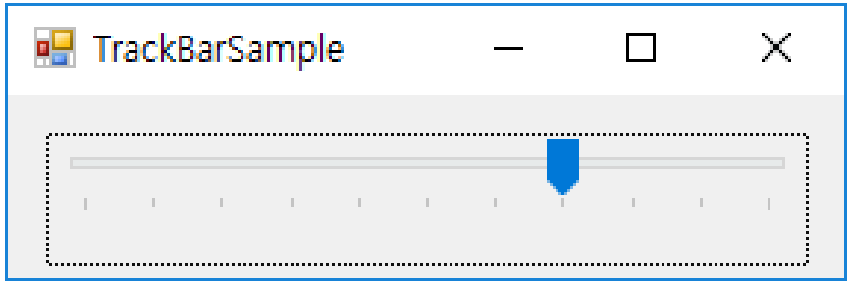
컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	NumericUpDownApp
Label : label1	Text	Log 0 =
Label : label2	Text	0 * 0 =
Label : label3	Text	√ 0 =
TextBox : textBox1	Text	
TextBox : textBox2	Text	
TextBox : textBox3	Text	
NumericUpDown : numericUpDown1	Minimum	0
	Maximum	1000
	Increment	0.5
	DecimalPlaces	1

컨트롤 : (Name)	이벤트	메소드명
NumericUpDown : numericUpDown1	ValueChanged	numericUpDown1_ValueChanged()

```
using System;
using System.Windows.Forms;

namespace Ex11_05_NumericUpDownApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            decimal d = numericUpDown1.Value;
            label1.Text = "Log " + d + " = ";
            textBox1.Text = System.Math.Log10((double)d).ToString();
            label2.Text = d + "*" +d + " = ";
            textBox2.Text = System.Math.Pow((double)d, 2).ToString();
            label3.Text = "√" + d + " = ";
            textBox3.Text = System.Math.Sqrt((double)d).ToString();
        }
    }
}
```

트랙바

- 범위 내에서 값을 선택할 수 있는 컨트롤
  - 슬라이더와 눈금으로 구성



  - 슬라이더의 이동
    - 마우스 드래그
    - 슬라이더의 좌우 공간 클릭
    - 마우스 휠의 회전
    - 키보드의 좌우 방향키, 페이지 업다운키

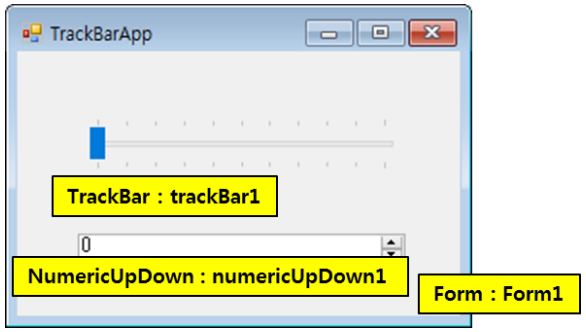
트랙바 작성

- 트랙 바의 추가
  - 【도구상자】➡【TrackBar】를 선택하여 폼에 추가
- 트랙 바의 값에 대한 범위와 이동량을 설정
  - 트랙바의 프로퍼티를 통해 설정

프로퍼티	설 명
Minimum	트랙 바의 최소 값
Maximum	트랙 바의 최대 값
Value	트랙 바의 현재 값
LargeChange	마우스 클릭이나 PageUp/PageDown 키에 대한 이동량
SmallChange	마우스 휠의 회전이나 키보드의 방향키에 대한 이동량
TickFrequency	눈금이 표시되는 값의 범위
TickStyle	트랙 바에 눈금이 표시되는 위치
Orientation	트랙 바의 방향(Horizontal | Vertical)

- 슬라이더 형태와 눈금이 표시되는 위치 설정
  - TickStyle 프로퍼티에 TickStyle 열거형 값을 배정하여 설정
  - TickStyle 열거형

기호상수	슬라이더	설 명
None		눈금을 표시하지 않음.
TopLeft		트랙 바의 Orientation 프로퍼티가 Horizontal로 설정된 경우 슬라이더의 상단에 눈금 표시. 트랙 바의 Orientation 프로퍼티가 Vertical로 설정된 경우 슬라이더의 좌측에 눈금 표시.
BottomRight		트랙 바의 Orientation 프로퍼티가 Horizontal로 설정된 경우 슬라이더의 하단에 눈금 표시. 트랙 바의 Orientation 프로퍼티가 vertical로 설정된 경우 슬라이더의 우측에 눈금 표시.
Both		슬라이더의 양쪽에 눈금 표시.



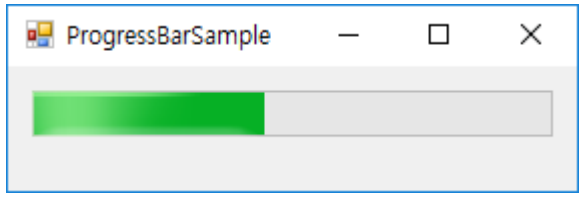
컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	TrackBarApp
TrackBar : trackBar1	Minimum	0
	Maximum	100
	LargeChange	5
	SmallChange	1
	TickFrequency	10
	TickStyle	Both
	Orientation	Horizontal
NumericUpDown : numericUpDown1	Minimum	0
	Maximum	100

컨트롤 : (Name)	이벤트	메소드명
TrackBar : trackBar1	Scroll	trackBar1_Scroll()
NumericUpDown : numericUpDown1	ValueChanged	numericUpDown1_ValueChanged()

```
using System;;
using System.Windows.Forms;

namespace Ex11_06_TrackBarApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            numericUpDown1.Value = trackBar1.Value;
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            trackBar1.Value = (int)numericUpDown1.Value;
        }
    }
}
```

프로그레스바

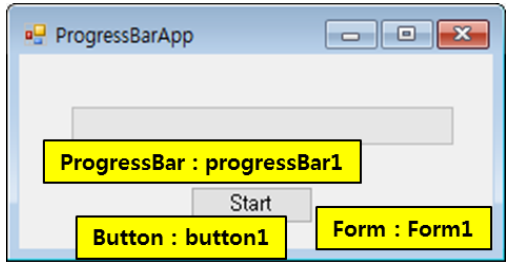
- 작업의 진행상황을 보여주는 컨트롤
  - 좌측에서 우측으로 사각형의 조각을 채우면서 진행
  - 애플리케이션의 설치과정이나 파일 복사과정에서 사용



프로그레스바 작성

- 프로그레스 바의 추가
  - 【도구상자】➡【ProgressBar】를 선택하여 폼에 추가

- 프로그레스 바의 값에 대한 범위를 설정
  - 프로그레스 바의 프로퍼티를 통해 설정
    - Maximum
      - 프로그레스 바의 최대값
    - Minimum
      - 프로그레스 바의 최소값



컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ProgressBarApp
Button : button1	Text	Start
ProgressBar : progressBar1	Minimum	0
	Maximum	100000

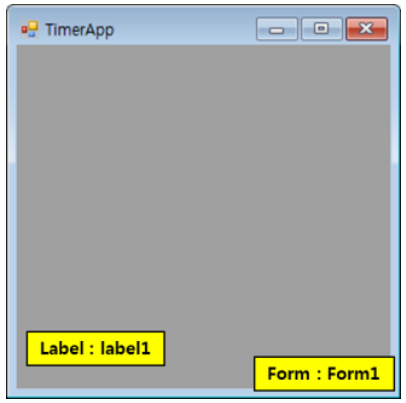
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	button1_Click()

```
using System;
using System.Windows.Forms;

namespace Ex11_07_ProgressBarApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            for (int i = progressBar1.Minimum; i < progressBar1.Maximum; i++)
                progressBar1.Value = i;
        }
    }
}
```

타이머

- 주기적인 간격으로 이벤트를 발생시키는 컴포넌트
  - 배경작업을 처리할 때 주로 사용
  - 일정한 간격에 따라 Tick 이벤트를 발생
    - Interval 프로퍼티를 통해 간격을 설정
    - 밀리 초(millisecond, 1/1000초)를 사용
  - 주기적으로 발생시키기 위해서는 Enable 프로퍼티를 참으로 설정
  - 항상 Interval 프로퍼티의 간격에 따라 Tick 이벤트가 발생하는 것은 아님
    - Tick 이벤트가 다른 이벤트에 비해 우선순위가 낮기 때문
  - 타이머 컴포넌트의 추가
    - 【도구상자】➡【Timer】를 선택하여 폼에 추가

https://opengameart.org/content/game-character-blue-flappy-bird-sprite-sheets



컴포넌트 : (Name)	프로퍼티	값
Timer : timer1	Enable	True
	Interval	100
ImageList : imageList1	Images	frame-1.png frame-2.png frame-3.png frame-4.png frame-5.png frame-6.png frame-7.png frame-8.png

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	TimerApp
Label : label1	Text	
	Dock	Fill
	BackColor	ButtonShadow

컴포넌트 : (Name)	이벤트	메소드명
Timer : timer1	Tick	timer1_Tick()

```
using System;
using System.Windows.Forms;

namespace Ex11_08_TimerApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private int index = 0;
        private void timer1_Tick(object sender, EventArgs e)
        {
            index %= imageList1.Images.Count;
            label1.Image = imageList1.Images[index++];
        }
    }
}
```

Reference

- ✓ C# 프로그래밍 입문, 오세만 외4, 생능출판
- ✓ 초보자를 위한 C# 200제, 강병익, 정보문화사
- ✓ 프랙티컬 C#, 이데이 히데유키, 김범준, 위키북스
- ✓ C#언어 프로그래밍 바이블, 김명렬 외1, 홍릉과학출판사
- ✓ C# and the .NET Platform, Andrew Troelsen, 장시혁, 사이텍미디어
- ✓ https://docs.microsoft.com/ko-kr/learn/browse/?products=dotnet&terms=c%23