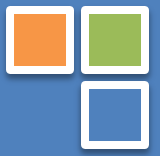


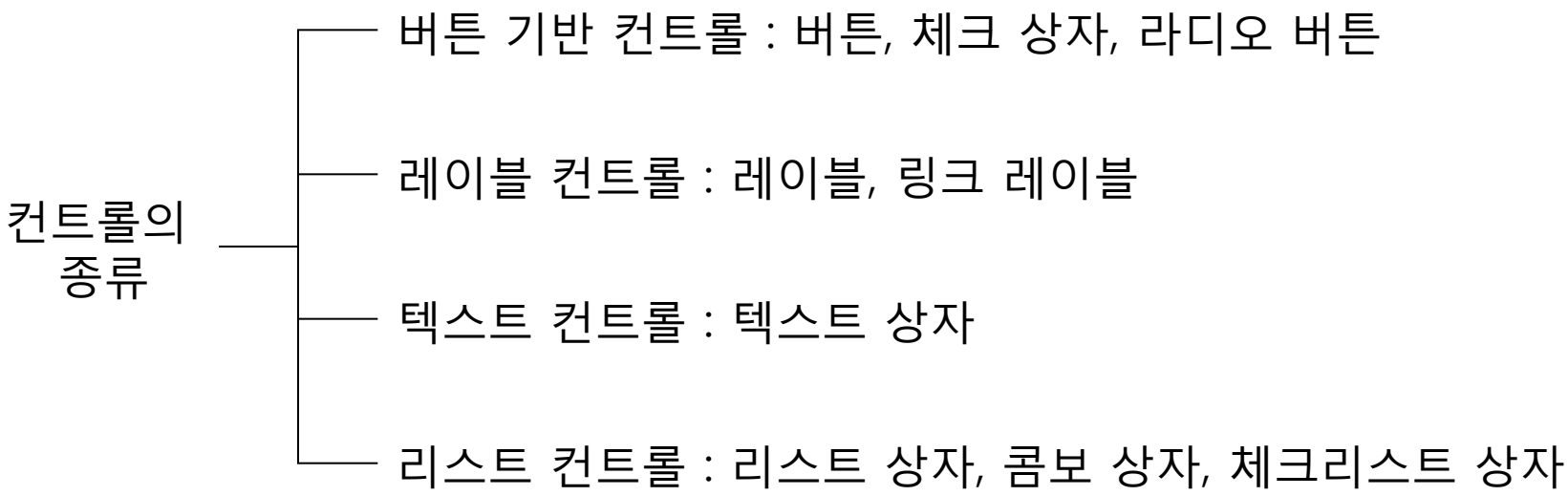
C# 입문 : 이론과 실습

제 8장 컨트롤

목차

컨트롤이란?

- 컨트롤
  - 화면에 표시되어 사용자와 상호작용을 수행하는 컴포넌트를 의미.
  - 종류 : 버튼, 레이블, 텍스트, 리스트



버튼 기반 컨트롤

- 버튼 기반 컨트롤
  - ButtonBase 추상 클래스를 상속받은 컨트롤을 의미.
- 버튼 기반 컨트롤의 종류
  - 버튼
  - 체크 상자
  - 라디오 버튼
- 버튼 기반 컨트롤의 기본적인 형태

버튼 [1/5]

- 버튼
  - 사용자 입력을 받는 가장 간단한 방법으로 버튼을 클릭함으로써 이벤트를 발생.

- 버튼을 폼에 추가하는 방법
  - 【도구상자】➡【Button】을 클릭한 후 드래그-앤-드롭으로 폼의 원하는 위치에 배치.
  - 【도구상자】➡【Button】을 더블 클릭하면 폼의 좌측 상단에 버튼이 추가되고 이를 원하는 위치로 이동.

버튼 [2/5]

- 버튼의 프로퍼티
  - 이름과 글자색등을 설정할 수 있음.
  - 버튼의 프로퍼티

프로퍼티	설명
Text	버튼 이름을 설정.
Font	버튼 이름에 해당하는 문자열의 글꼴을 설정.
ForeColor	버튼 이름에 해당하는 문자열의 색상을 설정.
TextAlign	버튼 이름에 해당하는 문자열을 정렬.
FlatStyle	버튼의 스타일(모양)을 설정.
Size	버튼의 크기(픽셀 단위)를 설정
BackColor	버튼의 배경색을 설정.
BackgroundImage	버튼의 배경 이미지를 설정.
Image	버튼 형태 위에 이미지를 설정.
ImageAlign	버튼 형태 위에 이미지를 정렬.

버튼 [3/5]

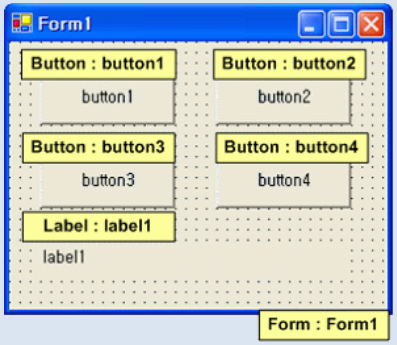
- 버튼의 스타일
  - 버튼의 스타일을 설정하기 위해서는 FlatStyle 프로퍼티를 지정해야 함.
  - System.Windows.Forms 네임스페이스에 포함되어 있음.
  - 4가지 종류가 있음.
    - FlatStyle 열거형

기호상수	버튼모양	설명
Flat		버튼의 모양을 평면으로 표시한다.
Standard		버튼의 모양을 입체적으로 표시한다.
Popup		버튼의 모양이 처음에는 평면으로 표시되었다가 마우스 포인터가 버튼 위로 이동하면 입체적으로 변경한다.
System		버튼의 모양을 시스템이 결정. 일반적으로 System으로 지정한 버튼의 모양은 Standard로 지정한 것과 같다.

버튼 [4/5]

[예제 8.1 – ButtonApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ButtonApp
Button : Button1	FlatStyle, Text	Flat
Button : Button2	FlatStyle, Text	Popup
Button : Button3	FlatStyle, Text	Standard
Button : Button4	FlatStyle, Text	System

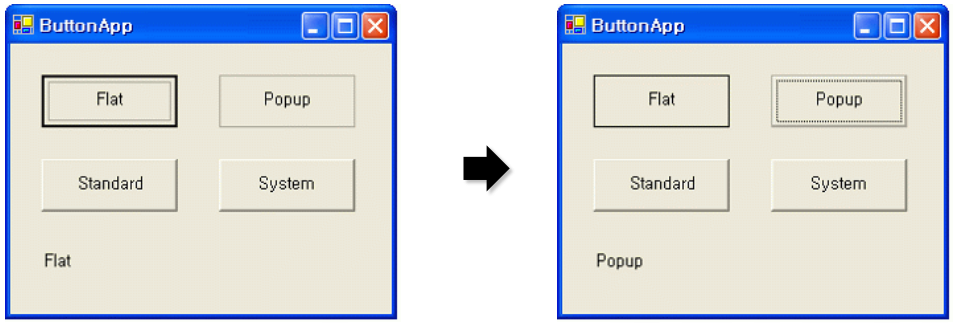
3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
Button : Button1	Click	Button1_Click()
Button : Button2	Click	Button2_Click()
Button : Button3	Click	Button3_Click()
Button : Button4	Click	Button4_Click()

버튼 [5/5]

```
private void button1_Click(object sender, EventArgs e) {
    label1.Text = FlatStyle.Flat.ToString();
}
private void button2_Click(object sender, EventArgs e) {
    label1.Text = FlatStyle.Popup.ToString();
}
private void button3_Click(object sender, EventArgs e) {
    label1.Text = FlatStyle.Standard.ToString();
}
private void button4_Click(object sender, EventArgs e) {
    label1.Text = FlatStyle.System.ToString();
}
```

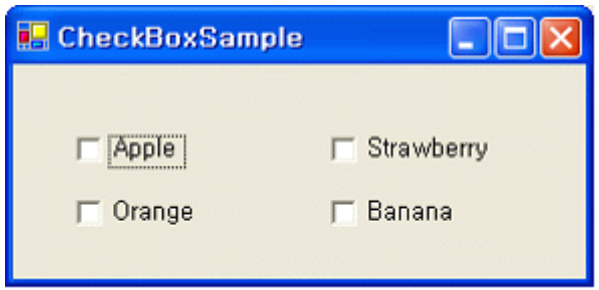
실행 방법 : 각각의 버튼을 클릭한다.
실행 결과 :



체크상자 [1/5]

- 체크상자
  - 주어진 항목들 중에서 선택할 수 있는 컨트롤
  - 특징
    - 주어진 항목을 복수로 선택할 수 있음.

- 체크상자의 기본적인 형태



체크상자 [2/5]

- 체크 상자의 프로퍼티
  - Control 클래스를 상속받은 컨트롤이기 때문에 버튼과 대부분 동일.
  - 체크 상자의 체크 상태를 나타내는 Checked 프로퍼티가 추가로 제공.

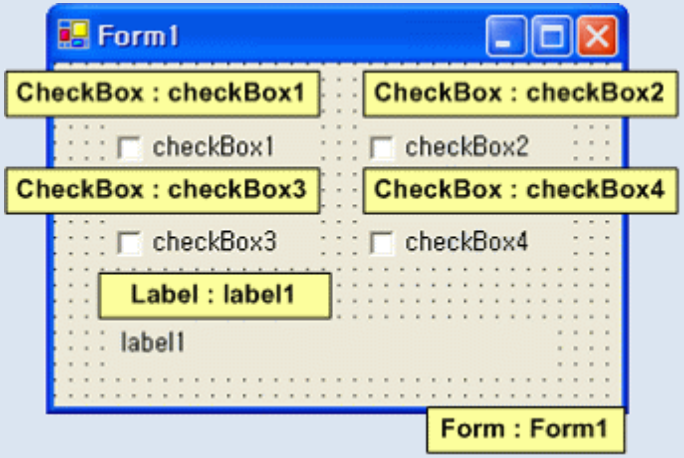
- Checked 프로퍼티
  - 참으로 설정하면 네모부분에 '✔'가 표시.
  - Checked 프로퍼티의 값이 변경될 때마다 CheckedChanged 이벤트 발생.

체크상자 [3/5]

- 예제 8.2
  - 4개의 항목을 갖는 체크 상자를 만들고 선택된 항목의 이름을 출력하는 예제.

[예제 8.2 – CkeckBoxApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	CheckBoxApp
CheckBox : checkBox1	Text	Apple
	Checked	True
CheckBox : checkBox2	Text	Strawberry
	Checked	False
CheckBox : checkBox3	Text	Orange
	Checked	False
CheckBox : checkBox4	Text	Banana
	Checked	False
Label : label1	Text	Apple

체크상자 [4/5]

3) 멤버

```
private string strTemp;
private void UpdateLabel(string s, bool b) {
    if(b) { label1.Text += s;
    } else {
        strTemp = label1.Text;          int i = strTemp.IndexOf(s.Substring(0, 1));
        int j = i + s.Length;           label1.Text = strTemp.Remove(i, j – i);
    }
}
```

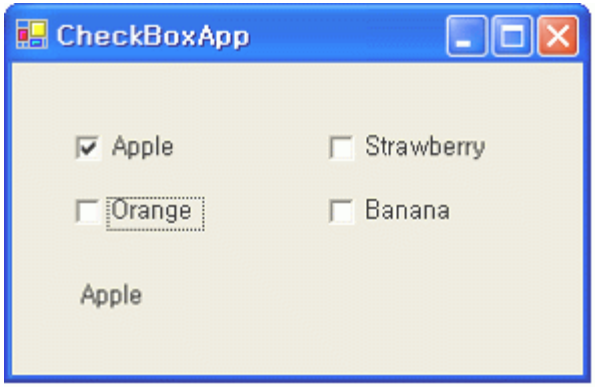
4) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
CheckBox : checkBox1	CheckedChange	checkBox1_CheckedChanged()
CheckBox : checkBox2	CheckedChange	checkBox2_CheckedChanged()
CheckBox : checkBox3	CheckedChange	checkBox3_CheckedChanged()
CheckBox : checkBox4	CheckedChange	checkBox4_CheckedChanged()

```
private void checkbox1_CheckedChanged(object sender, EventArgs e) {
    UpdateLabel(checkBox1.Text, checkBox1.Checked);    }
private void checkbox2_CheckedChanged(object sender, EventArgs e) {
    UpdateLabel(checkBox2.Text, checkBox2.Checked);    }
private void checkbox3_CheckedChanged(object sender, EventArgs e) {
    UpdateLabel(checkBox3.Text, checkBox3.Checked);    }
private void checkbox4_CheckedChanged(object sender, EventArgs e) {
    UpdateLabel(checkBox4.Text, checkBox4.Checked);    }
```

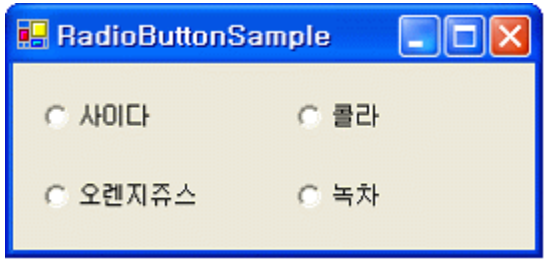
체크상자 [5/5]

실행 방법 : 체크 상자를 클릭하여 선택하거나 해제한다.
실행 결과 :



라디오 버튼 [1/3]

- 라디오 버튼
  - 주어진 항목들 중에서 오직 한 개만을 선택할 수 있는 컨트롤
- 라디오 버튼의 기본적인 형태



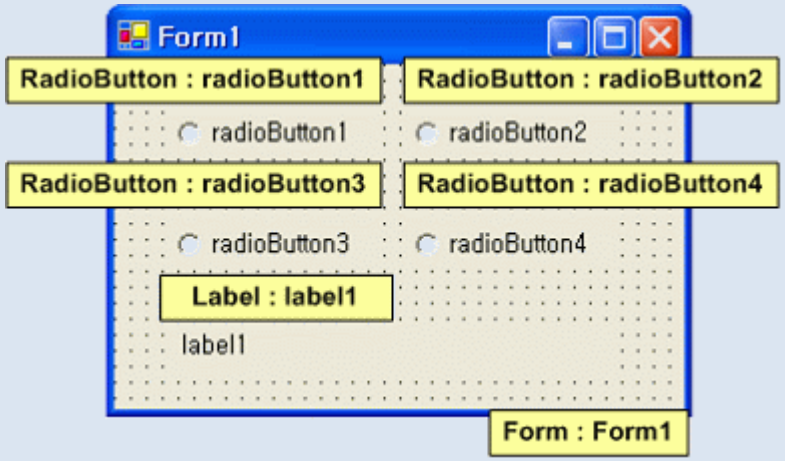
- 라디오 버튼의 프로퍼티
  - 체크 상자와 동일

라디오 버튼 [2/3]

- 예제 8.3
  - 예제 8.2의 체크 상자를 라디오 버튼으로 대체한 예제

[예제 8.3 – RadioButtonApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	RadioButtonApp
RadioButton : radioButton1	Text	사이다
	Checked	True
RadioButton : radioButton2	Text	콜라
	Checked	False
RadioButton : radioButton3	Text	오렌지쥬스
	Checked	False
RadioButton : radioButton4	Text	녹차
	Checked	False
Label : label1	Text	사이다

라디오 버튼 [3/3]

3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
RadioButton : radioButton1	CheckedChange	radioButton1_CheckedChanged()
RadioButton : radioButton2	CheckedChange	radioButton2_CheckedChanged()
RadioButton : radioButton3	CheckedChange	radioButton3_CheckedChanged()
RadioButton : radioButton4	CheckedChange	radioButton4_CheckedChanged()

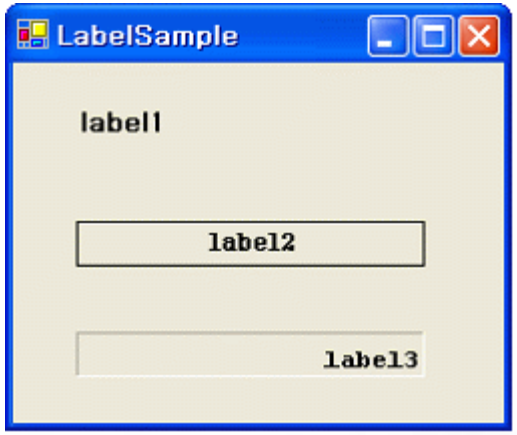
```
private void radioButton1_CheckedChanged(object sender, EventArgs e) {
    label1.Text = radioButton1.Text;  }
private void radioButton2_CheckedChanged(object sender, EventArgs e) {
    label1.Text = radioButton2.Text;  }
private void radioButton3_CheckedChanged(object sender, EventArgs e) {
    label1.Text = radioButton3.Text;  }
private void radioButton4_CheckedChanged(object sender, EventArgs e) {
    label1.Text = radioButton4.Text;  }
```

실행 방법 : 라디오 버튼을 클릭하여 선택

실행 결과 :

레이블 [1/5]

- 레이블
  - 각종 정보를 폼에 표시할 때 사용하는 컨트롤
  - 레이블의 용도
    1. 폼에 있는 컨트롤을 식별하는 정보를 표시.
    2. 애플리케이션의 실행에 대한 응답 정보를 표시.
    3. 특정 컨트롤을 클릭했을 때 실행되는 작업을 설명하는 메시지를 표시.
- 레이블의 기본적인 형태



레이블 [2/5]

- 레이블의 프로퍼티
  - Font, BorderStyle, TextAlign

- BorderStyle
  - 레이블의 테두리를 설정할 때 사용하는 프로퍼티
  - BorderStyle 열거형

기호상수	순서 값	설명
None	0	테두리가 없음
FixedSingle	1	단일 선 테두리
Fixed3D	2	3차원 테두리

레이블 [3/5]

- TextAlign
  - 레이블과 같은 컨트롤에서 표시되는 문자열에 대한 정렬을 위한 프로퍼티
  - System.Drawing 네임스페이스에 포함된 ContentAlignment 열거형을 사용하여 지정할 수 있음.
    - ContentAlignment 열거형

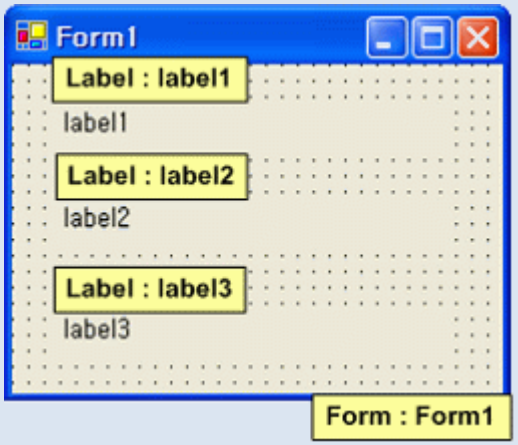
기호상수	순서 값	설명
TopLeft	0x0001	좌측 상단 정렬
TopCneter	0x0002	가운데 상단 정렬
TopRight	0x0004	우측 상단 정렬
MiddleLeft	0x0010	좌측 중앙 정렬
MiddleCenter	0x0020	가운데 중앙 정렬
MiddleRight	0x0040	우측 중앙 정렬
BottomLeft	0x0100	좌측 하단 정렬
BottomCenter	0x0200	가운데 하단 정렬
BottomRight	0x0400	우측 중앙 정렬

레이블 [4/5]

- 예제 8.4
  - 3개의 테두리와 9개의 문자열 정렬 중 대표적인 3개의 정렬을 사용한 예제.

[예제 8.4 – LabelApp.cs]

1) 폼 설계

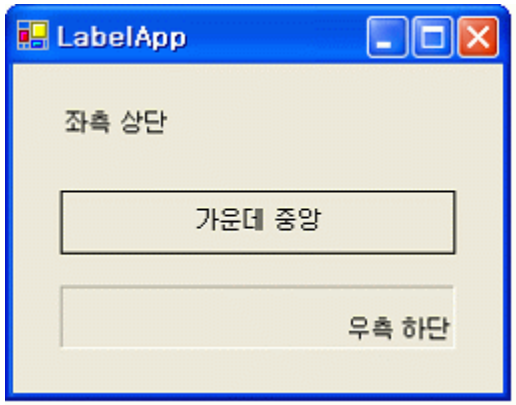


2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	LabelApp
Label : label1	Text	좌측 상단
	BorderStyle	None
	TextAlign	TopLeft
Label : label2	Text	가운데 중앙
	BorderStyle	FixedSingle
	TextAlign	MiddleCenter
Label : label3	Text	우측 하단
	BorderStyle	Fixed3D
	TextAlign	BottomRight

레이블 [5/5]

실행 결과 :



링크 레이블 [1/3]

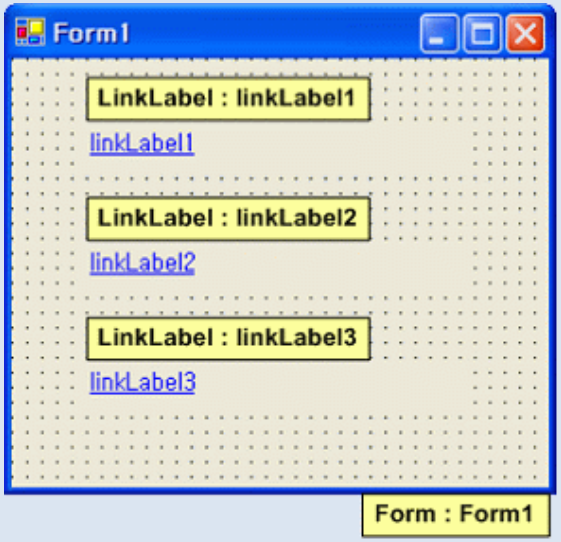
- 링크 레이블
  - 하이퍼링크(hyper link)를 설정할 수 있는 레이블 컨트롤
  - 레이블 컨트롤을 상속받았기 때문에 레이블의 기본적인 기능을 모두 가지고 있음.
  - 링크 레이블은 링크를 클릭했을때 발생하는 LinkClicked 이벤트에 대한 이벤트 처리기를 작성.
    - 이벤트 처리를 위해서는 **Process** 클래스의 정적 메소드인 **Start()** 메소드를 사용함.
- 링크 레이블의 기본적인 형태

링크 레이블 [2/3]

- 예제 8.5
  - 웹 브라우저를 실행 링크 레이블, 이메일 클라이언트를 실행하는 링크 레이블과 메모장 프로그램을 실행하는 링크 레이블을 구현하는 예제.

[예제 8.5 – LinkLabelApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	LinkLabelApp
LinkLabel : linkLabel1	Text	http://plac.dongguk.ac.kr
LinkLabel : linkLabel2	Text	mailto:s4064@dongguk.edu
LinkLabel : linkLabel3	Text	C:\log.txt

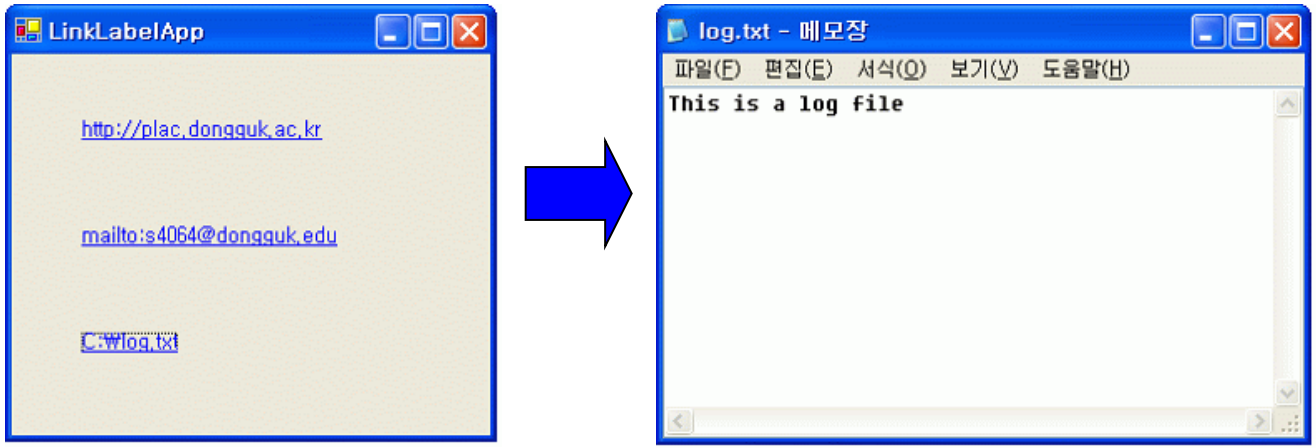
링크 레이블 [3/3]

3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
LinkLabel : linkLabel1	LinkClicked	linkLabel1_LinkClicked()
LinkLabel : linkLabel2	LinkClicked	linkLabel2_LinkClicked()
LinkLabel : linkLabel3	LinkClicked	linkLabel3_LinkClicked()

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) {
    Process.Start(linkLabel1.Text);  }
private void linkLabel2_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) {
    Process.Start(linkLabel2.Text);  }
private void linkLabel3_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) {
    Process.Start(linkLabel3.Text);  }
```

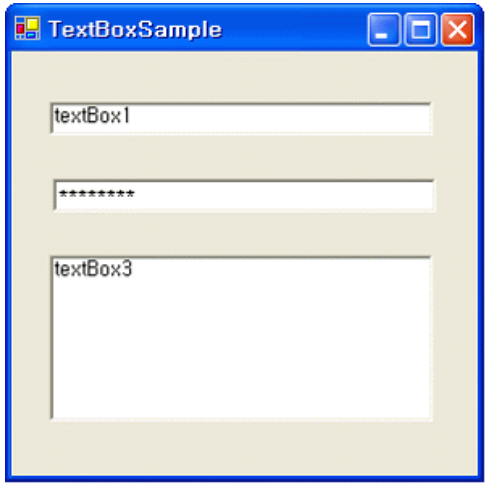
실행 방법 : 각각의 링크를 클릭
실행 결과 :



텍스트 상자 [1/4]

- 텍스트 상자
  - 사용자가 직접 텍스트를 입력할 수 있는 기본적인 컨트롤로서 사용자로부터 값을 입력 받거나 애플리케이션의 실행 결과를 출력하고자 할 때 유용하게 사용할 수 있음.

- 텍스트 상자의 기본적인 형태



텍스트 상자 [2/4]

- 텍스트 상자의 프로퍼티

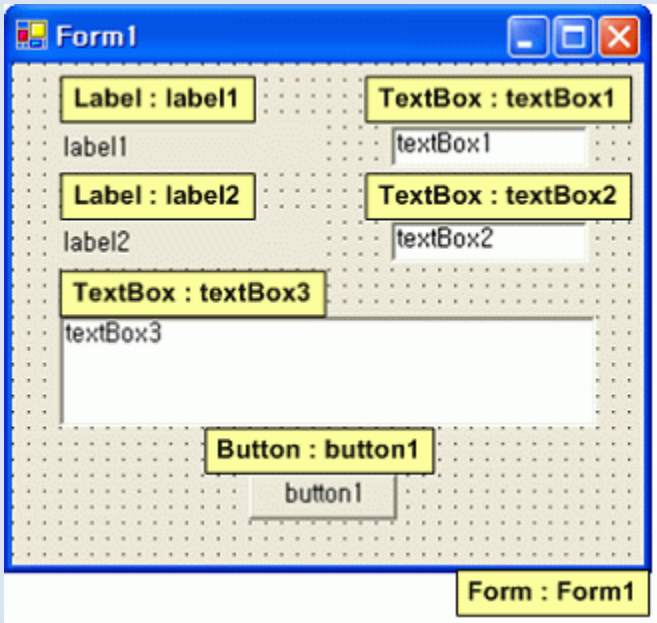
프로퍼티	설명
MaxLength	텍스트 상자에 입력할 수 있는 최대 문자 수를 설정.
Multiline	텍스트 상자의 영역을 여러 줄로 설정.
PasswordChar	텍스트 상자의 암호 입력에 사용할 문자를 설정.
ReadOnly	텍스트 상자의 텍스트를 변경하지 못하도록 설정.
WordWrap	텍스트 상자의 텍스트가 영역을 초과할 경우 자동으로 줄을 바꾸도록 설정.
ScrollBars	Multiline 프로퍼티가 참인 경우 텍스트 상자에 스크롤 바를 설정 (None | Horizontal | Vertical | Both)

텍스트 상자 [3/4]

- 예제 8.6
  - 사용자 이름과 패스워드를 입력받아 출력하는 예제.

[예제 8.6 – TextBoxApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	TextBoxlApp
Label : label1	Text	Enter your name
Label : label2	Text	Enter your password
TextBox : textBox1	Text	
TextBox : textBox2	Text	
	PasswordChar	*
TextBox : textBox3	Text	
	Multiline	True
Button : button1	Text	OK

텍스트 상자 [4/4]

3) 이벤트 처리기

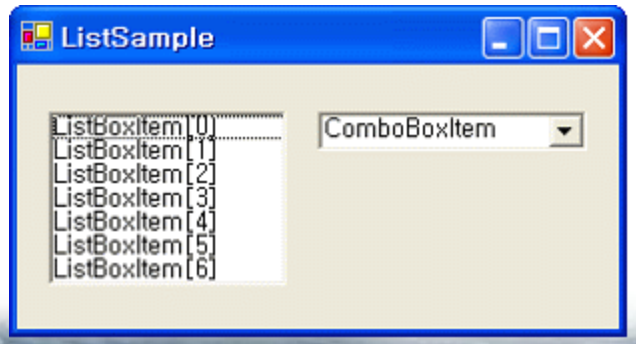
컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	Button1_Click()

```
private void button1_Click(object sender, EventArgs e) {
    textBox3.Text  = "Name : " + textBox1.Text + "₩r₩nPassword : " + textBox2.Text;
}
```

실행 방법 : textBox1과 textBox2에 이름과 비밀번호를 입력한 후, OK를 클릭한다.
실행 결과 :

리스트

- 리스트 기반 컨트롤
  - ListControl 클래스를 상속받은 컨트롤을 가리킴.
- 리스트 기반 컨트롤의 종류
  - 리스트 상자
  - 콤보 상자
  - 체크리스트 상자
- 리스트 기반 컨트롤의 기본적인 형태



리스트 상자 [1/5]

- 리스트 상자
  - 사용자가 선택할 수 있는 항목들의 목록을 표시해 주는 컨트롤

- 리스트 상자의 프로퍼티

프로퍼티	설명
Item	리스트 상자의 항목을 설정.
MultiColumn	리스트 상자가 여러 열을 표시할 수 있도록 설정.
ScrollAlwaysVisible	항상 세로 스크롤바가 표시되도록 설정.
SelectionMode	리스트 상자에서 항목을 선택하는 방법을 설정.
SelectionIndex	리스트 상자에서 현재 선택된 아이템이 인덱스를 반환(0부터 시작).
SelectionItem	리스트 상자에서 현재 선택된 아이템을 반환.

리스트 상자 [2/5]

- 리스트 상자의 항목을 선택하는 방법
  - SelectionMode 프로퍼티 값에 따라 변경
  - SelectionMode 열거형

기호상수	설명
None	항목을 선택할 수 없음.
One	하나의 항목만 선택할 수 있음.
MultiSimple	여러 항목을 선택할 수 있음.
MultiExtented	여러 항목을 선택할 수 있으며, <Shift>, <Ctrl>, 마우스 포인터를 이용해 선택할 수 있음.

- 리스트 상자의 항목
  - Items 프로퍼티를 통하여 추가 및 삭제 가능.
    - 문자열 컬렉션 편집기
    - ObjectCollection 클래스 메소드

리스트 상자 [3/5]

- ObjectCollection 클래스의 메소드

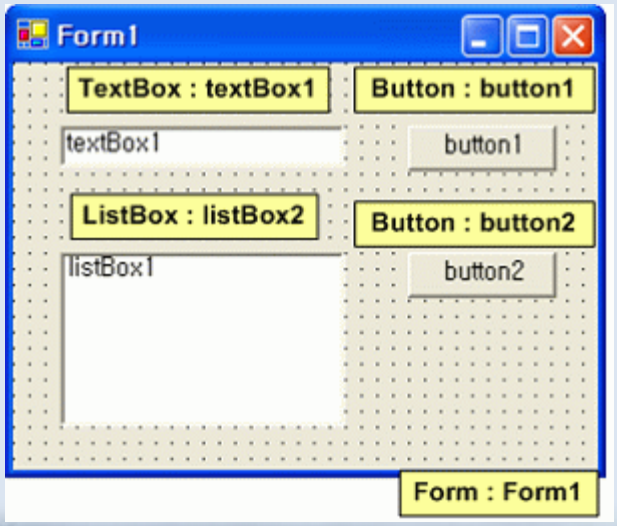
메소드	설명
Add(object item)	리스트 상자에 항목을 추가한다.
Clear()	리스트 상자의 모든 항목을 제거한다.
FindString(string s, int index)	리스트 상자의 항목 중 지정된 인덱스 다음부터 지정된 문자열로 시작하는 항목의 인덱스를 반환한다.
IndexOf(object item)	지정한 항목의 인덱스를 반환한다.
Insert(int index, object item)	지정된 인덱스에 항목을 추가한다.
Remove(object item)	지정된 항목을 제거한다.
RemoveAt(int index)	지정된 인덱스의 항목을 제거한다.

리스트 상자 [4/5]

- 예제 8.7
  - 텍스트 상자를 통해 입력받은 텍스트를 ObjectCollection 클래스의 메소드를 사용하여 리스트 상자에 추가하고 리스트 상자에서 선택된 항목을 삭제하는 예제.

[예제 8.7 – ListBoxApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ListBoxApp
TextBox : textBox1	Text	
ListBox : listBox1	Items	C C++ JAVA C#
Button : button1	Text	추가
Button : button2	Text	삭제

리스트 상자 [5/5]

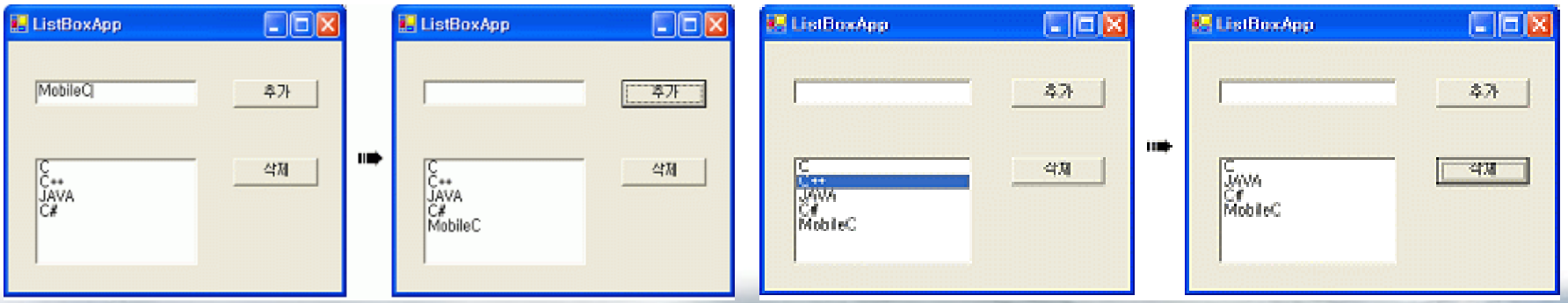
4) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	Button1_Click()
Button : button2	Click	Button2_Click()

```
private void button1_Click(object sender, EventArgs e) {
    if (textBox1.Text != " ") {
        listBox1.Items.Add(textBox1.Text);
        textBox1.Text = "":   } }
private void button2_Click(object sender, EventArgs e) {
    if (listBox1.SelectedIndex > -1)
        listBox1.Items.Remove(listBox1.SelectedIndex);  }
```

실행 방법 : ① 텍스트 상자에 문자열을 입력한 후, 추가 버튼을 클릭한다.
② 리스트 상자의 항목을 선택한 후, 삭제 버튼을 클릭한다.

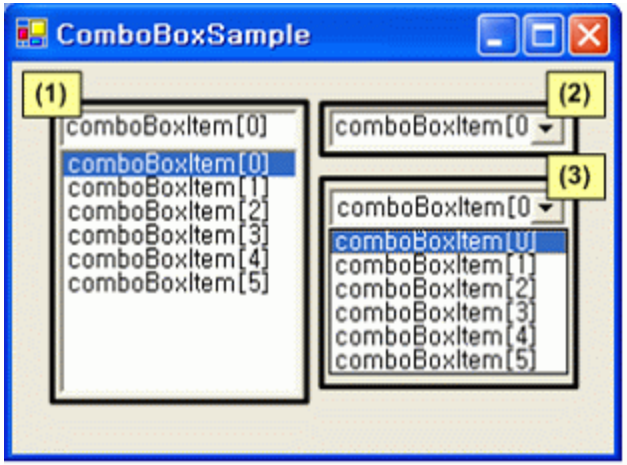
실행 결과 :



콤보 상자 [1/4]

- 콤보 상자
  - 사용자가 상자를 클릭하면 목록이 나타나는 드롭다운(drop-down) 형식의 컨트롤

- 콤보 상자의 기본적인 형태



콤보 상자 [2/4]

- DropDownStyle
  - 콤보 상자의 형태를 결정.
  - ComboBoxStyle을 열거형으로 가짐.
  - ComboBoxSytle 열거형

기호상수	설명
Simple	[그림 8.11]의 (1)과 같이 선택 항목을 항상 볼 수 있음.
DropDown	[그림 8.11]의 (2)와 같이 화살표 버튼을 클릭해야 선택항목을 볼 수 있음.
DropDownList	[그림 8.11]의 (3)고 k같이 화살표 버튼뿐만 아니라 텍스트 부분을 클릭하여도 선택 항목을 볼 수 있음.

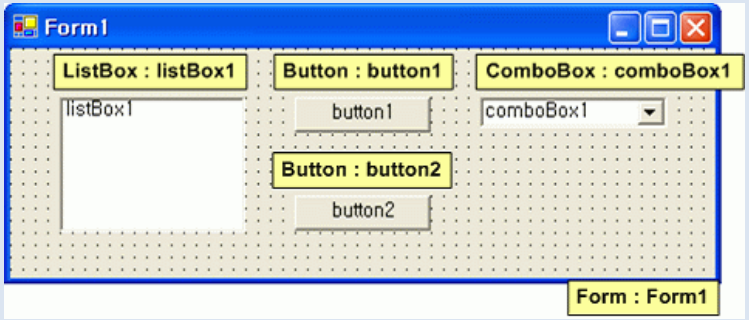
- 콤보 상자의 항목
  - Items 프로퍼티를 통하여 추가 및 삭제 가능.
    - 문자열 컬렉션 편집기
    - ObjectCollect 클래스 메소드

콤보 상자 [3/4]

- 예제 8.8
  - 리스트 상자와 콤보 상자를 폼에 배치한 후 두 개의 버튼을 이용하여 리스트 상자와 콤보 상자 사이에 항목을 주고 받을 수 있는 예제.

[예제 8.8 – ComboBoxApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	ComboBoxApp
ListBox : listBox1	Items	Button CheckBox RadioButton Lable
ComboBox : comboBox1	Items	LinkedLabel TextBox ListBox ComboBox
Button : button1	Text	>>
Button : button2	Text	<<

콤보 상자 [4/4]

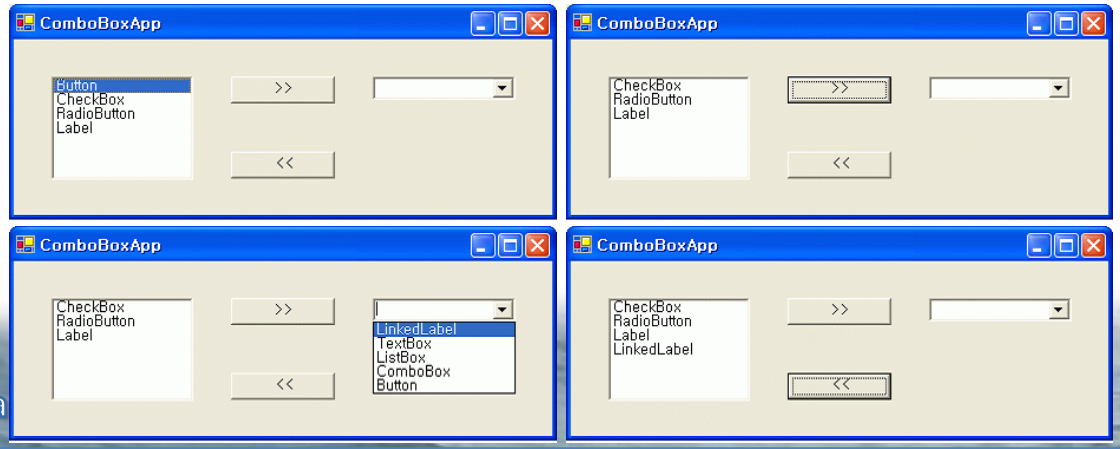
4) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
Button : button1	Click	Button1_Click()
Button : button2	Click	Button2_Click()

```
private void button1_Click(object sender, EventArgs e) {
    if (listBox1.SelectedItem != null) {
        comboBox1.Items.Add(listBox1.SelectedItem);
        listBox1.Item.Remove(listBox1.SelectedItem):  }  }
private void button2_Click(object sender, EventArgs e) {
    if (comboBox1.SelectedItem != null) {
        listBox1.Items.Add(comboBox1.SelectedItem);
        comboBox1.Item.Remove(comboBox1.SelectedItem):  }  }
```

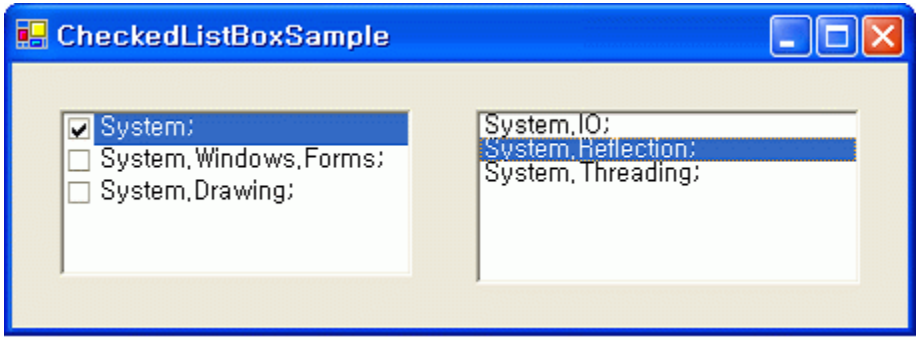
실행 방법 : ① 리스트 상자의 항목을 선택한 후, >> 버튼을 클릭한다.
② 콤보 상자의 항목을 선택한 후, << 버튼을 클릭한다.

실행 결과 :



체크리스트 상자 [1/4]

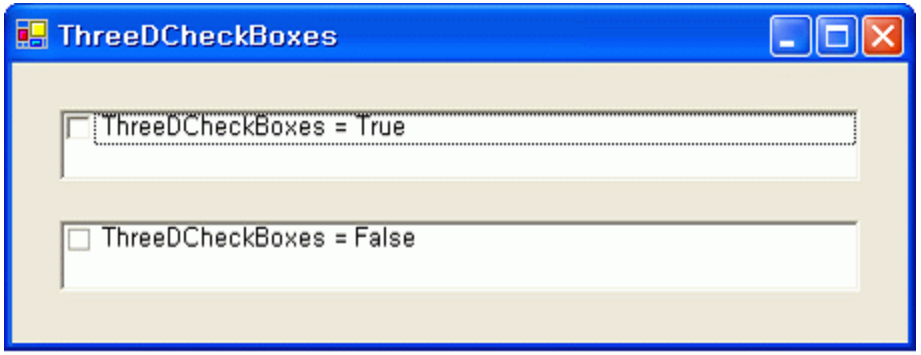
- 체크 리스트 상자
  - 리스트 상자의 항목에 체크 상자를 추가한 형태의 컨트롤
- 체크 리스트 상자의 기본적인 형태



- 체크 리스트 상자의 프로퍼티
  - 리스트 상자와 대부분 동일
  - ThreeDCheckBoxes, CheckOnClick 프로퍼티가 추가됨.

체크리스트 상자 [2/4]

- ThreeDCheckBoxes
  - 체크리스트 상자에 포함된 체크 상자의 모양을 결정,
  - 프로퍼티의 설정 형태(참, 거짓)



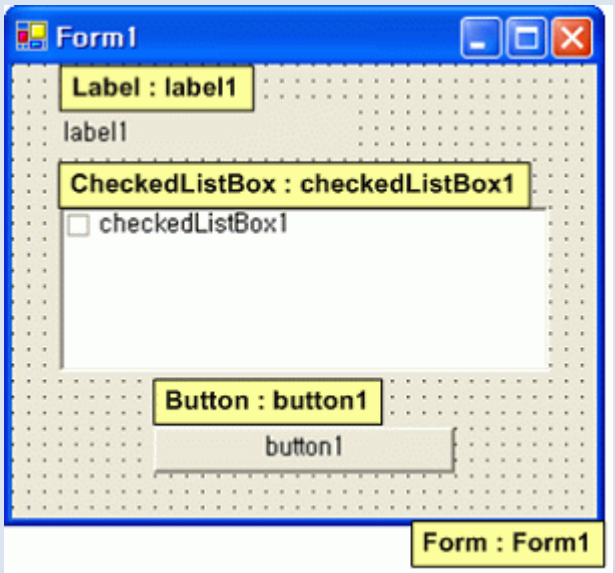
- CheckOnClick
  - 항목을 클릭했을때 체크 상자에 '✔ ' 표시가 나타나도록 설정.

체크리스트 상자 [3/4]

- 예제 8.9
  - 취미에 관련된 다양한 항목을 예시해 준 후, 클릭하여 선택하면 CheckedItems 프로퍼티를 이용하여 선택된 항목을 출력하는 예제.

[예제 8.9 – CheckedBoxApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값	
Form : Form1	Text	CheckedListBoxApp	
Label : label1	Items	취미를 선택하시오	
CheckedListBox : checkedlistBox1	Items	영화감상 음악감상 축구 농구 골프	스키 스노우보드 수영 장기 바둑
	Multi Column	True	
Button : button1	Text	선택	

체크리스트 상자 [4/4]

4) 이벤트 처리기

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

```
private void button1_Click(object sender, EventArgs e) {
    string strTemp = "";
    foreach(object obj in checkedListBox1.CheckedItems) {
        strTemp += obj.ToString();
        strTemp += " ": }
    MessageBox.Show("당신의 취미는" + strTemp + "입니다."); }
```

실행 방법 : 체크리스트 상자의 항목을 선택한 후, 선택 버튼을 클릭한다.
실행 결과 :