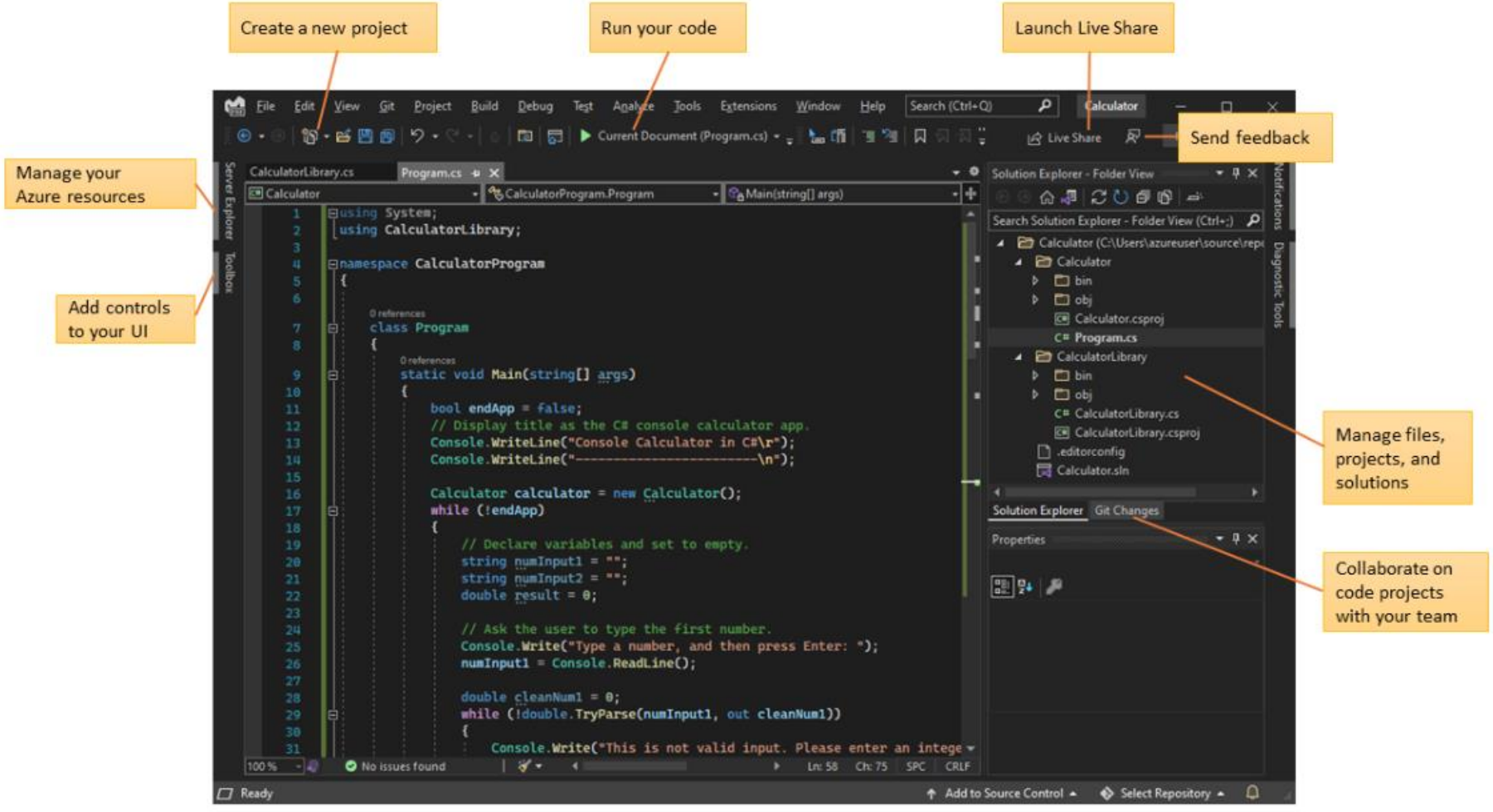
C# 윈도우 폼

광운대학교 소프트웨어학부
문석재

Visual Studio IDE 시작 | C#

- IDE(통합 개발 환경)는 소프트웨어 개발의 다양한 측면에서 사용할 수 있는 다양한 기능을 갖춘 프로그램.
- Visual Studio IDE는 코드를 편집, 디버그 및 빌드한 다음 앱을 게시하는 데 사용할 수 있는 창의적인 실행 패드.
- Visual Studio는 대부분의 IDE가 제공하는 표준 편집기 및 디버거로서 뿐만 아니라, 컴파일러와 코드 완성 도구, 그래픽 디자이너를 비롯해 소프트웨어 개발 프로세스를 향상시키는 많은 기능을 포함하고 있음.

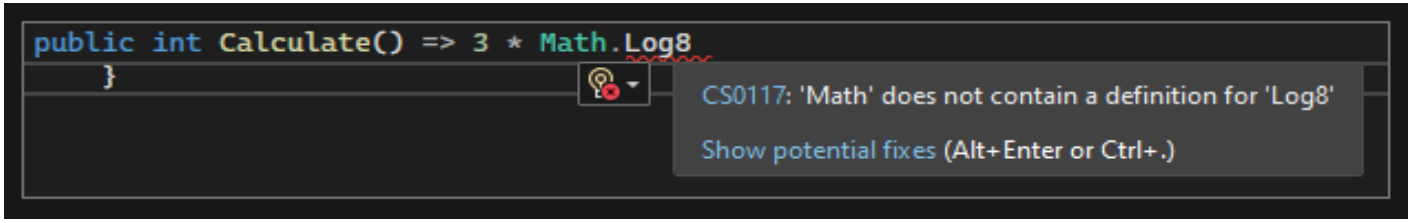
Visual Studio IDE 시작 | C#



주요 특징 및 기능의 위치를 나타내는 설명과 함께 Visual Studio 2022 IDE를 보여 주는 스크린샷.

Visual Studio IDE 시작 | C#

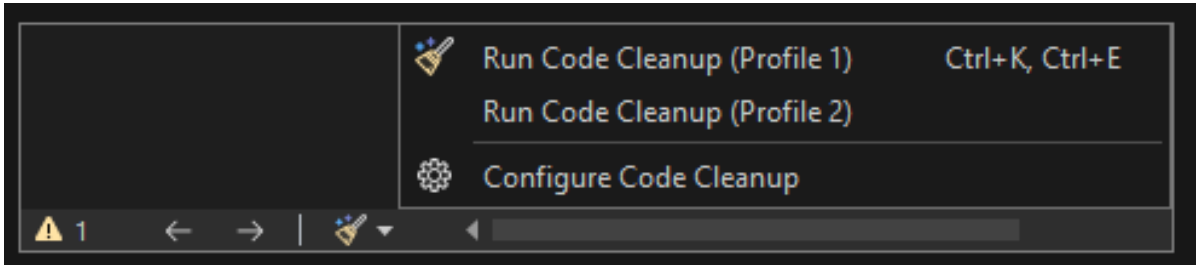
- 주요 생산성 향상 기능
  - 오류 표시선 및 빠른 작업
  - 오류 표시선은 물결 모양의 밑줄로, 입력할 때 코드의 오류 또는 잠재적인 문제를 알려준다. 이러한 시각적 단서는 빌드 또는 런타임 중에 오류 검색을 기다리지 않고 문제를 즉시 해결하는 데 도움이 된다.
  - 오류 표시선 위로 마우스를 가져가면 오류에 대한 추가 정보가 표시됩니다. 오류를 해결하기 위해 수행할 수 있는 Quick Actions 을 보여 주는 전구가 왼쪽 여백에 나타날 수도 있다.



Visual Studio IDE 시작 | C#

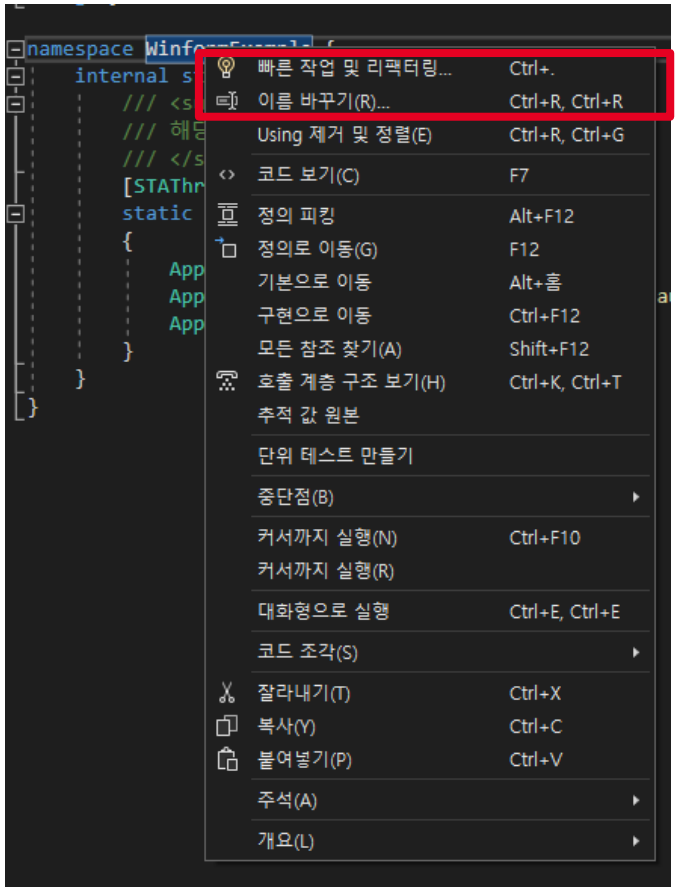
➢ 주요 생산성 향상 기능

- 코드 정리
- 버튼을 클릭하여 코드 형식을 지정하고 코드 스타일 설정, .editorconfig 규칙 및 Roslyn 분석기에서 제안하는 코드 수정 사항을 적용할 수 있다. 현재 C# 코드에만 사용할 수 있는 코드 정리는 코드 검토로 이동하기 전에 코드의 문제를 해결하는 데 도움이 된다.



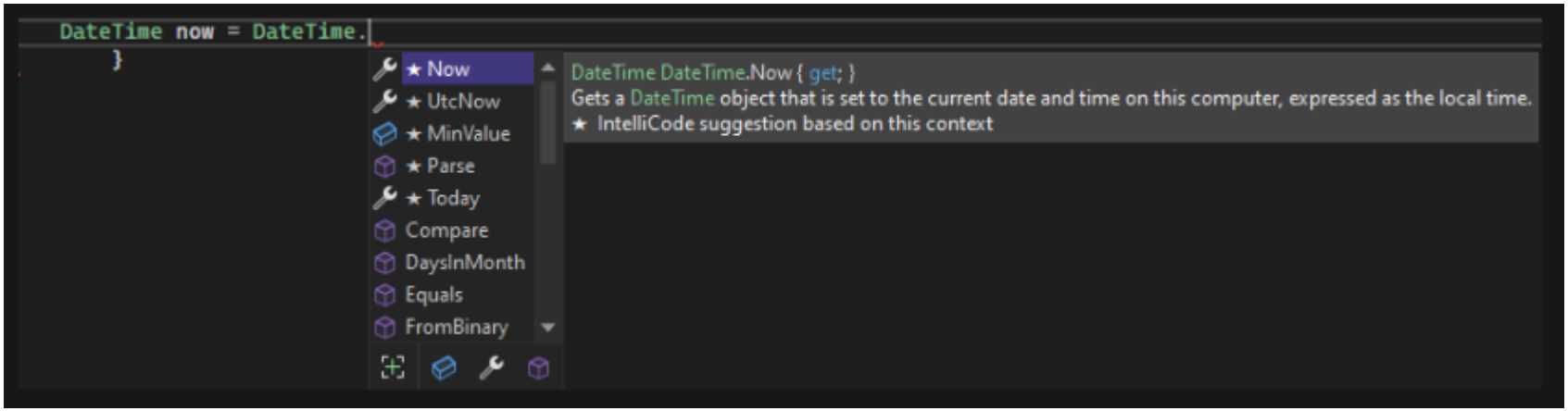
Visual Studio IDE 시작 | C#

- 주요 생산성 향상 기능
  - 리팩터링
  - 리팩터링에는 변수의 지능형 이름 바꾸기, 새 메서드로 코드 줄을 하나 이상 추출, 메서드 매개 변수의 순서 변경과 같은 작업이 포함된다.



Visual Studio IDE 시작 | C#

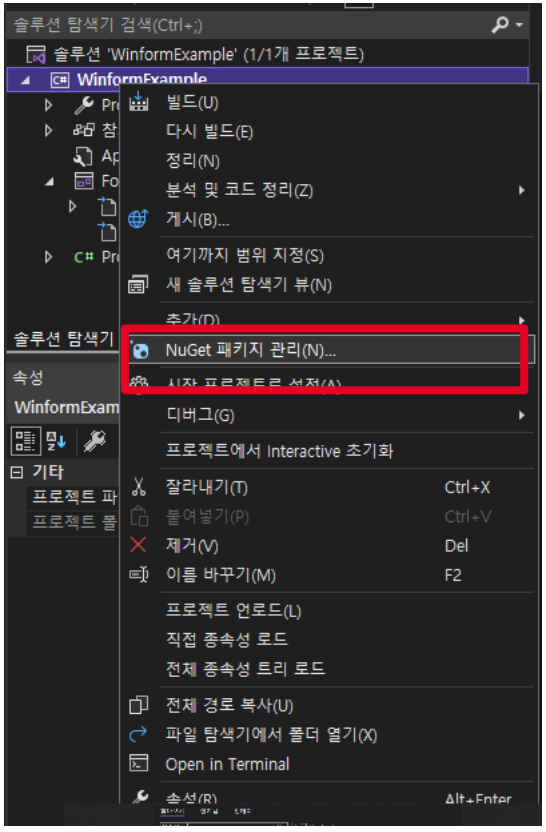
- 주요 생산성 향상 기능
  - IntelliSense
  - IntelliSense는 편집기에서 직접 코드에 대한 정보를 표시하고 경우에 따라 약간의 코드를 자동으로 작성하는 기능 집합이다. 다른 곳에서 형식 정보를 조회할 필요가 없도록 기본 설명서를 편집기에 인라인으로 포함하는 것과 같다.
  - 다음 그림에서는 IntelliSense에서 형식에 대한 멤버 목록을 표시하는 방법을 보여 준다.

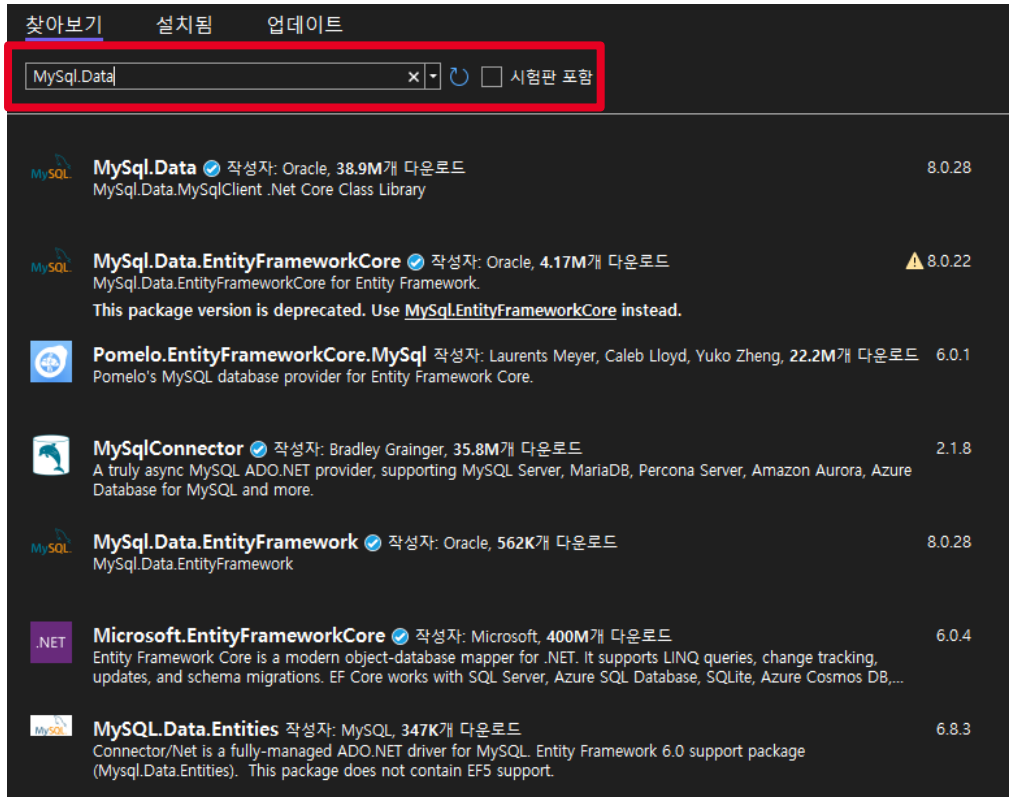


Visual Studio IDE 시작 | C#

➢ 주요 생산성 향상 기능

- Visual Studio 검색
- Visual Studio 검색(또는 Ctrl+Q)은 IDE 기능과 코드를 한 곳에서 신속하게 찾을 수 있는 좋은 방법이다.

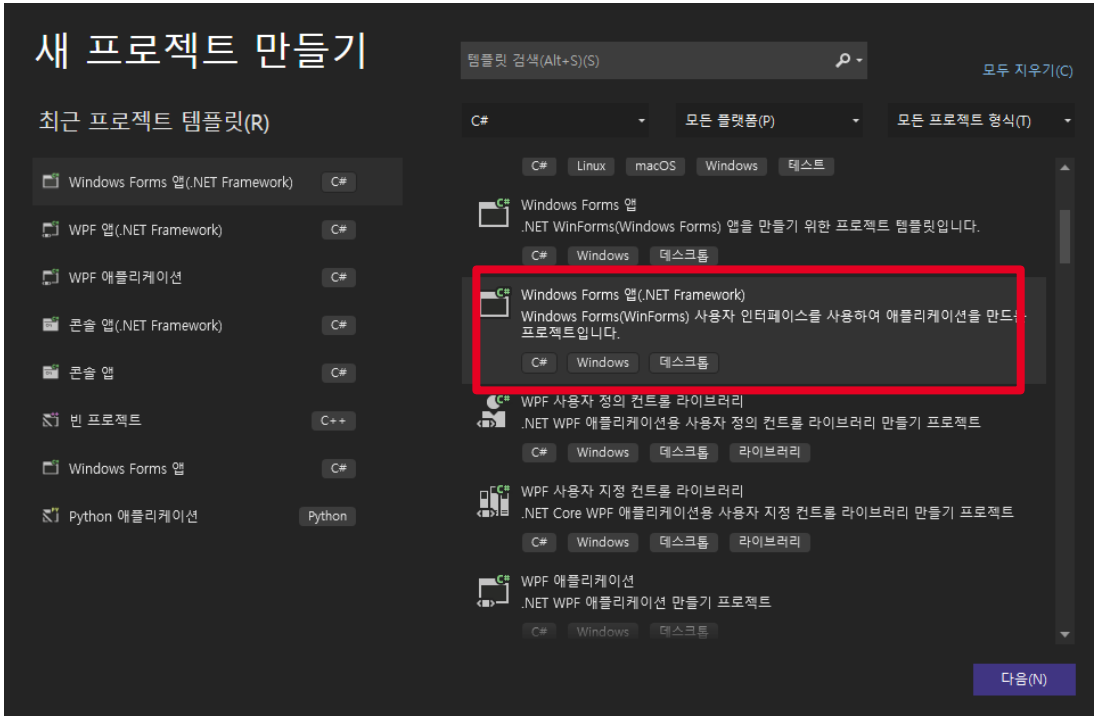




Visual Studio IDE 시작 | C#

➢ 프로그램 만들기

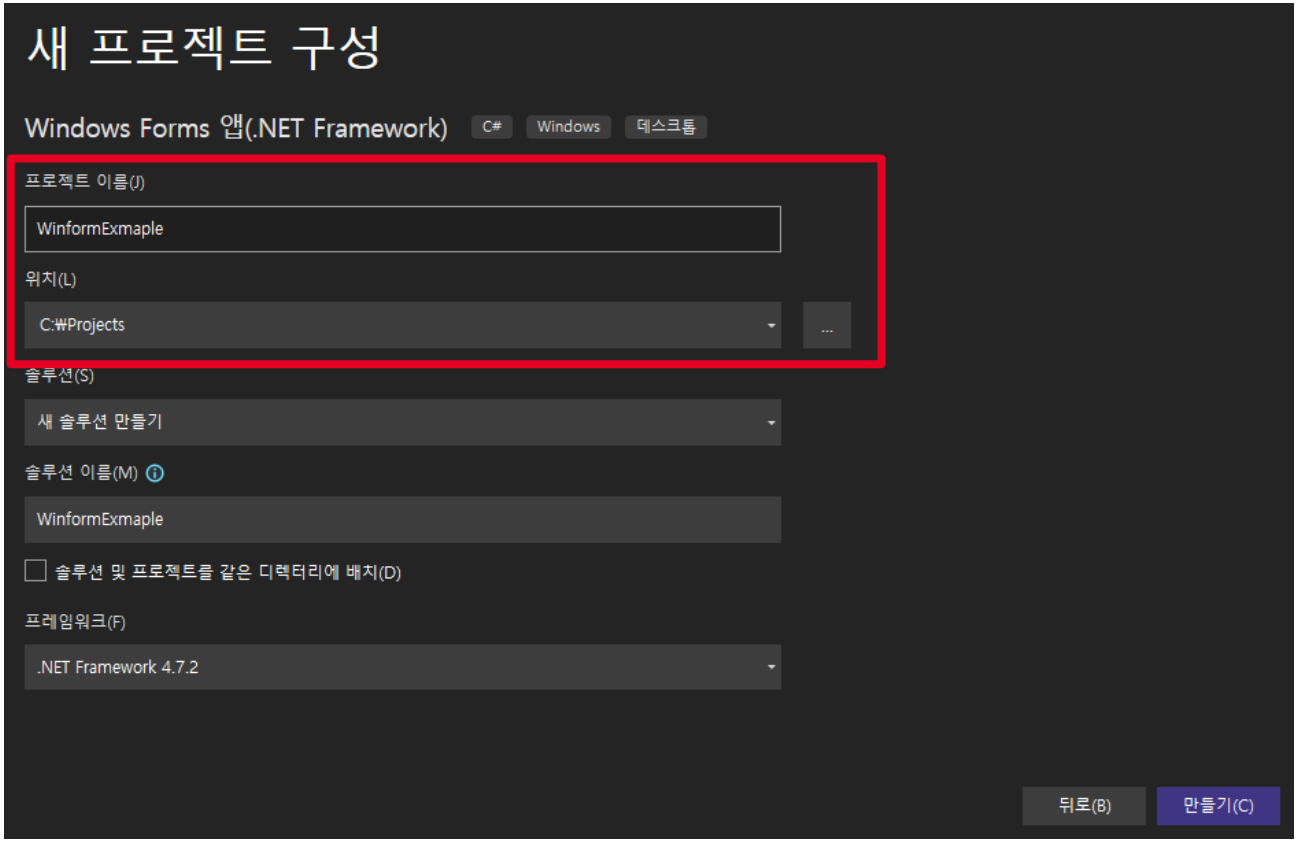
- 새 프로젝트 만들기를 선택한다. 새 프로젝트 만들기 창이 열리고 여러 프로젝트 템플릿이 표시된다.
- 템플릿에는 지정된 프로젝트 형식에 필요한 기본 파일 설정이 포함된다.



Visual Studio IDE 시작 | C#

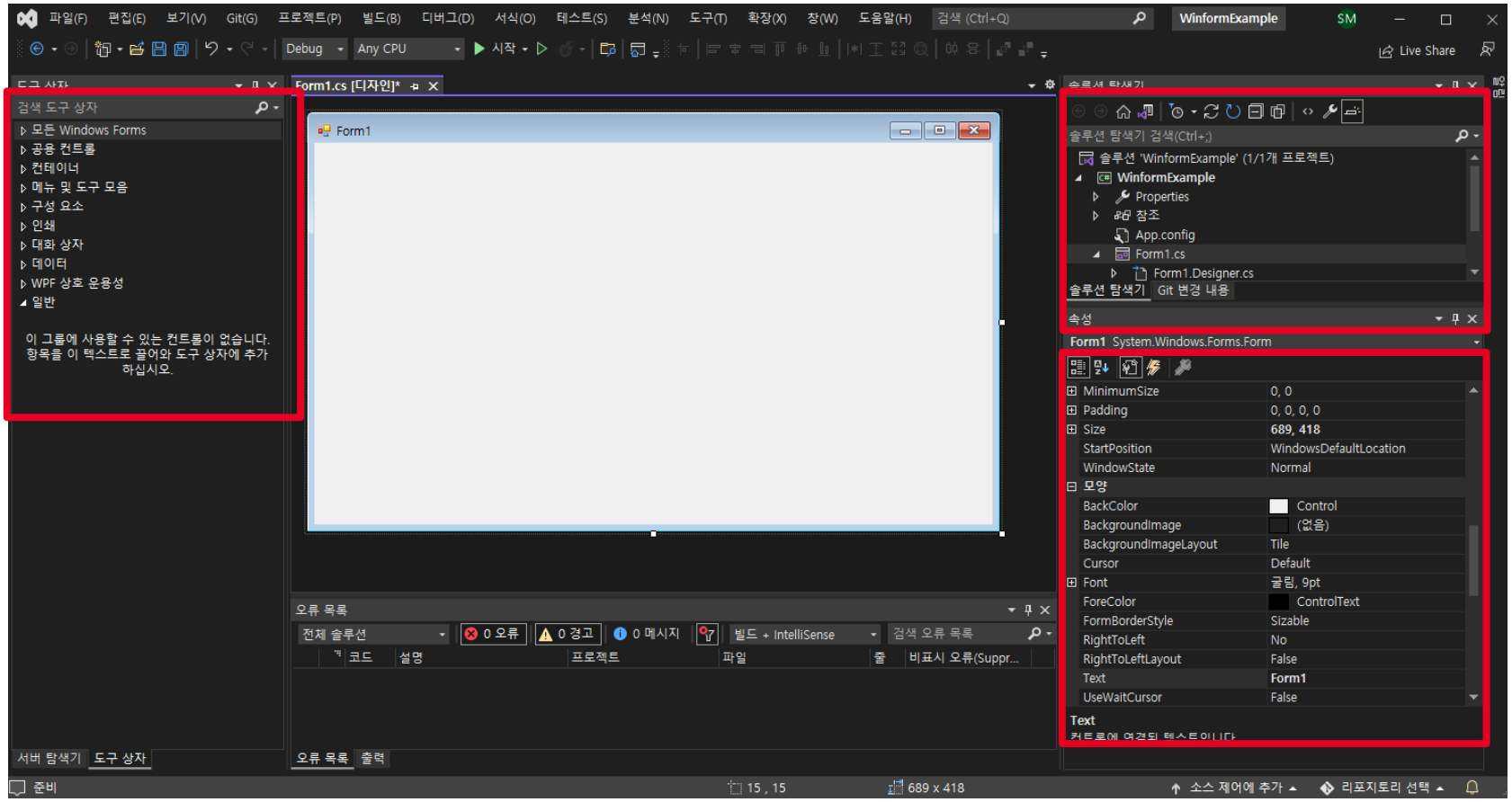
➢ 프로그램 만들기

- 새 프로젝트 구성 창에서 프로젝트 이름 상자에 WinformExample 를 입력한다. 필요에 따라 C:\Users\<name>\source\repos 의 기본 위치에서 프로젝트 디렉터리 위치를 변경한 후 다음을 클릭한다.



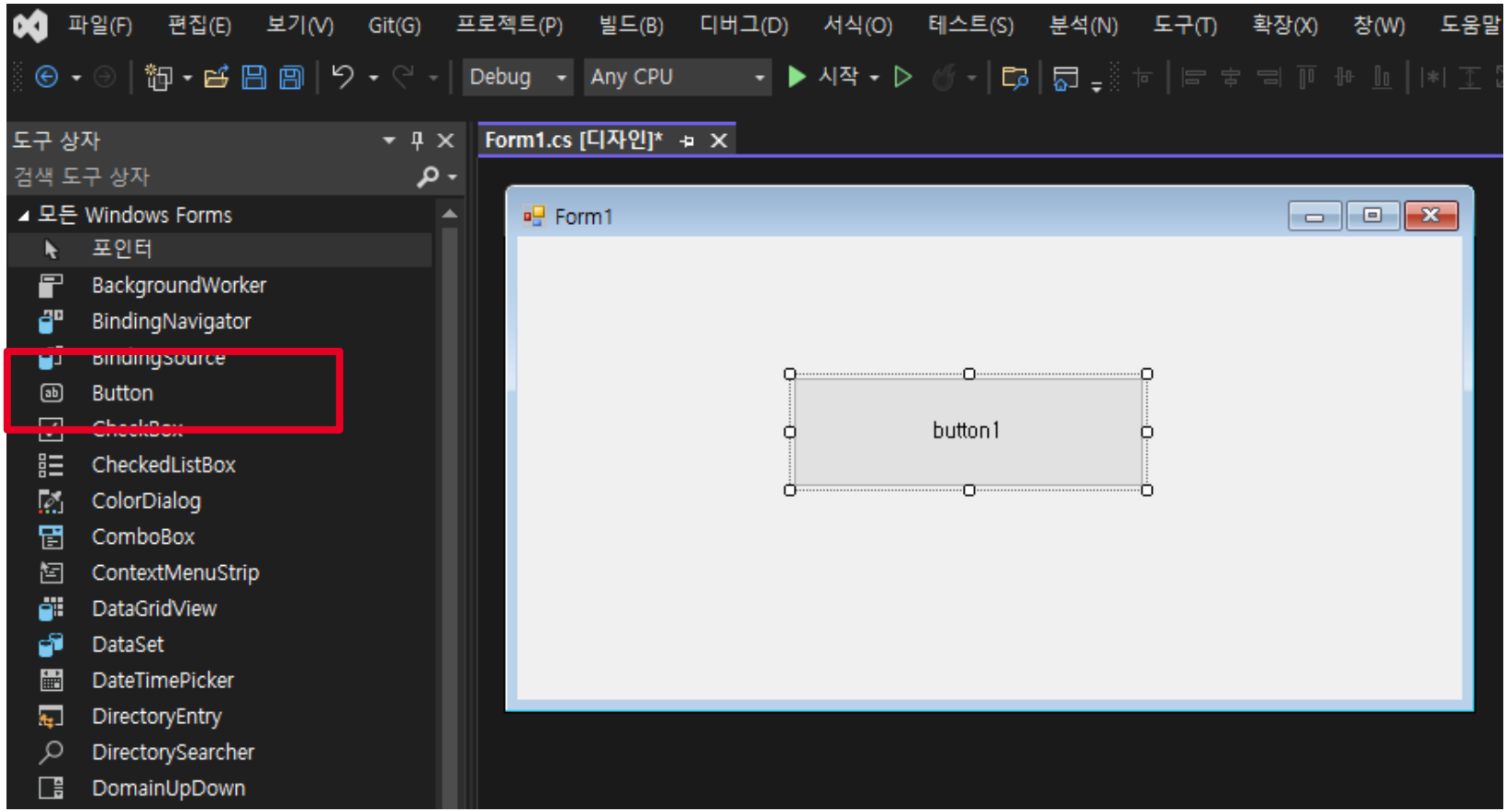
Visual Studio IDE 시작 | C#

- 양식에 버튼 추가– 도구 상자 를 선택하여 도구 상자 플라이아웃 창을 연다.
- (도구 상자 플라이아웃 옵션이 표시되지 않으면 메뉴 모음에서 열 수 있다. 이렇게 하려면 보기 〉 도구 상자 를 선택한다. 또는 Ctrl+Alt+X 키를 누른다.)



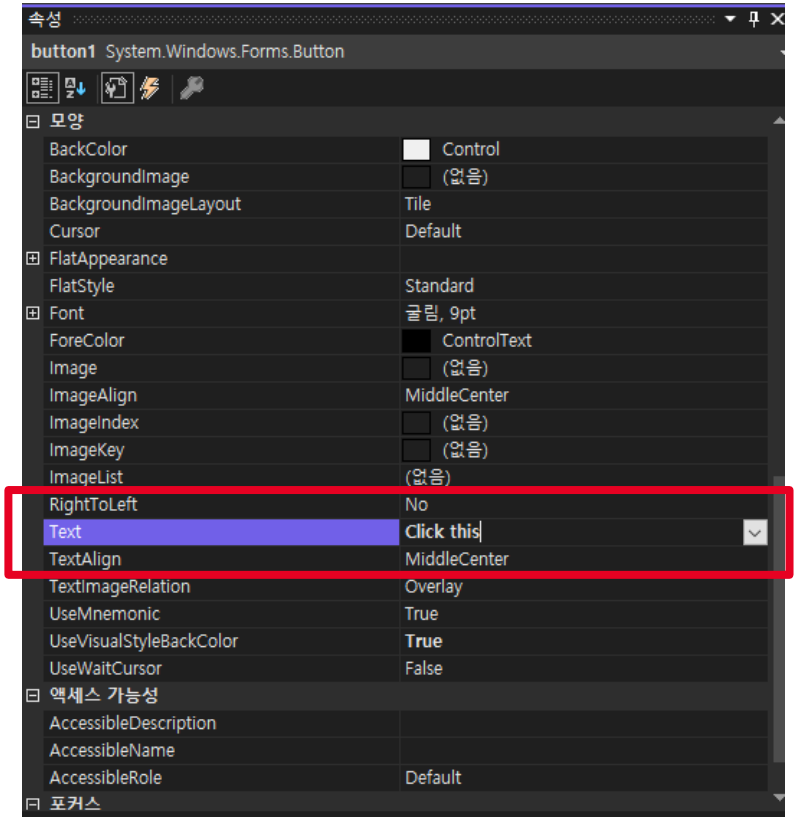
Visual Studio IDE 시작 | C#

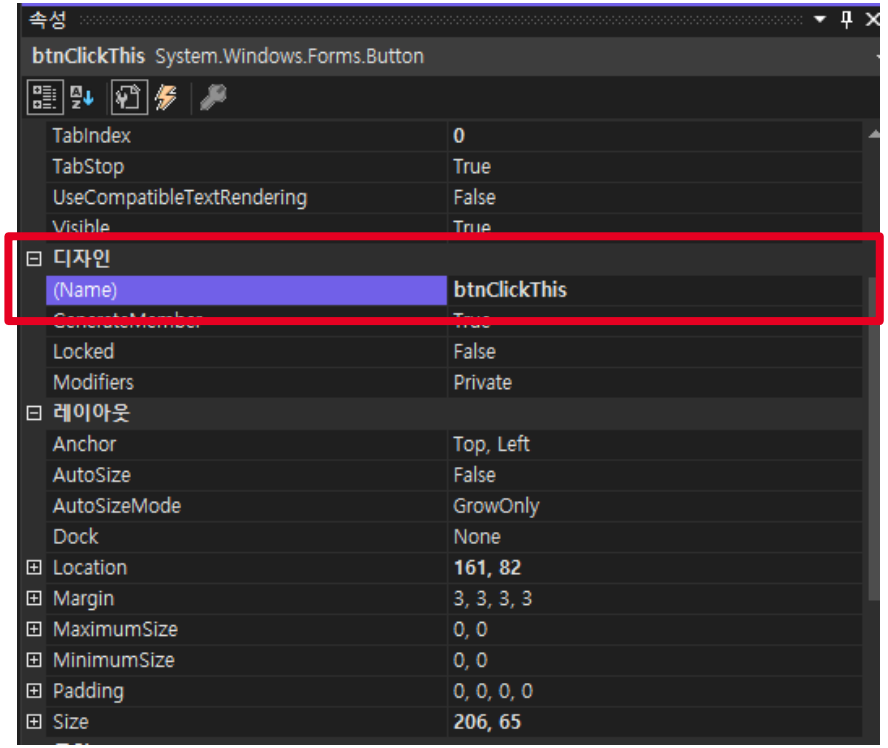
➢ Button 컨트롤을 선택하고 양식으로 끌어온다.



Visual Studio IDE 시작 | C#

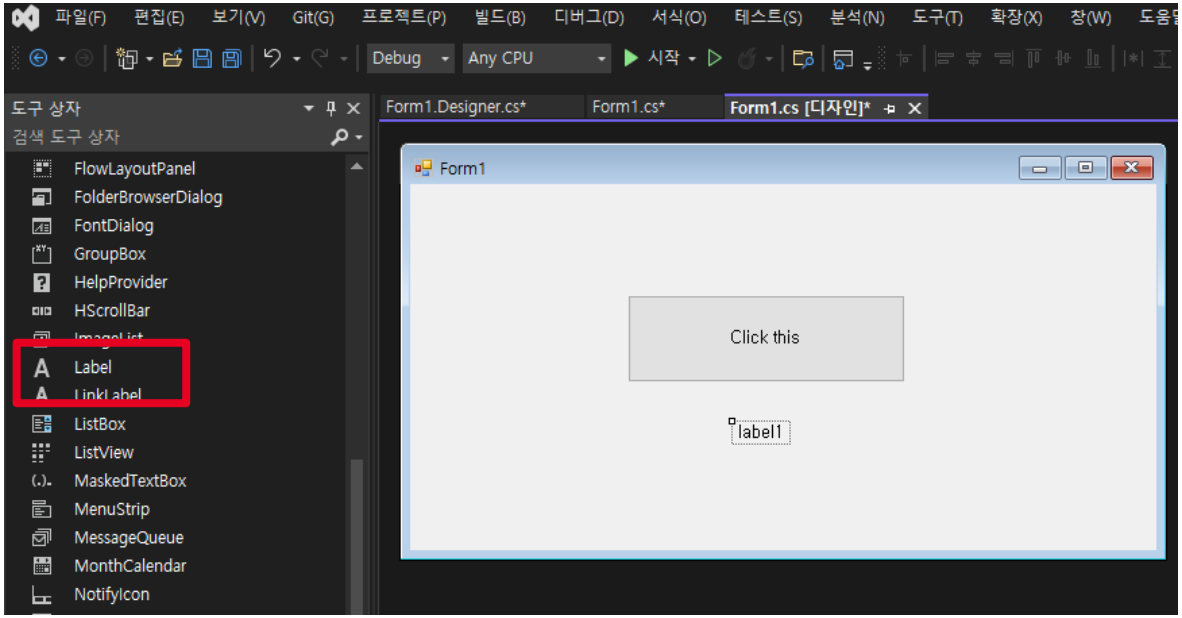
- 속성창에서 Text를 찾고 이름을 button1에서 “Click this”로 변경한 다음, Enter 키를 누른다.
- 속성 창의 디자인 섹션에서 이름을 button1에서 “btnClickThis”로 변경한 다음, Enter 키를 누른다.

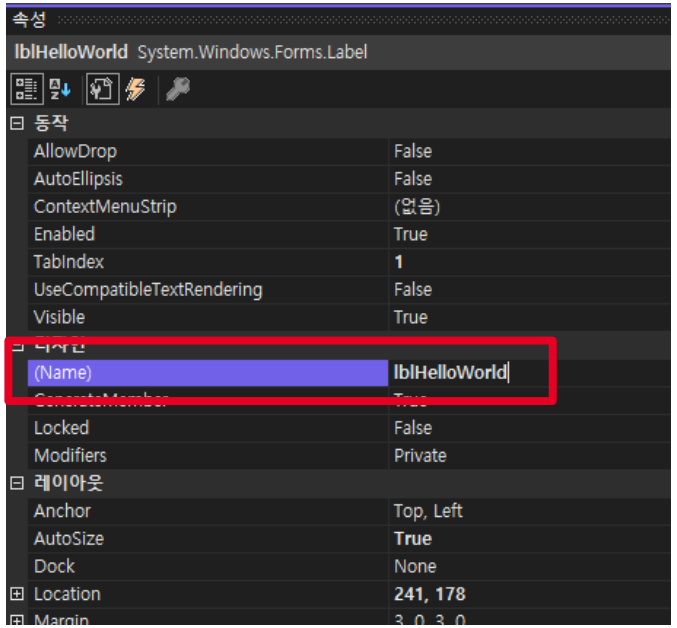




Visual Studio IDE 시작 | C#

- 양식에 레이블 추가
  - 동작을 만드는 button 컨트롤을 추가했다. 텍스트를 보낼 label 컨트롤을 추가해 보겠다. 도구 상자 창에서 label 컨트롤을 선택한 다음, 양식으로 끌어오고, 항목 클릭 button 아래에 놓는다. 속성 창의 디자인 섹션 또는 (DataBindings) 에서 이름을 label1 에서 lblHelloWorld로 변경한 다음, Enter 키를 누른다.





Visual Studio IDE 시작 | C#

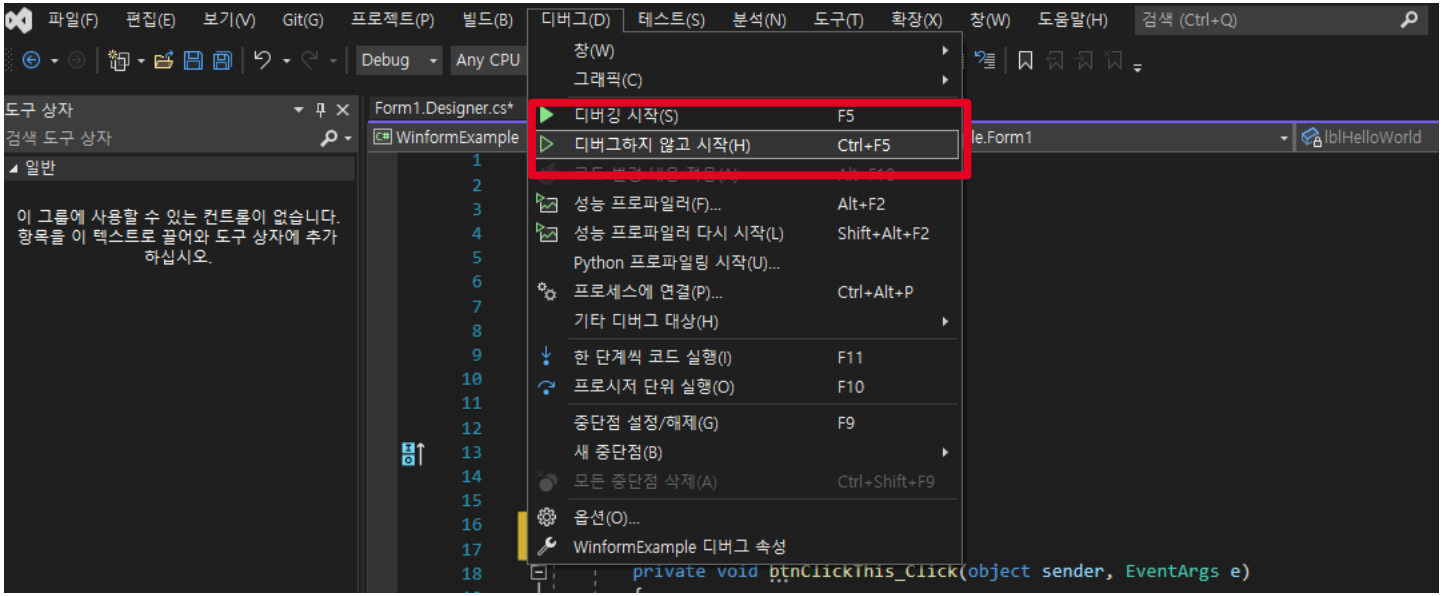
➢ 양식에 코드 추가

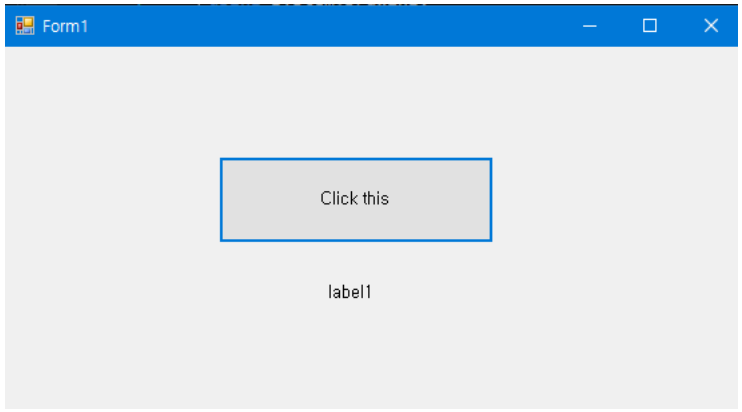
- Form1.cs [Design] 창에서 항목 클릭 버튼을 두 번 클릭하여 Form1.cs 창을 연다.
- (또는 솔루션 탐색기 에서 Form1.cs 를 확장한 다음, Form1 을 선택할 수 있다.)
- Form1.cs 창의 private void 줄 다음에 다음 스크린샷과 같이 lblHelloWorld.Text = "Hello World!";를 입력한다.

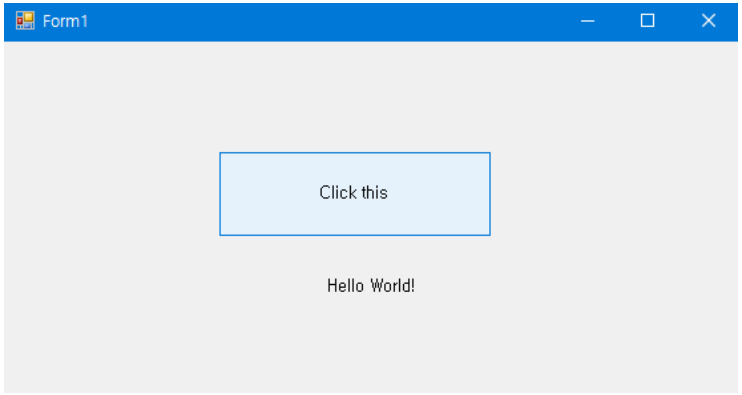
```
namespace WinformExample {

    public partial class Form1 : Form {
        public Form1()
        {
            InitializeComponent();
        }
        private void btnClickThis_Click(object sender, EventArgs e)
        {
            lblHelloWorld.Text = "Hello World!";
        }
    }
}
```

Visual Studio IDE 시작 | C#

➢ 디버그 –〉 디버그 하지 않고 시작

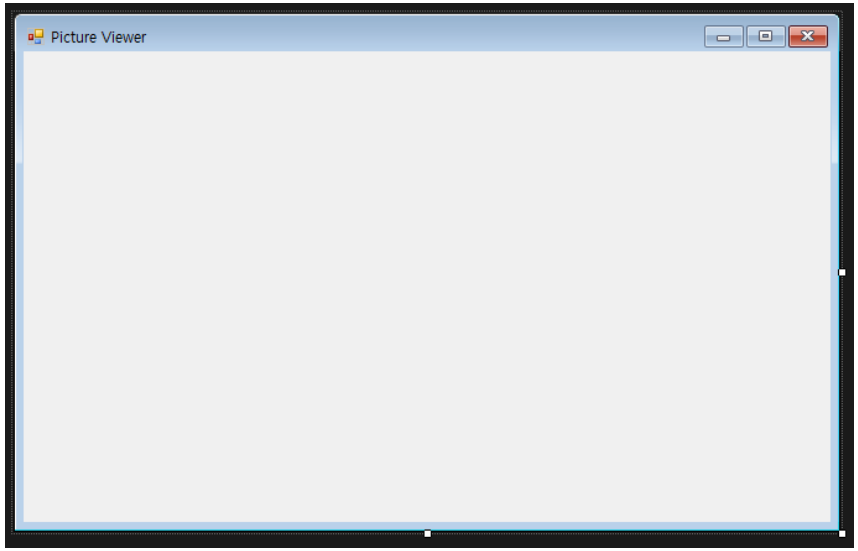


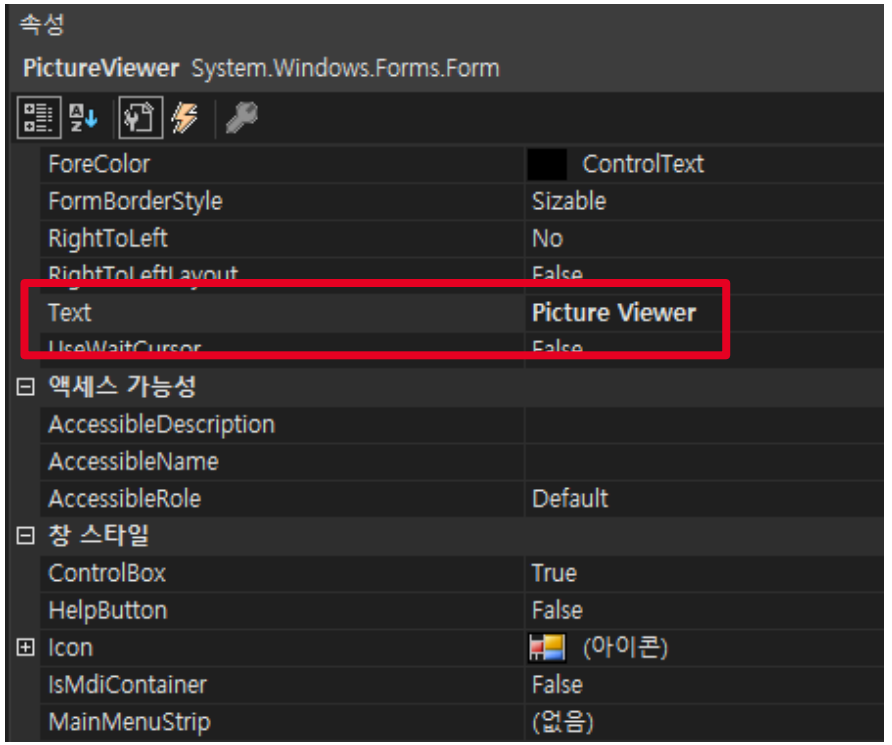




Visual Studio의
사진 뷰어 앱 만들기

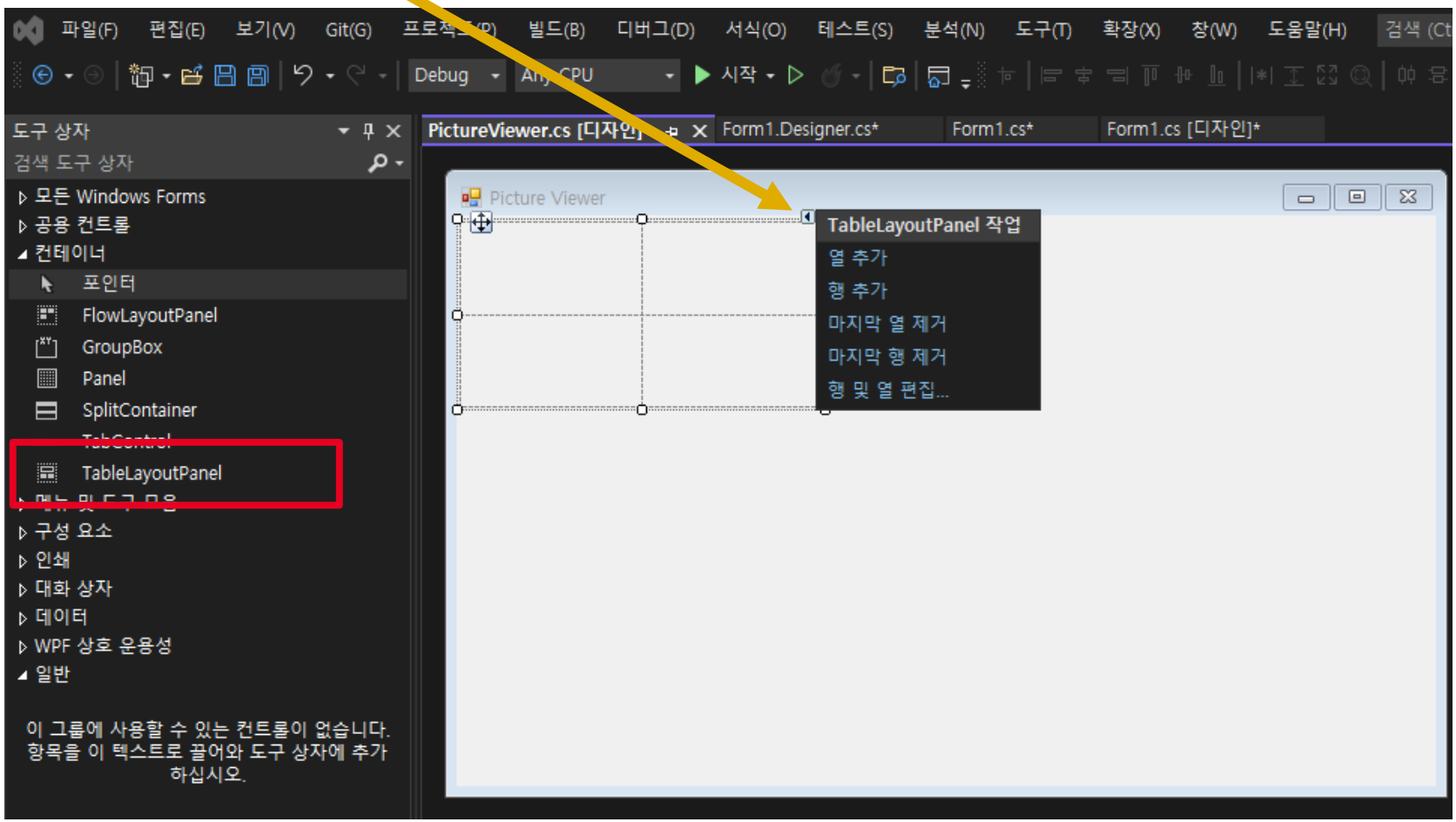
Visual Studio의 사진 뷰어 앱 만들기





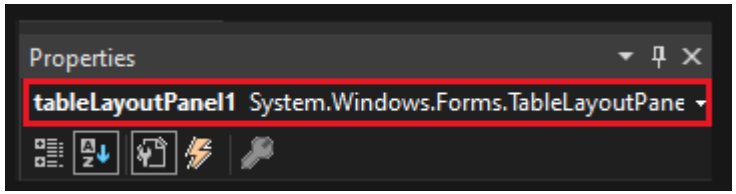
Visual Studio의 사진 뷰어 앱 만들기

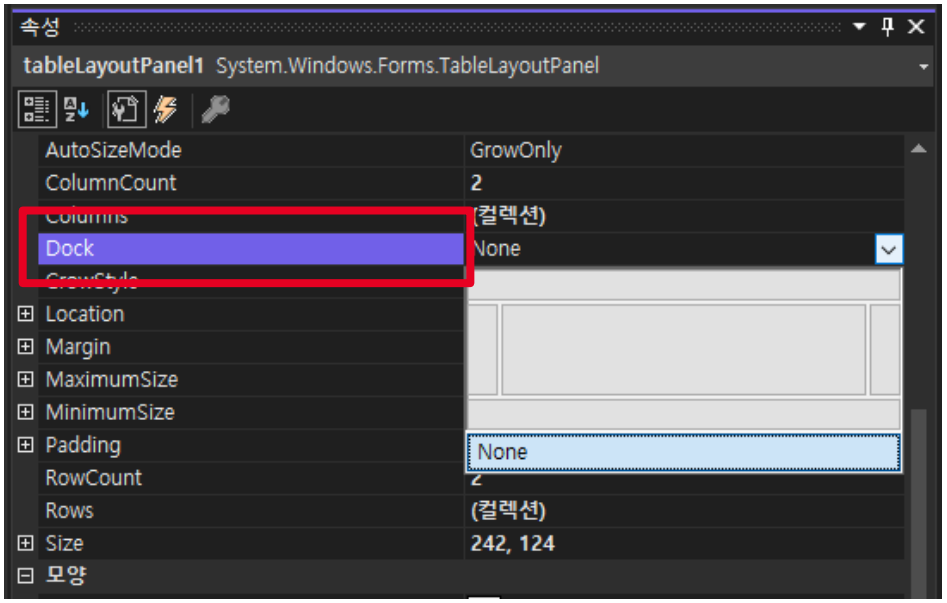
➢ 옆에 있는 작은 삼각형 기호를 선택하여 그룹을 엽니다.



Visual Studio의 사진 뷰어 앱 만들기

➢ TableLayoutPanel 을 선택한다. 선택한 컨트롤을 속성 창에서 확인할 수 있다.



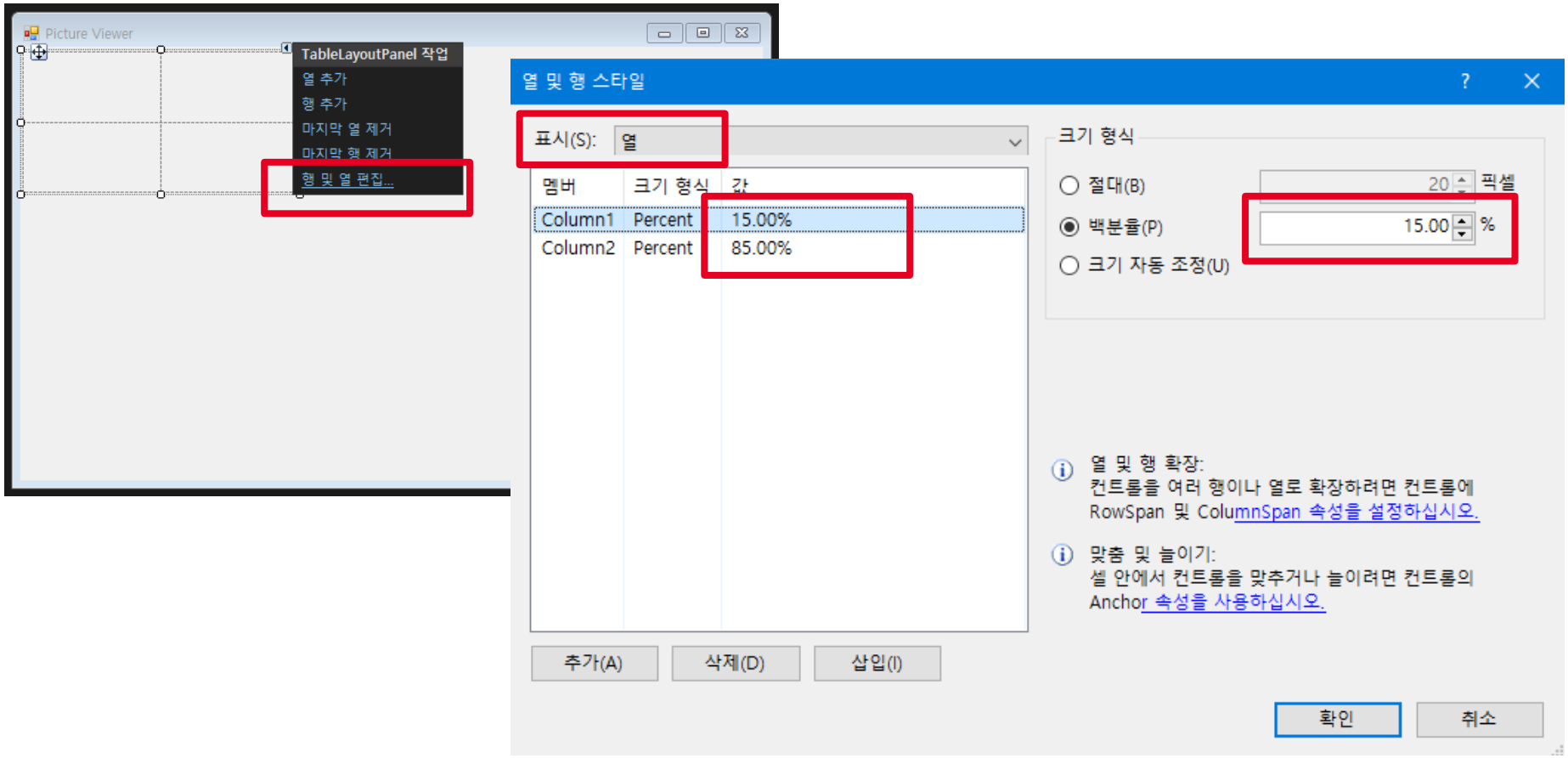


- 도킹은 창이 다른 창이나 영역에 연결되는 방식을 나타낸다.

- 폼의 크기를 다시 조정하면 TableLayoutPanel은 도킹된 상태를 유지한 채 폼의 크기에 맞게 자동으로 크기가 조정된다.

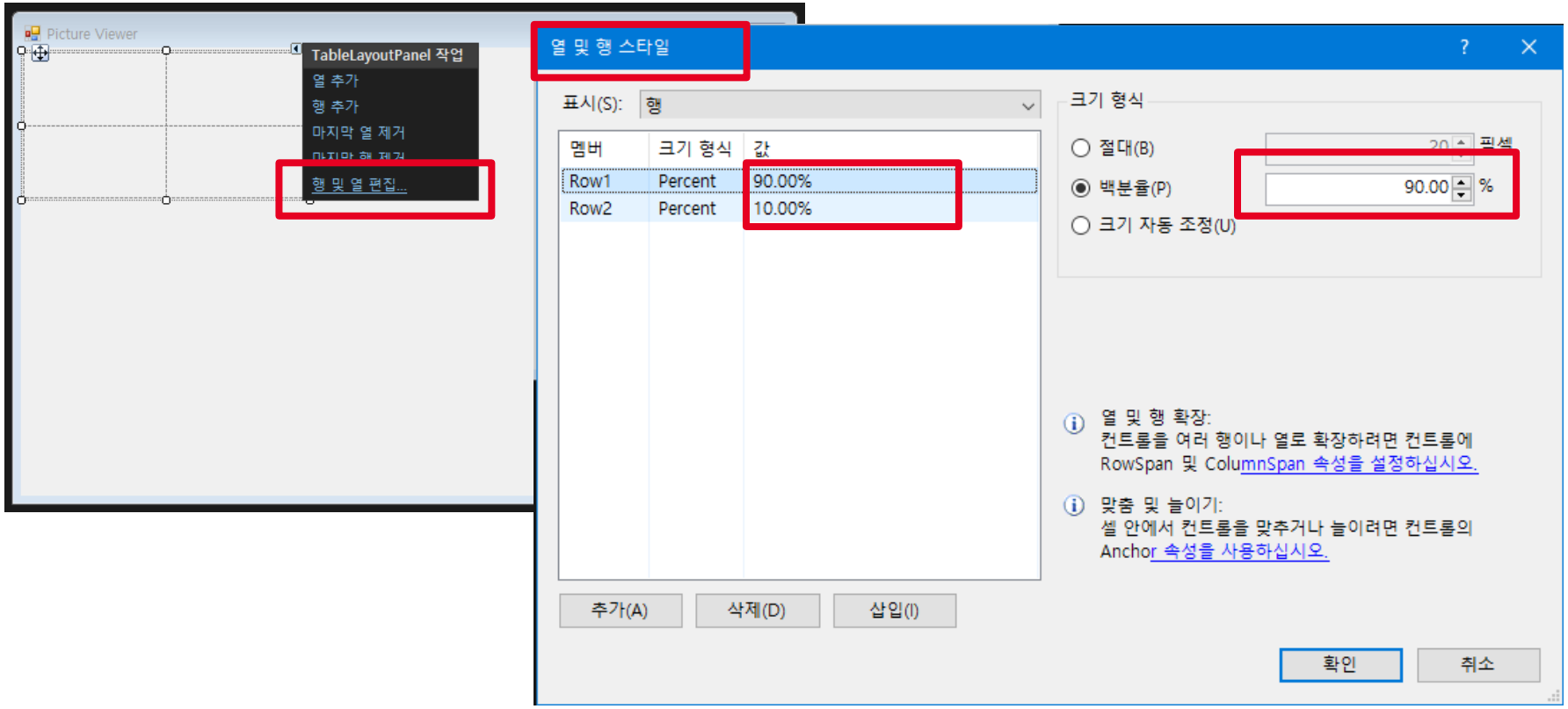
Visual Studio의 사진 뷰어 앱 만들기

- Column1 을 선택하고 크기를 15%로 설정합니다. 백분율 단추가 선택되어 있는지 확인한다.



Visual Studio의 사진 뷰어 앱 만들기

- Row1 을 90%로 설정하고 Row2 를 10%로 설정한다. 확인을 선택하여 변경 내용을 저장한다.
- Column2 를 선택하고 크기를 85%로 설정한다.



Visual Studio의 사진 뷰어 UI 컨트롤 추가

- 다음 작업을 수행하는 방법을 알아본다.
  - 애플리케이션에 컨트롤 추가
  - 레이아웃 패널에 단추 추가
  - 컨트롤 이름 및 위치 변경
  - 대화 상자 구성 요소 추가

Visual Studio의 사진 뷰어 UI 컨트롤 추가

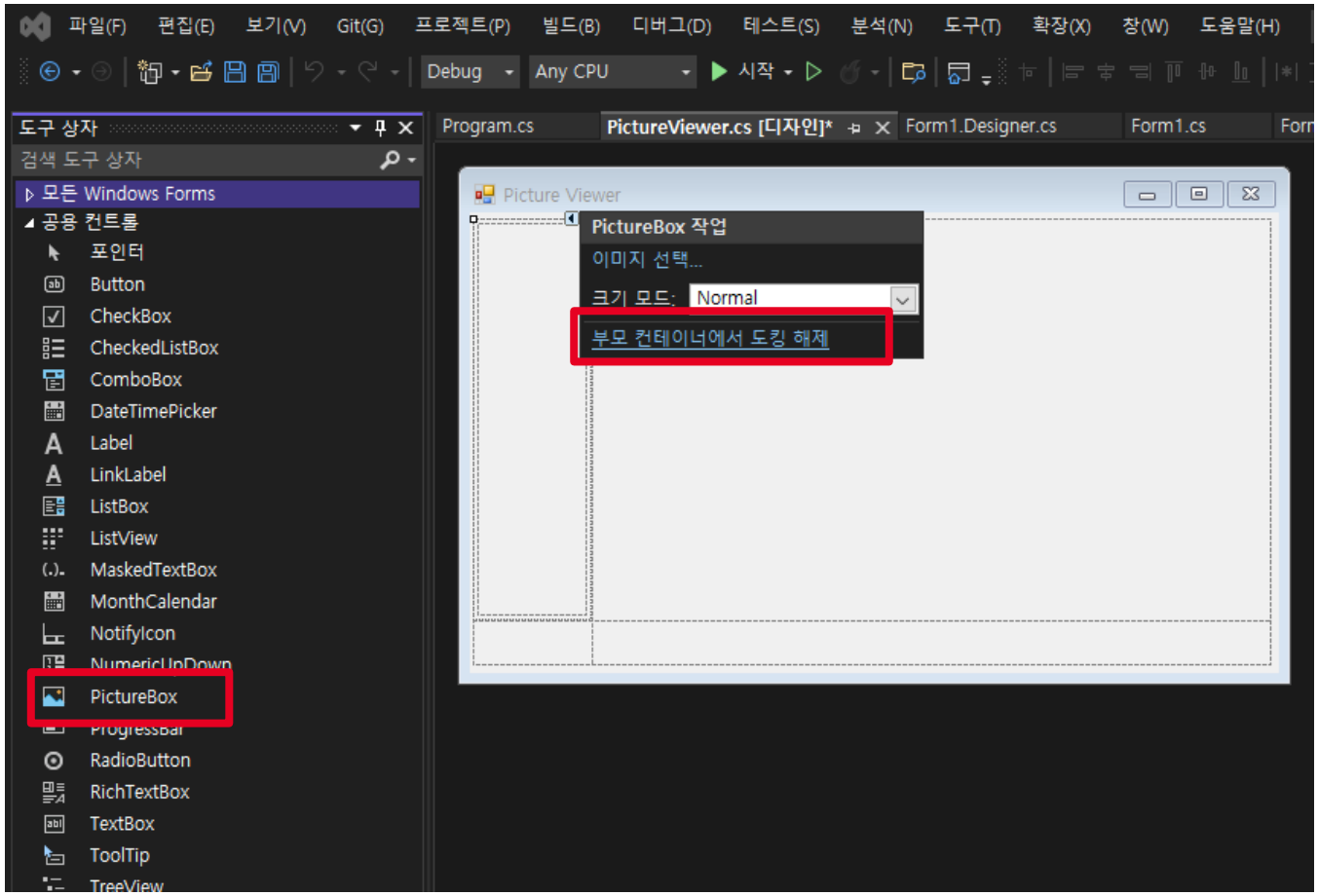
➢ 애플리케이션에 컨트롤 추가

- PictureBox를 두 번 클릭하여 PictureBox 컨트롤을 폼에 추가한다. Visual Studio IDE는 PictureBox 컨트롤을 TableLayoutPanel의 첫 번째 빈 셀에 추가한다.
- 새 PictureBox 컨트롤을 선택한 다음, 새 PictureBox 컨트롤의 검은색 삼각형을 선택하여 작업 목록을 표시한다.
- PictureBox Dock 속성을 채우기로 설정하는 부모 컨테이너에서 도킹을 선택한다. 속성 창에서 해당 값을 볼 수 있습니다.
- PictureBox의 속성 창에서 ColumnSpan 속성을 2 로 설정한다.

Visual Studio의 사진 뷰어 UI 컨트롤 추가

➢ 애플리케이션에 컨트롤 추가

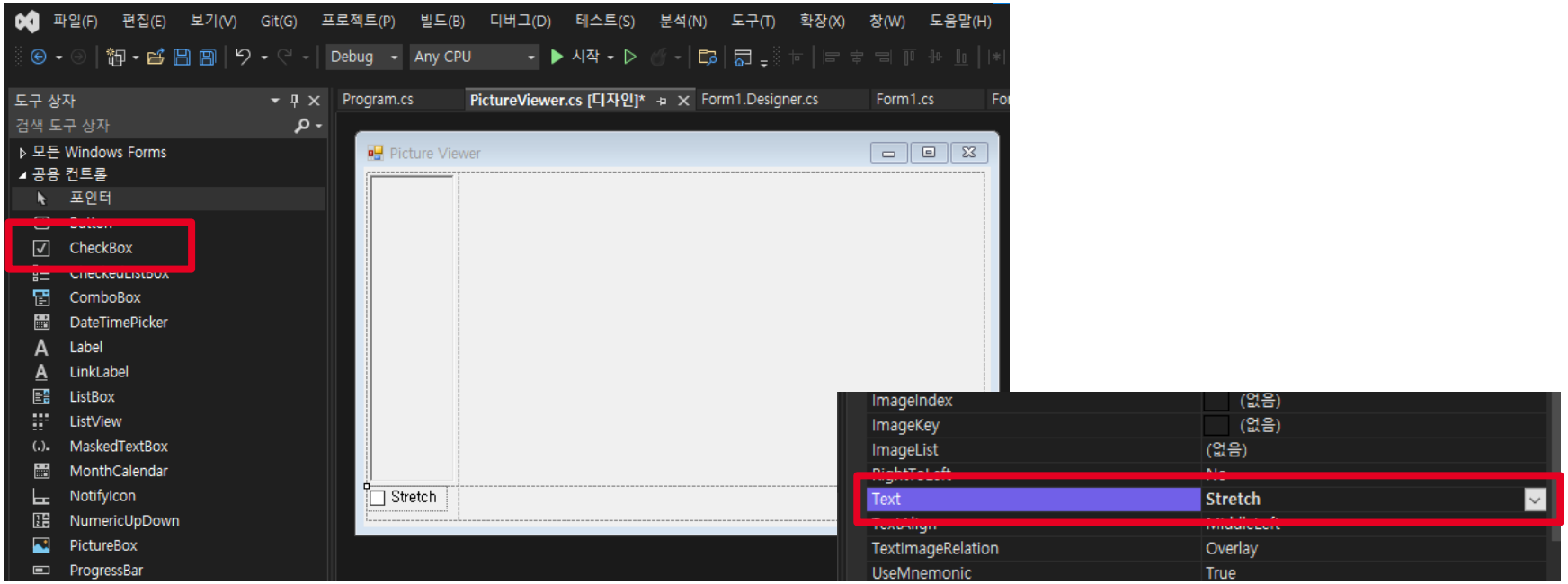
- BorderStyle 속성을 Fixed3D 로 설정한다.



Visual Studio의 사진 뷰어 UI 컨트롤 추가

➢ 애플리케이션에 컨트롤 추가

- Windows Forms 디자이너 에서 TableLayoutPanel 을 선택한다. 도구 상자 에서 CheckBox 항목을 두 번 클릭하여 새 CheckBox 컨트롤을 추가한다. PictureBox가 TableLayoutPanel의 처음 두 셀을 차지하므로 CheckBox 컨트롤은 왼쪽 아래 셀에 추가된다.



Visual Studio의 사진 뷰어 UI 컨트롤 추가

➢ 애플리케이션에 컨트롤 추가

- 다음 단계에서는 TableLayoutPanel의 새 레이아웃 패널에 4개의 단추를 추가하는 방법을 보여준다.
- 폼에서 TableLayoutPanel을 선택한다. 도구 상자 를 열고 컨테이너 를 선택한다. FlowLayoutPanel 을 두 번 클릭하여 TableLayoutPanel의 마지막 셀에 새 컨트롤을 추가한다.
- FlowLayoutPanel의 Dock 속성을 채우기로 설정한다. 검은색 삼각형을 선택한 다음, 부모 컨테이너에서 도킹 을 선택한다.
- FlowLayoutPanel: 한 행의 다른 컨트롤을 차례로 정렬하는 컨테이너
- 새 FlowLayoutPanel을 선택한 다음, 도구 상자 를 열고 공용 컨트롤 을 선택한다. 버튼 항목을 두 번 클릭하여 button1 이라는 버튼 컨트롤을 추가한다.

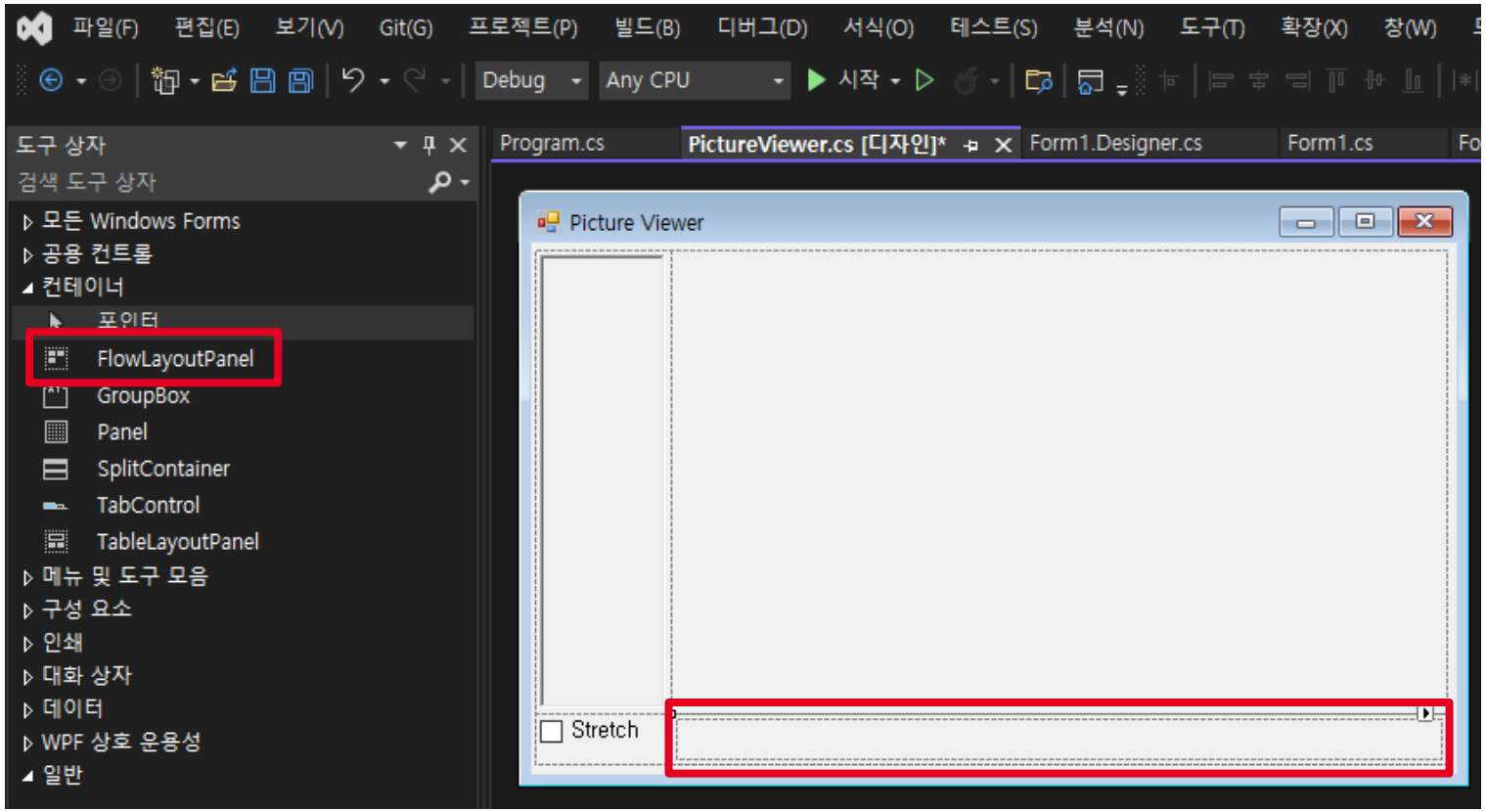
Visual Studio의 사진 뷰어 UI 컨트롤 추가

➢ 애플리케이션에 컨트롤 추가

- 버튼을 다시 두 번 클릭하여 다른 단추를 추가한다
- 이 방식으로 두 개의 버튼를 더 추가합니다. 또 다른 옵션은 button2 를 선택한 다음, 편집 〉 복사 를 선택하거나 Ctrl+C 를 누르는 것이다. 그런 다음, 메뉴 모음에서 편집 〉 붙여넣기 를 선택하거나 Ctrl+V 를 눌러 단추의 복사본을 붙여 넣는다. 다시 한 번 붙여 넣는다. IDE에서 FlowLayoutPanel에 button3 및 button4 를 추가한다.
- 첫 번째 버튼을 선택하고 Text 속성을 'Show a picture'로 설정한다.
- 그런 후 다음 세 버튼의 Text 속성을 'Clear the picture', 'Set the background color', 'Close'로 설정한다.
- 버튼의 크기를 조정하고 정렬하려면 FlowLayoutPanel을 선택한다. FlowDirection 속성을 RightToLeft 로 설정한다.
- 버튼은 셀의 오른쪽에 정렬되며 그림 표시 버튼가 오른쪽에 표시되도록 순서가 반전된다. FlowLayoutPanel 주위에 버튼을 끌어 원하는 순서로 정렬할 수 있다.
- 'Close' 버튼를 클릭하여 선택한다. 그런 다음 나머지 버튼를 동시에 선택하려면 Ctrl 키를 누른 채 선택한다.
- 속성 창에서 AutoSize 속성을 True 로 설정한다. 버튼의 크기가 텍스트에 맞게 조정된다.

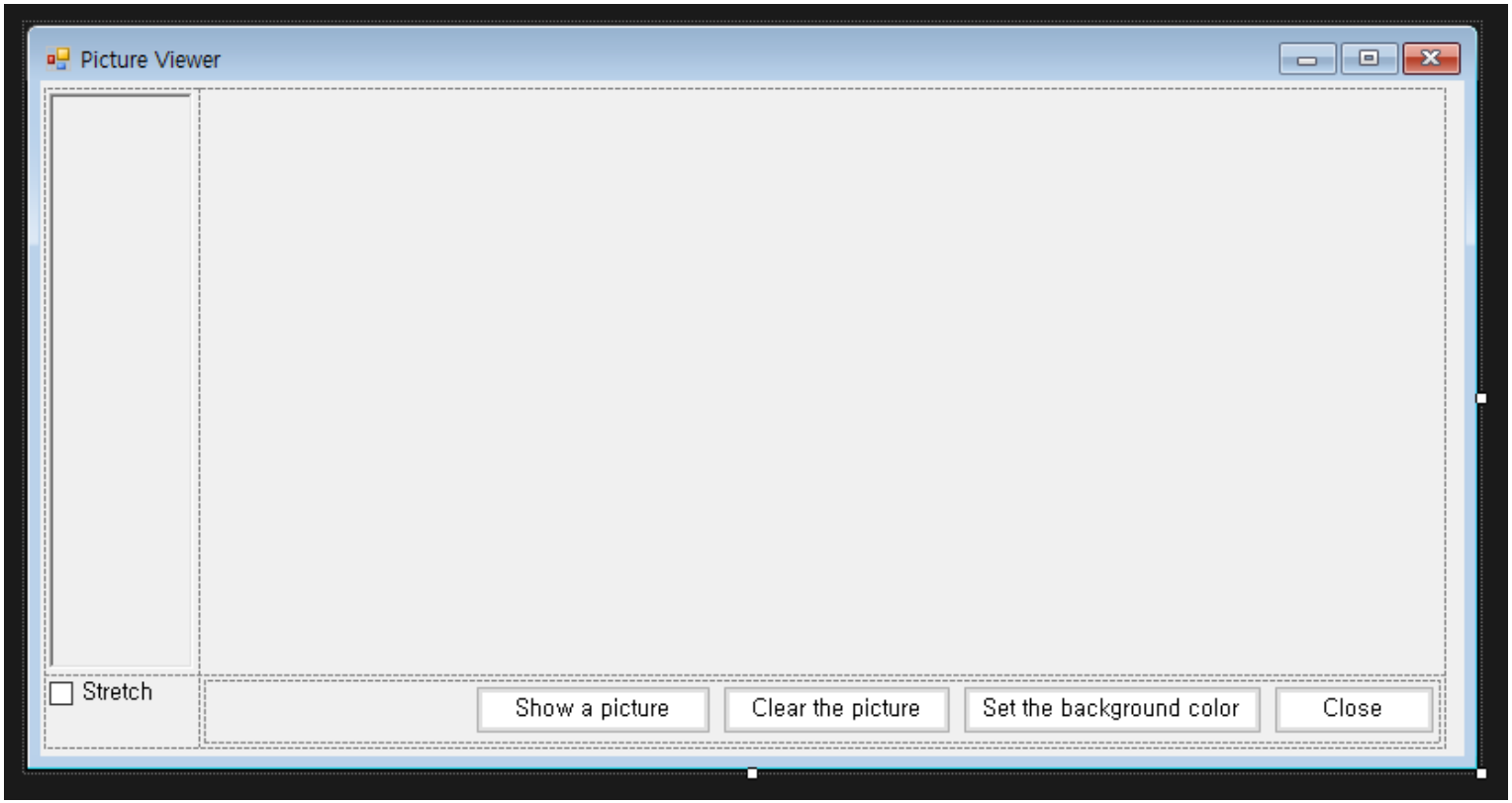
Visual Studio의 사진 뷰어 UI 컨트롤 추가

- 애플리케이션에 컨트롤 추가
  - FlowLayoutPanel 선택 후, 부모와 도킹 클릭



Visual Studio의 사진 뷰어 UI 컨트롤 추가

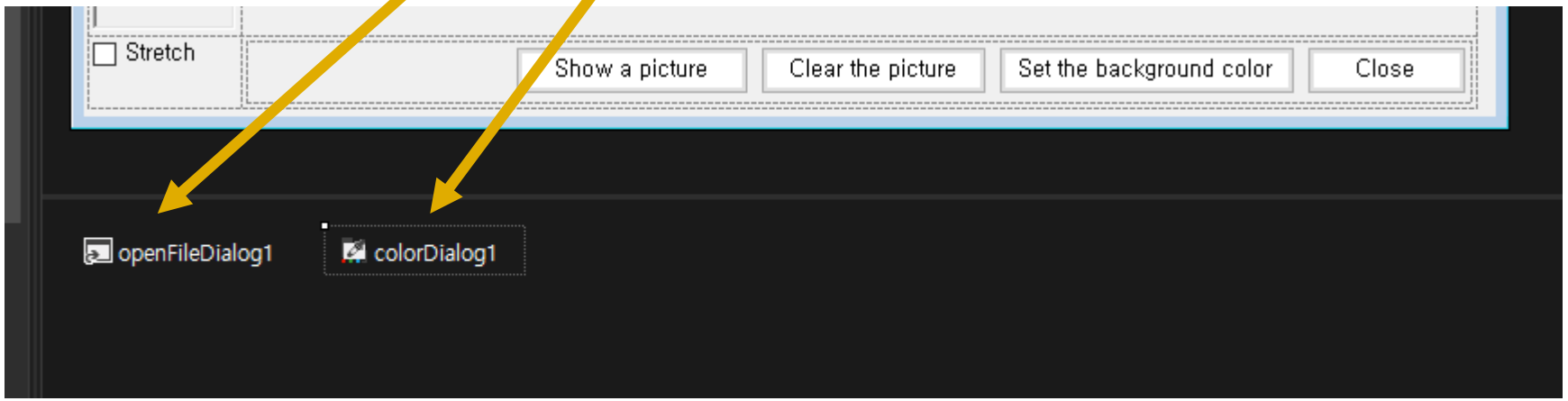
- 'Show a picture', Clear the picture', 'Set the background color', 'Close' 의 (Name) 속성을 각각 아래와 같이 변경
- showButton, clearButton, backgroundButton, closeButton



Visual Studio의 사진 뷰어 UI 컨트롤 추가

➢ 대화 상자 구성 요소 추가

- OpenFileDialog 구성 요소 및 ColorDialog 구성 요소를 폼에 추가한다.
- Windows Forms 디자이너(Form1.cs [Design])를 선택합니다. 그런 다음, 도구 상자 를 열고 대화 상자 그룹을 선택한다.
- OpenFileDialog 를 두 번 클릭하여 openFileDialog1 이라는 구성 요소를 폼에 추가한다.
- ColorDialog 를 두 번 클릭하여 colorDialog1 이라는 구성 요소를 추가한다. 구성 요소는 Windows Forms 디자이너의 아래쪽에 아이콘으로 표시된다.



Visual Studio의 사진 뷰어 UI 컨트롤 추가

➢ 대화 상자 구성 요소 추가

- openFileDialog1 아이콘을 선택하고 두 가지 속성을 설정한다.
- Filter 속성을 다음으로 설정한다.

콘솔 복사

```
JPEG Files (*.jpg)|*.jpg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp|All files (*.*)|*.*
```

- Title 속성을 'Select a picture' 로 설정한다.
- Filter 속성 설정은 Select a picture(사진 선택) 대화 상자에 표시되는 형식을 지정한다.
- JPEG Files (*.jpg)|*.jpg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp|All files (*.*)|*.*

컨트롤에 대한 이벤트 처리기 추가

- 대화 상자 구성 요소 추가
  - showButton을 더블클릭

```
namespace WinformExample {
    public partial class PictureViewer : Form {
        public PictureViewer()
        {
            InitializeComponent();
        }
        private void showButton_Click(object sender, EventArgs e)
        {

        }
    }
}
```

컨트롤에 대한 이벤트 처리기 추가

➢ 대화 상자 구성 요소 추가

- showButton, clearButton, backgroundButton, closeButton을 더블클릭

```
namespace WinformExample {
    public partial class PictureViewer : Form {
        public PictureViewer()
        {
            InitializeComponent();
        }
        private void showButton_Click(object sender, EventArgs e)
        {

        }
        private void clearButton_Click(object sender, EventArgs e)
        {

        }
        private void backgroundButton_Click(object sender, EventArgs e)
        {

        }
        private void closeButton_Click(object sender, EventArgs e)
        {

        }
    }
}
```

컨트롤에 대한 이벤트 처리기 추가

- 대화 상자를 여는 코드 작성
  - showButton, clearButton

```
private void showButton_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.Load(openFileDialog1.FileName);
    }
}
```

```
private void clearButton_Click(object sender, EventArgs e)
{
    pictureBox1.Image = null;
}
```

컨트롤에 대한 이벤트 처리기 추가

- 대화 상자를 여는 코드 작성
  - backgroundButton, closeButton을 더블클릭

```
private void backgroundButton_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
        pictureBox1.BackColor = colorDialog1.Color;
}
```

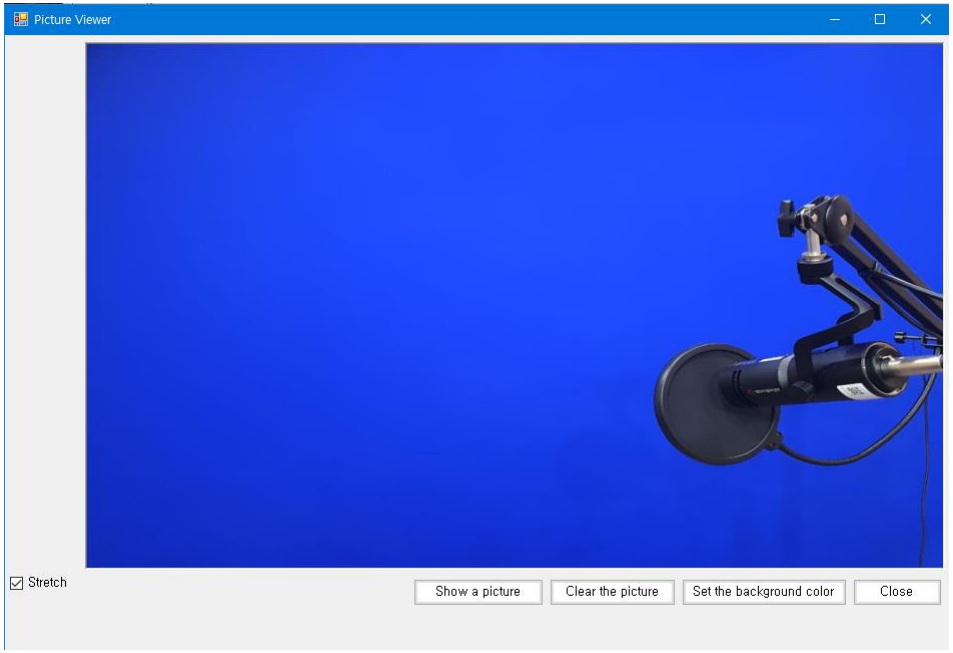
```
private void closeButton_Click(object sender, EventArgs e)
{
    this.Close();
}
```

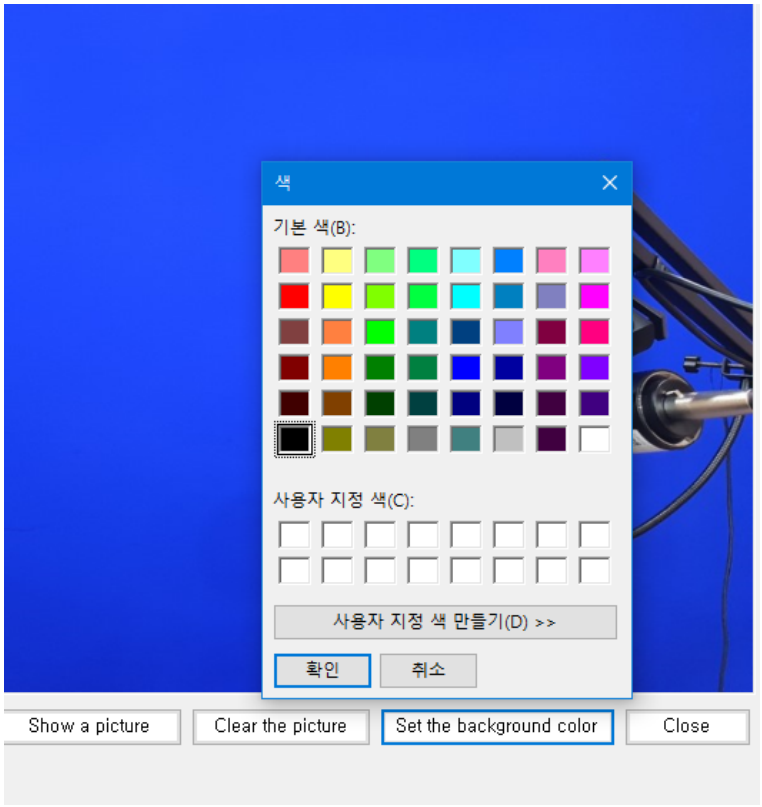
컨트롤에 대한 이벤트 처리기 추가

- 대화 상자를 여는 코드 작성
  - checkBox1을 더블클릭

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
        pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
    else
        pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
}
```

이미지 앱 실행

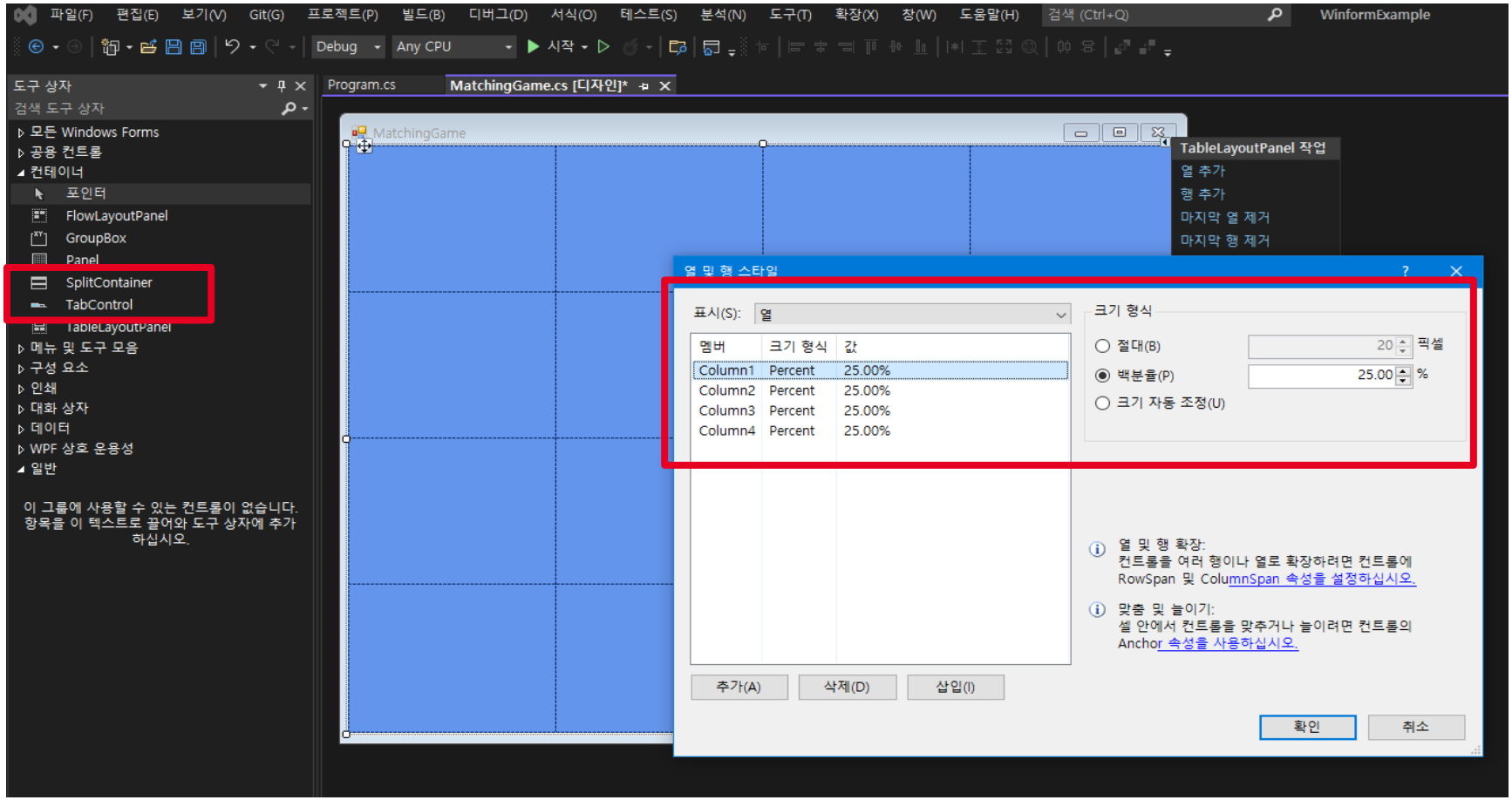

Picture Viewer
Stretch
Show a picture
Clear the picture
Set the background color
Close


색
기본 색(B):
사용자 지정 색(C):
사용자 지정 색 만들기(D) >>
확인
취소
Show a picture
Clear the picture
Set the background color
Close

매칭 게임 WinForms 앱 만들기

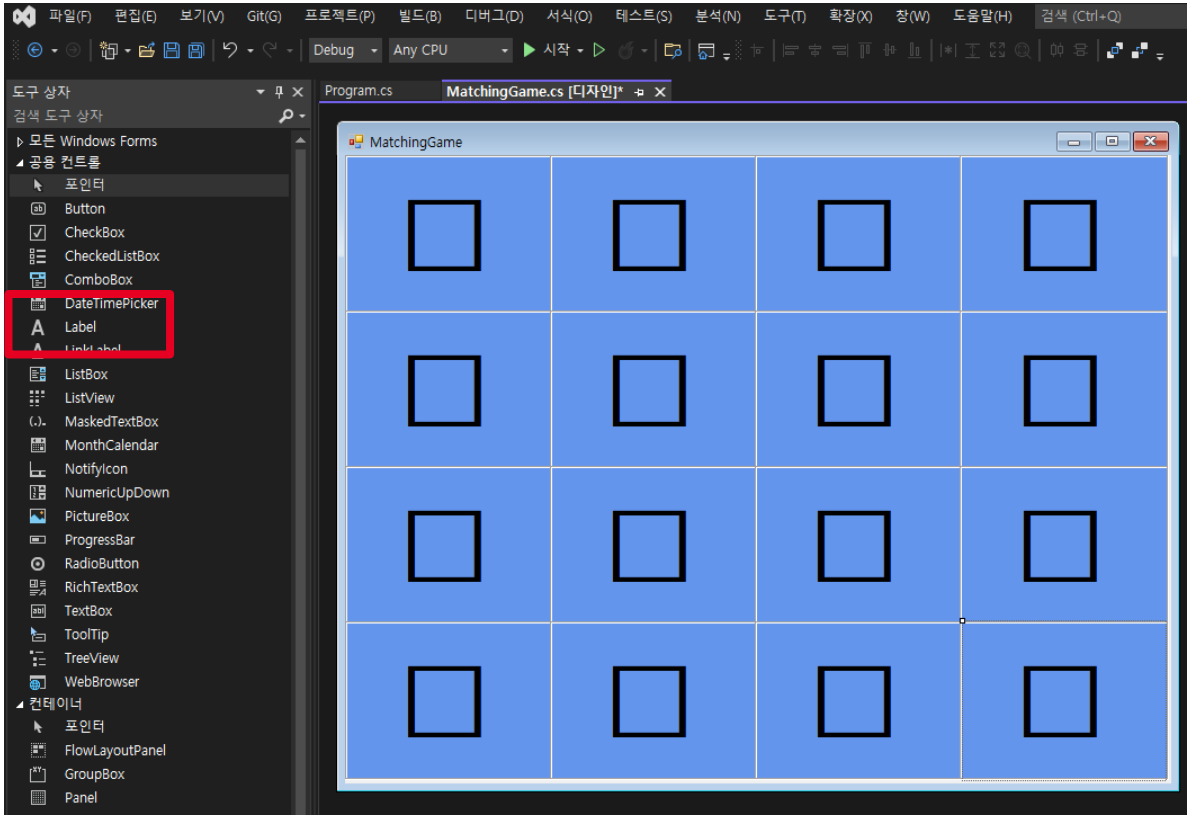
매칭 게임 WinForms 앱 만들기

- 도구 상자의 컨테이너 범주에서 TableLayoutPanel 컨트롤 선택하여 폼에 넣고 도킹
- CellBorderStyle 속성을 Inset 으로 설정한다. 이 값을 선택하면 보드의 각 셀 사이에 시각적 테두리가 생긴다.



매칭 게임 WinForms 앱 만들기

➢ 표시할 레이블 추가 및 서식 지정



➢ Label 삽입 후 속성값 설정:

- BackColor : CornflowerBlue
- AutoSize : False
- Dock : Fill
- TextAlign : MiddleCenter
- 글꼴 : Webdings, 굵게 크기48
- Text : 문자 c 로 설정

➢ 이 후 Label 컨트롤을 16개의 Cell 에 복사할 것

매칭 게임 WinForms 앱 만들기

➢ 임의의 개체 및 아이콘 목록 추가

- Visual Studio를 엽니다. 매칭 게임 프로젝트가 최근 항목 열기 아래에 나타난다.
- C#을 사용하는 경우 Form1.cs 를 선택하고 Visual Basic를 사용하는 경우 Form1.vb 를 선택합니다. 그런 다음 보기 〉 코드 를 선택한다. 또는 F7 키를 선택하거나 Form1 을 두 번 클릭한다. Visual Studio IDE에 Form1의 코드 모듈이 표시된다.
- 기존 코드에 다음 코드를 추가한다.

```
public partial class MatchingGame : Form
{
    Random random = new Random();

    List<string> icons = new List<string>()
    {
        "!", "!", "N", "N", ",", ",", "k", "k",
        "b", "b", "v", "v", "w", "w", "z", "z"
    };
```

매칭 게임 WinForms 앱 만들기

- 각 레이블에 임의 아이콘 할당
  - AssignIconsToSquares() 메서드를 추가한다.

```
private void AssignIconsToSquares()
{
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            int randomNumber = random.Next(icons.Count);
            iconLabel.Text = icons[randomNumber];
            icons.RemoveAt(randomNumber);
        }
    }
}
```

```
public MatchingGame()
{
    InitializeComponent();
    AssignIconsToSquares();
}
```

매칭 게임 WinForms 앱 만들기

➢ 각 레이블에 임의 아이콘 할당

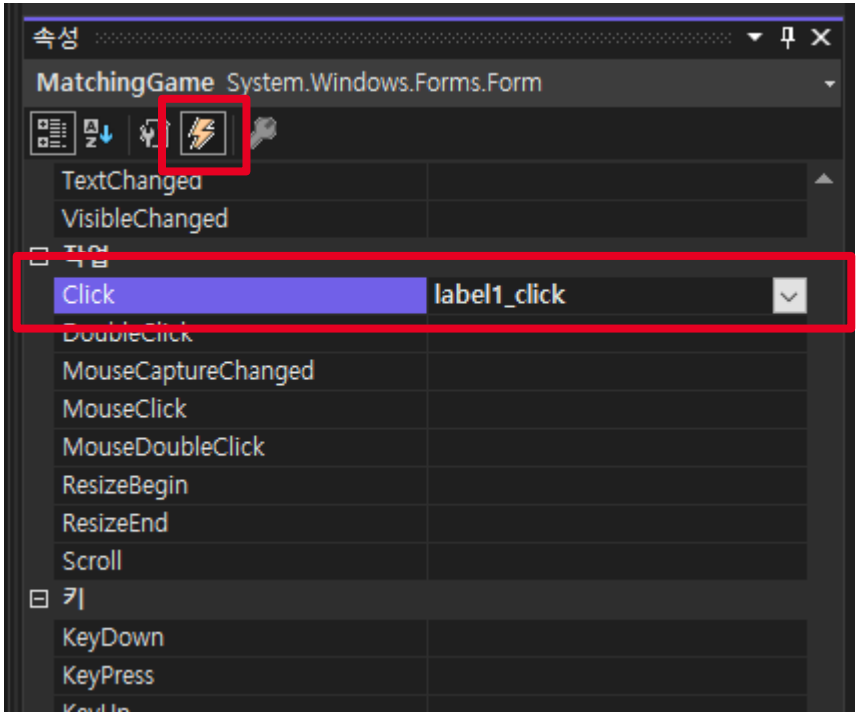
- AssignIconsToSquares() 메서드를 아래 코드 추가한다.

```
private void AssignIconsToSquares()
{
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            int randomNumber = random.Next(icons.Count);
            iconLabel.Text = icons[randomNumber];
            icons.RemoveAt(randomNumber);

            //글자를 안보이게 보임
            iconLabel.ForeColor = iconLabel.BackColor;
        }
    }
}
```

매칭 게임 WinForms 앱 만들기

➢ 레이블에 이벤트 처리기 추가

- Windows Forms 디자이너 에서 폼을 엽니다. Form1.cs 를 선택한 다음 보기 〉 디자이너를 선택한다.
- 첫 번째 레이블 컨트롤을 선택한다. 그런 다음 Ctrl 키를 누른 상태에서 다른 레이블을 각각 선택한다. 모든 레이블이 선택되어야 한다.
- 속성 창에서 번개 모양의 이벤트 버튼을 선택한다. 클릭 이벤트의 경우 상자에서 label1_Click 을 선택한다.



매칭 게임 WinForms 앱 만들기

- 레이블에 이벤트 처리기 추가
  - label1_click 메소드 추가 코드

```
private void label1_click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;

    if (clickedLabel != null)
    {
        if (clickedLabel.ForeColor == Color.Black)
            return;

        clickedLabel.ForeColor = Color.Black;
    }
}
```

매칭 게임 WinForms 앱 만들기

➢ 레이블 참조 추가

- 다음 코드를 사용하여 폼에 레이블 참조를 추가한다.

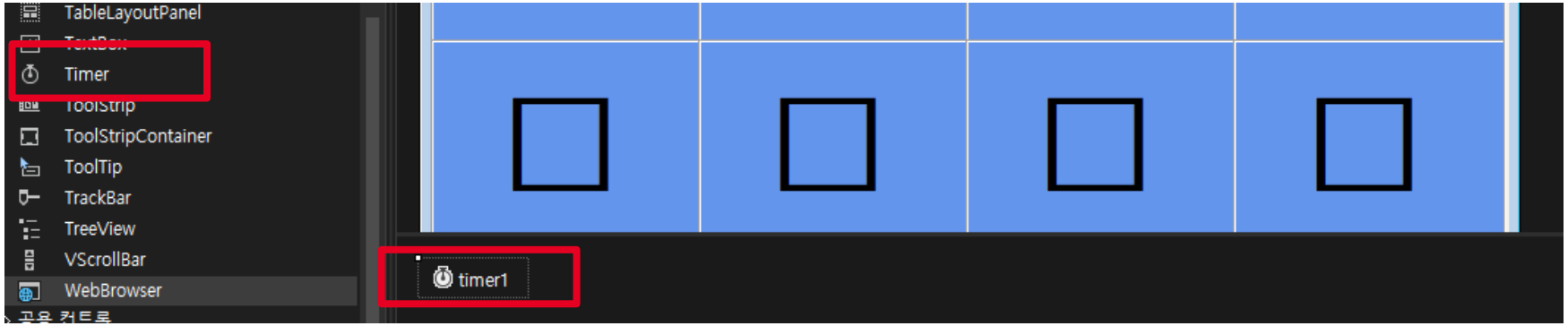
```
public partial class MatchingGame : Form
{
    Label firstClicked = null;
    Label secondClicked = null;

    Random random = new Random();

    List<string> icons = new List<string>()
    {
        "!", "!", "N", "N", ",", ",", "k", "k",
        "b", "b", "v", "v", "w", "w", "z", "z"
    };
```

```
private void label1_Click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;

    if (clickedLabel != null)
    {
        if (clickedLabel.ForeColor == Color.Black)
            return;

        if (firstClicked == null)
        {
            firstClicked = clickedLabel;
            firstClicked.ForeColor = Color.Black;

            return;
        }
    }
}
```

매칭 게임 WinForms 앱 만들기

➢ 타이머 추가

- 도구 상자 탭을 선택하고 구성 요소 범주에서 타이머 구성 요소를 두 번 클릭하거나 폼으로 끈다. 폼 아래 공간에 timer1 이라는 타이머 아이콘이 나타난다.




```
private void timer1_Tick(object sender, EventArgs e)
{
    timer1.Stop();

    firstClicked.ForeColor = firstClicked.BackColor;
    secondClicked.ForeColor = secondClicked.BackColor;

    firstClicked = null;
    secondClicked = null;
}
```

매칭 게임 WinForms 앱 만들기

➢ 플레이어가 이겼는지 확인

- label1_Clicek 메서드 코드 추가

```
if (firstClicked == null)
{
    firstClicked = clickedLabel;
    firstClicked.ForeColor = Color.Black;

    return;
}
secondClicked = clickedLabel;
secondClicked.ForeColor = Color.Black;

if (firstClicked.Text == secondClicked.Text)
{
    firstClicked = null;
    secondClicked = null;
    return;
}

timer1.Start();
```

매칭 게임 WinForms 앱 만들기

➢ 플레이어가 이겼는지 확인

- CheckForWinner 메서드 추가

```
private void CheckForWinner()
{
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;

        if (iconLabel != null)
        {
            if (iconLabel.ForeColor == iconLabel.BackColor)
                return;
        }
    }

    MessageBox.Show("You matched all the icons!", "Congratulations");
    Close();
}
```

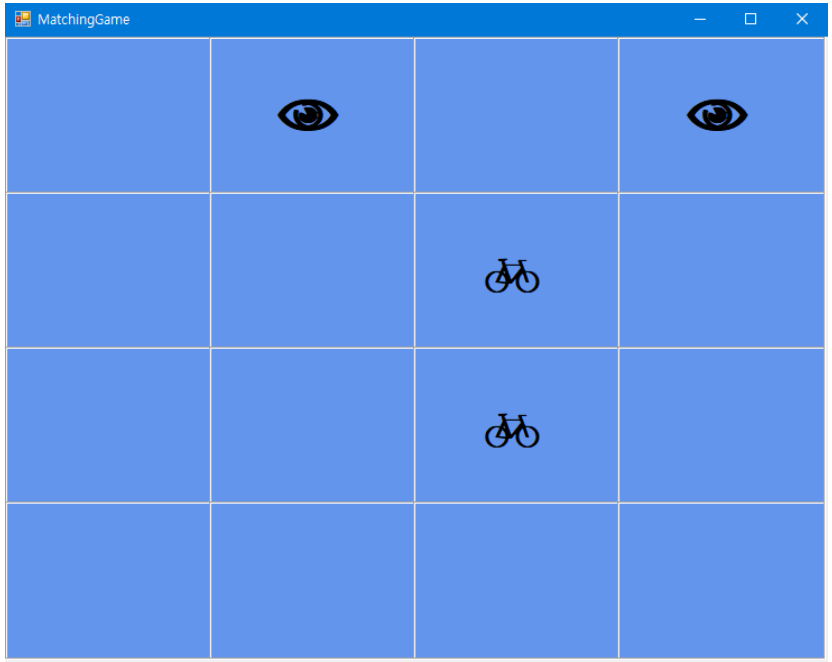
매칭 게임 WinForms 앱 만들기

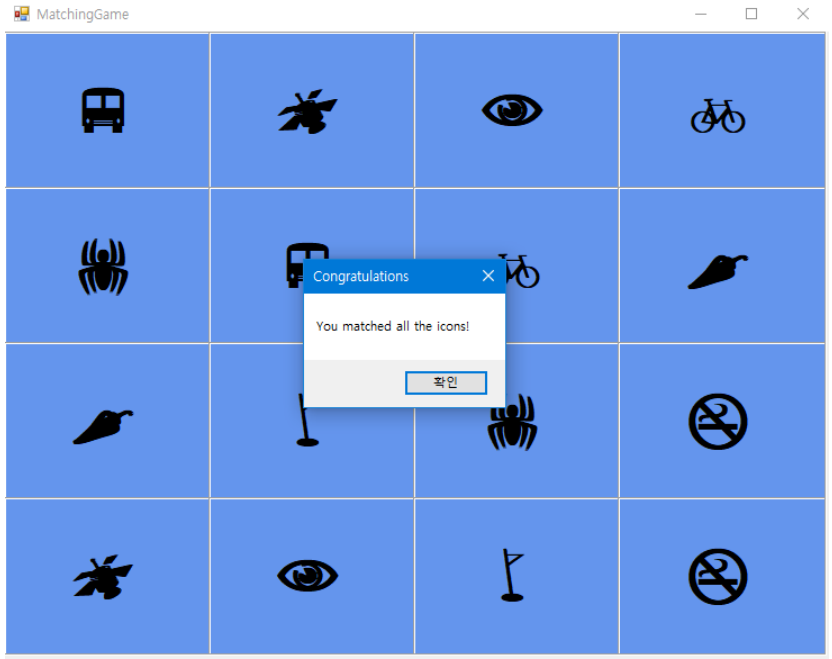
➢ 플레이어가 이겼는지 확인

- label1_Clicek 메서드 코드 추가

```
if (firstClicked == null)
{
    firstClicked = clickedLabel;
    firstClicked.ForeColor = Color.Black;

    return;
}
secondClicked = clickedLabel;
secondClicked.ForeColor = Color.Black;

CheckForWinner();

if (firstClicked.Text == secondClicked.Text)
{
    firstClicked = null;
    secondClicked = null;
    return;
}

timer1.Start();
```

매칭 게임 WinForms 앱 만들기





간단한 WinForms 프로그램

간단한 WinForms 프로그램

- 위의 윈폼 프로그램은 Program.cs와 Form1.cs/Form1.Designer.cs 파일을 생성한다.
- 우선 프로그램 시작 포인트인 Main()을 살펴 보면, 이 메인에서는 Form1 클래스이 객체를 하나 생성하여, Application.Run()에 파라미터로 넣고 실행한다.
- Application.Run()은 Form 객체를 화면에 보여주고, 메시지 루프를 만들어 마우스,키보드 등의 입력을 UI (User Interface) 쓰레드에 전달하는 기능을 한다.

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

namespace WindowsFormsApp1
{
    internal static class Program
    {
        /// <summary>
        /// 해당 애플리케이션의 주 진입점입니다.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

간단한 WinForms 프로그램

- 실제 폼 UI는 Form1.cs과 Form1.Designer.cs 파일에서 정의되는데, Form1.designer.cs는 폼에 포함되는 모든 UI 컨트롤 등에 대한 정보를 가지고 있으며 실행 시 이 정보를 기반으로 UI를 그리게 된다.

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

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

간단한 WinForms 프로그램

- 또한 Visual Studio는 이 파일을 기반으로 폼을 해석(Rendering)해서 VS 디자이너에 보여주게 된다. Form1.cs는 Form1.designer.cs와 동일한 클래스로서 주로 UI의 이벤트를 핸들링 하는 코드들을 작성하게 된다.

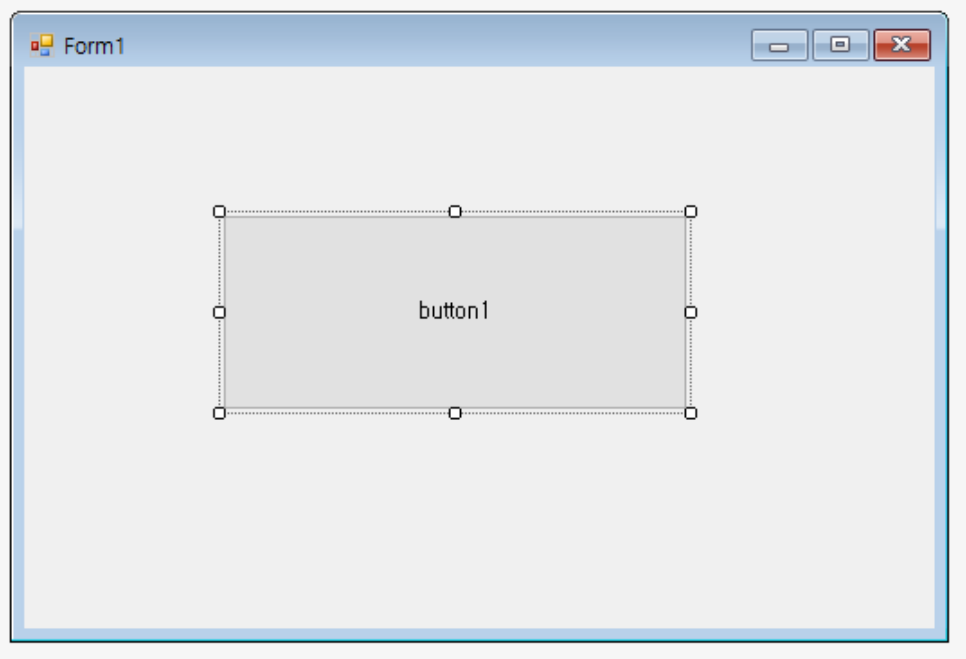
```
namespace WindowsFormsApp1
{
    partial class Form1
    {
        /// <summary>
        /// 필수 디자이너 변수입니다.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 사용 중인 모든 리소스를 정리합니다.
        /// </summary>
        /// <param name="disposing">관리되는 리소스를 삭제해야 하면 true이고,
그렇지 않으면 false입니다.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

간단한 WinForms 프로그램


```
#region Windows Form 디자이너에서 생성한 코드
        /// <summary>
        /// 디자이너 지원에 필요한 메서드입니다.
        /// 이 메서드의 내용을 코드 편집기로 수정하지 마세요.
        /// </summary>
        private void InitializeComponent()
        {
            this.SuspendLayout();
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 12F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(506, 312);
            this.Name = "Form1";
            this.Text = "Form1";
            this.ResumeLayout(false);

        }
        #endregion
    }
}
```

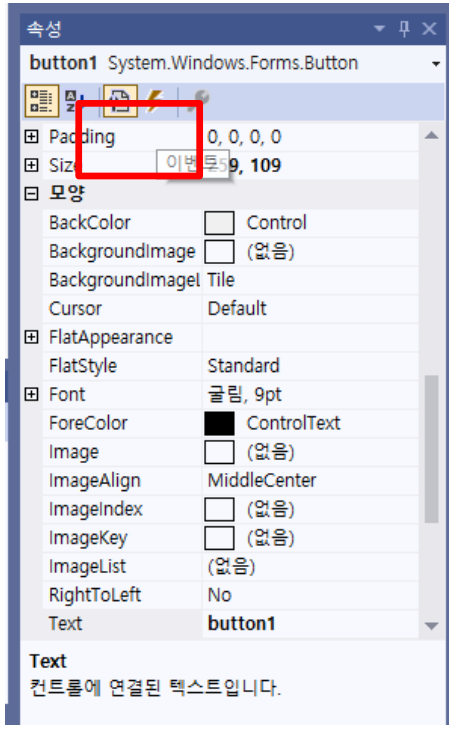
Button 컨트롤

- 모든 표준 윈도우 컨트롤들은 도구상자(ToolBox)에서 폼으로 끌어다 놓는 방식(Drag and Drop)으로 폼 위에 생성할 수 있다.
- 버튼의 경우 도구 상자의 Common Controls 카테고리에서 끌어다 폼에 올려 놓고 사이즈를 조정하면 된다.
- 컨트롤이 클릭된 상태에서 속성창(Property Window)에서 각 컨트롤의 속성을 새로 지정할 수 있다.



Button 이벤트 핸들링

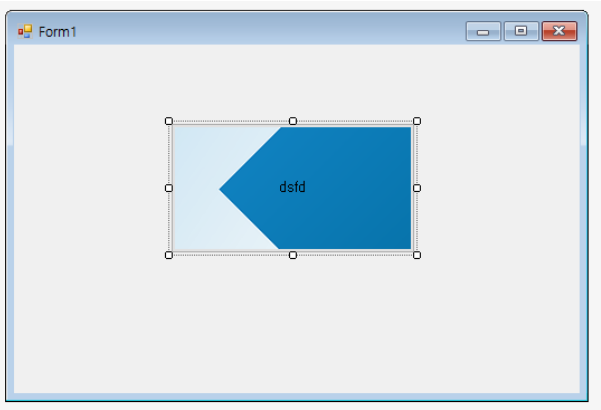
- 컨트롤들은 외부와 통신하기 위해 이벤트를 가지고 있는데, 어떤 이벤트들이 있는지를 알기 위해서는 속성창의 이벤트 버튼을 눌러 확인한다.
- 기본 디폴트 이벤트의 경우 컨트롤을 더블 클릭하면, 바로 이벤트 핸들러 코드가 생성된다.
- 컨트롤 디폴트 이벤트가 아닌 경우는, 속성창에서 해당 이벤트를 찾아 더블 클릭하면 된다.
- 버튼의 경우 더블 클릭하면 디폴트 이벤트인 Click 이벤트 핸들러가 자동으로 생성된다

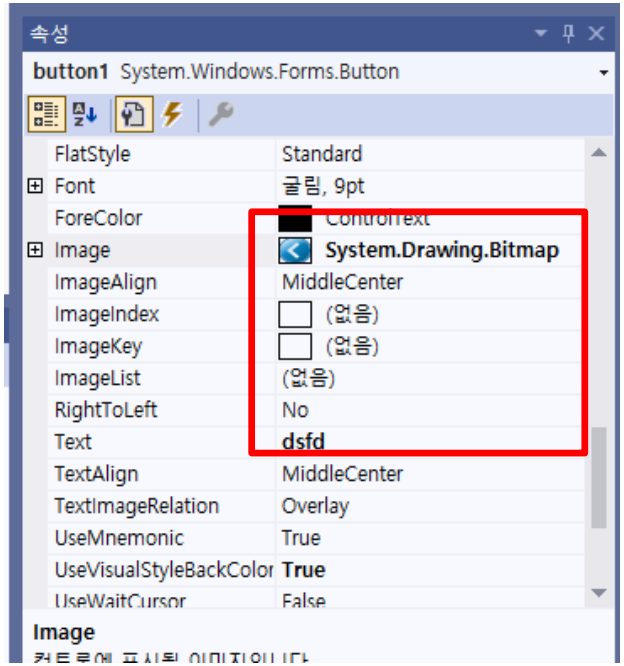



```
namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            button1.BackColor = Color.Cyan;
            button1.ForeColor = Color.Blue;
            button1.Text = "Processing...";
        }
    }
}
```

이미지 버튼

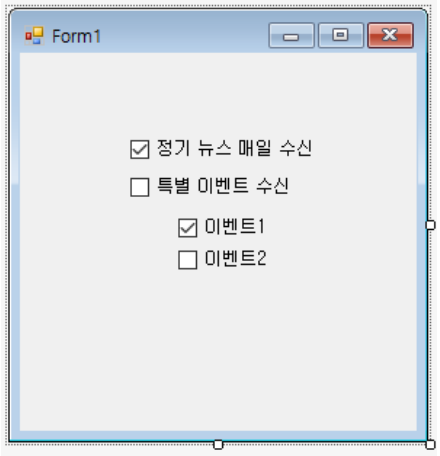
- 윈폼의 Button 컨트롤은 버튼 안에 텍스트 혹은 이미지 넣을 수 있으며, 텍스트와 이미지를 모두 넣을 수도 있다.
- 이미지를 넣기 위해서는 ToolBox에서 ImageList 를 드래그하여 폼에 끌어다 두고, 하단에 표시된 ImageList를 클릭하여 속성창에서 Images에 이미지들을 넣으면 된다.
- 또한 ImageList의 속성창에서 ImageSize를 변경하여 크기를 조정한다. ImageList가 완성되었으면, Button 속성창에서 ImageList를 지정하고, ImageList 안에 포함된 이미지의 인덱스를 ImageIndex에 지정하면 된다.





CheckBox 컨트롤 및 핸들링

- CheckBox 컨트롤은 체크상자와 레이블로 이루어져 있다.
- 이 컨트롤에서 가장 많이 쓰는 속성은 Checked로서 True나 False를 지정할 수 있다. Checked 프로퍼티를 통해 체크상자의 UI를 나타낼 수 있다.
- CheckBox 컨트롤은 CheckState 속성: 세가지 체크상자 상태를 나타낼 수 있다. Checked, Unchecked, Intermediate 3가지 중 하나의 CheckState 속성을 가질 수 있는데, Intermediate은 보통 부분적으로 선택된 경우를 나타낸다.

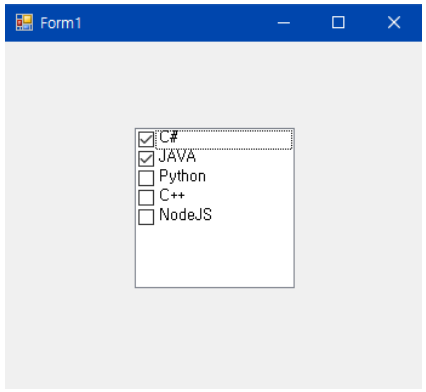



```
private void checkBox1_CheckedChanged(object sender,
EventArgs e)
        {
            checkBox1.Checked = checkBox1.Checked;
            checkBox1.Checked = checkBox1.Checked;
        }
```

- CheckBox 컨트롤의 이벤트중에 가장 많이 사용되는 이벤트인 CheckedChanged는 사용자가 체크박스를 클릭하여 체크상태가 변경된 경우에 호출된다.

CheckedListBox 컨트롤 및 핸들링

- CheckedListBox 컨트롤은 여러 개의 CheckBox들이 ListBox안에 들어 있는 Items Collection 컨트롤이다. 일종의 컨테이너 컨트롤이다.
- 일반적으로 Items라는 속성을 가지며, Items안에 Child 컨트롤들을 갖는다.
- Items에 고정된 데이터를 넣기 위해서는 속성 창의 Items 속성을 설정하거나, 초기화 코드에서 *checkedListBox1.Items.Add("대한민국");* 와 같이 데이터를 직접 넣을 수 있다.
- 가변적인 데이터는 주로 데이터 바인딩(Data Binding)을 사용한다.

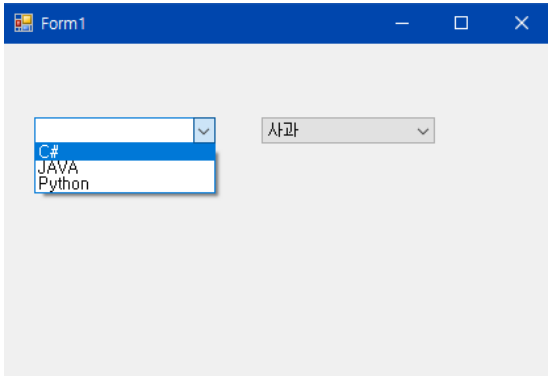



```
public Form1()
    {
        InitializeComponent();
        checkedListBox1.SetItemChecked(0, true);
        checkedListBox1.SetItemChecked(1, true);
    }
private void checkedListBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        int index = checkedListBox2.SelectedIndex;
        string item = checkedListBox2.SelectedItem.ToString();
        Console.WriteLine(index + "/" + item + "이 선택됨");
    }
```

- SelectedIndexChanged: 사용자가 컨테이너 내부 아이템들 중에서 어떤 아이템을 선택했을 때 발생하는 이벤트이다.

ComboBox 컨트롤 및 핸들링

- ComboBox 컨트롤은 여러개의 아이템들 중에 하나를 고를 때 사용한다.
- Items Collection 컨트롤이므로 Items 속성에 데이터를 지정한다.
- ComboBox 컨트롤의 UI 모드: Simple, DropDown, DropDownList
  - DropDown: Item값들 중에 하나를 선택하거나 사용자가 직접 다른 값을 쳐 넣을 수 있는 모드
  - DropDownList: 단지 기존 아이템들에서 하나만 선택할 수 있는 모드



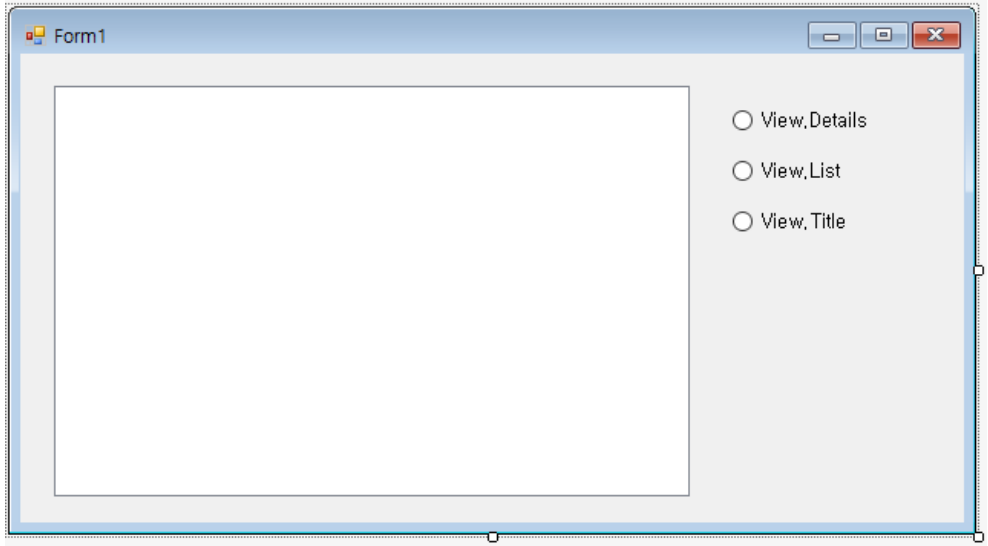

```
public Form1()
        {
            InitializeComponent();
            comboBox1.Items.Add("Item 1");
            comboBox1.Items.Clear();
            comboBox1.Items.AddRange(new string[] { "C#", "JAVA", "Python" });
        }
```

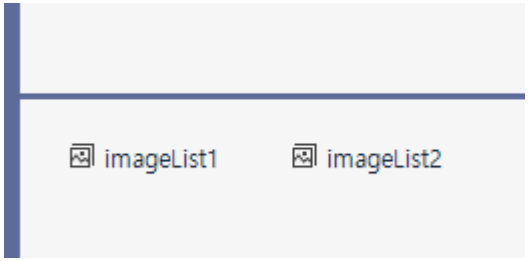
ListView 컨트롤

① 빈 윈폼 프로젝트에 ListView 컨트롤 하나를 배치한다.

② ListViewe Item들 앞에 이미지를 넣기 위해 ImageList1, ImageList2 컨트롤을 아래와 같이 추가한다.

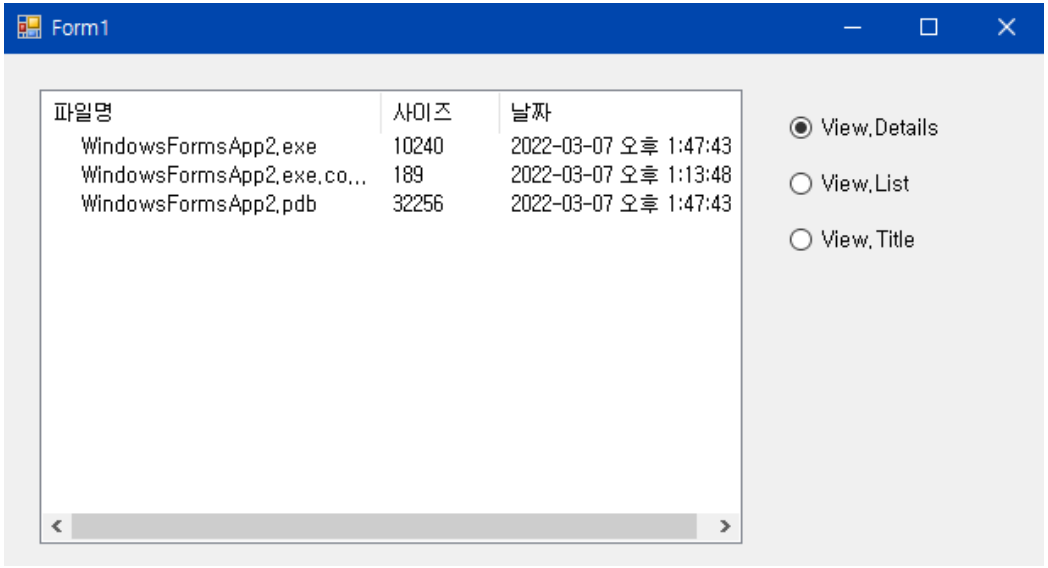
③ radiobutton 아래 그림처럼 Text 수정한다.





ListView 컨트롤

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
    {
        listView1.View = View.Details;
    }

    private void radioButton2_CheckedChanged(object sender, EventArgs e)
    {
        listView1.View = View.List;
    }

    private void radioButton3_CheckedChanged(object sender, EventArgs e)
    {
        listView1.View = View.Tile;
    }
```



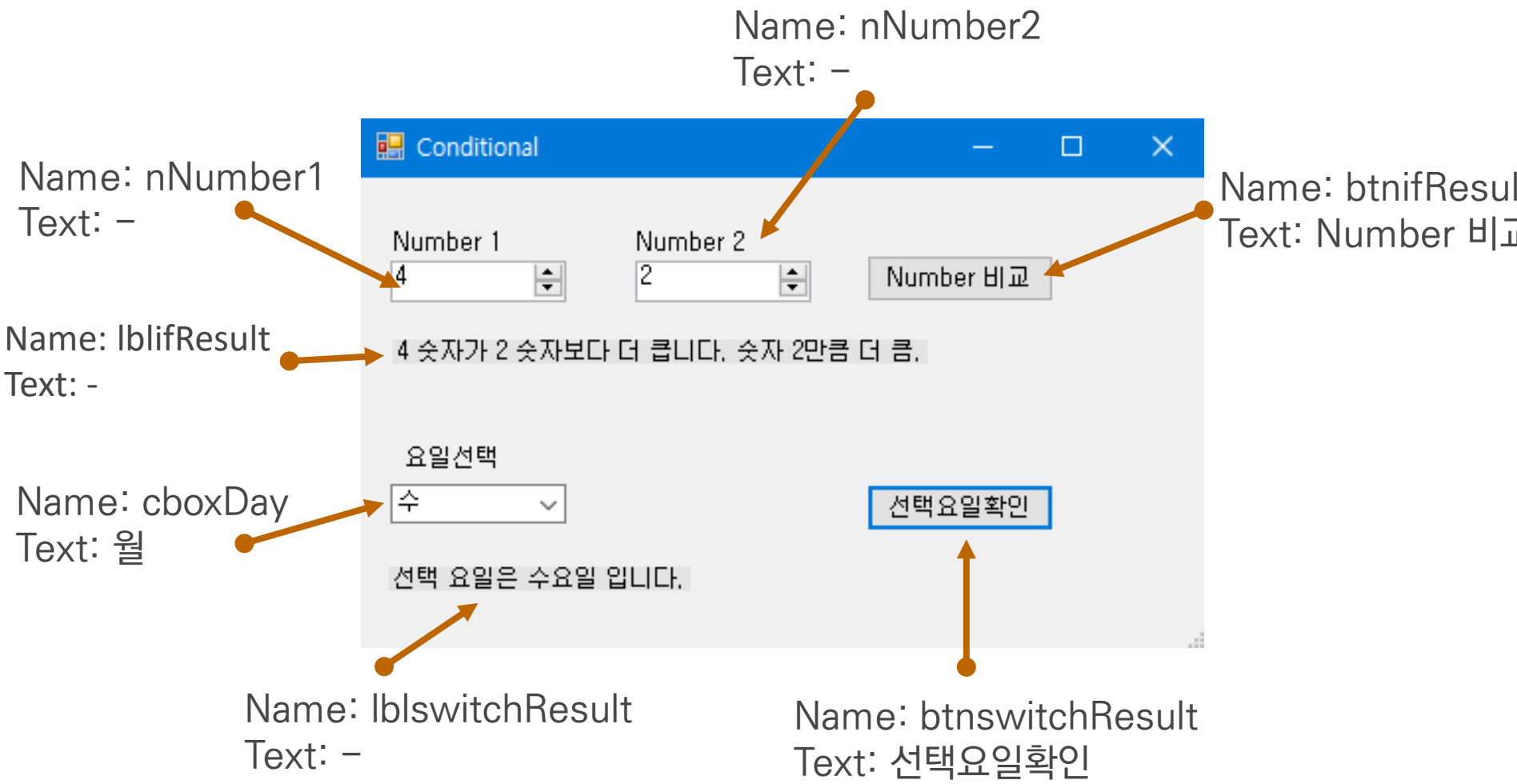
ListView 컨트롤


```
private void Form1_Load(object sender, EventArgs e) {
        //현재 디렉토리 내의 파일리스트 얻기
        string currDir = Environment.CurrentDirectory;
        DirectoryInfo di = new DirectoryInfo(currDir);
        FileInfo[] files = di.GetFiles();

        // 리스트뷰 아이템을 업데이트 하기 시작.
        // 업데이트가 끝날 때까지 UI 갱신 중지.
        listView1.BeginUpdate();

        // 뷰모드 지정
        listView1.View = View.Details;

        // 아이콘을 위해 이미지 지정
        listView1.LargeImageList = imageList1;
        listView1.SmallImageList = imageList2;
```

```
foreach (var fi in files) {
        // 각 파일별로 ListViewItem객체를 하나씩 만듦
        // 파일명, 사이즈, 날짜 정보를 추가
        ListViewItem lvi = new ListViewItem(fi.Name);
        lvi.SubItems.Add(fi.Length.ToString());
        lvi.SubItems.Add(fi.LastWriteTime.ToString());
        lvi.ImageIndex = 0;
        // ListViewItem객체를 Items 속성에 추가
        listView1.Items.Add(lvi);
}

// 컬럼명과 컬럼사이즈 지정
listView1.Columns.Add("파일명", 200,
HorizontalAlignment.Left);
        listView1.Columns.Add("사이즈", 70,
HorizontalAlignment.Left);
        listView1.Columns.Add("날짜", 150,
HorizontalAlignment.Left);

 // 리스뷰를 Refresh하여 보여줌
 listView1.EndUpdate();
}
```

WinForm기반의 if conditional

➢ 화면 디자인 구성



WinForm기반의 if conditional

➢ 코드 구성 : public Conditional()

```
public partial class Conditional : Form {
    public Conditional()
    {
        InitializeComponent();
        lblifResult.Text = "-";
        lblswitchResult.Text = "-";
    }
```

WinForm기반의 if conditional

➢ 코드 구성 : ifTest()

```
private void ifTest()
{
    int number1 = Convert.ToInt32(nNumber1.Value.ToString());
    int number2 = Convert.ToInt32(nNumber2.Value.ToString());

    if (number1 == number2)
    {
        lblifResult.Text = string.Format("{0} 숫자와 {1} 숫자는 같습니다. 숫자 {2} ", number1, number2, number1);
    }
    else if (number1 > number2)
    {
        int res = number1 - number2;
        lblifResult.Text = string.Format("{0} 숫자가 {1} 숫자보다 더 큽니다. 숫자 {2}만큼 더 큼.", number1, number2, res);
    }
    else if (number1 < number2)
    {
        int res = number2 - number1;
        lblifResult.Text = string.Format("{0} 숫자가 {1} 숫자보다 더 작습니다. 숫자 {2}만큼 더 작음.", number1, number2, res);
    }
}
```

WinForm기반의 if conditional

➢ 코드 구성 : switchTest()

```
private void switchTest()
{
    string selectDay = cboxDay.Text;

    switch (selectDay) {
        case "월":
            lblswitchResult.Text = "선택 요일은 월요일 입니다.";
            break;
        case "화":
            lblswitchResult.Text = "선택 요일은 화요일 입니다.";
            break;
        case "수":
            lblswitchResult.Text = "선택 요일은 수요일 입니다.";
            break;
        case "목":
            lblswitchResult.Text = "선택 요일은 목요일 입니다.";
            break;
        case "금":
            lblswitchResult.Text = "선택 요일은 금요일 입니다.";
            break;
        case "토":
            lblswitchResult.Text = "선택 요일은 토요일 입니다.";
            break;
        case "일":
            lblswitchResult.Text = "선택 요일은 일요일 입니다.";
            break;
    }
}
```

WinForm기반의 if conditional

➢ 코드 구성 : btnifResult_Click, btnswitchResult_Click

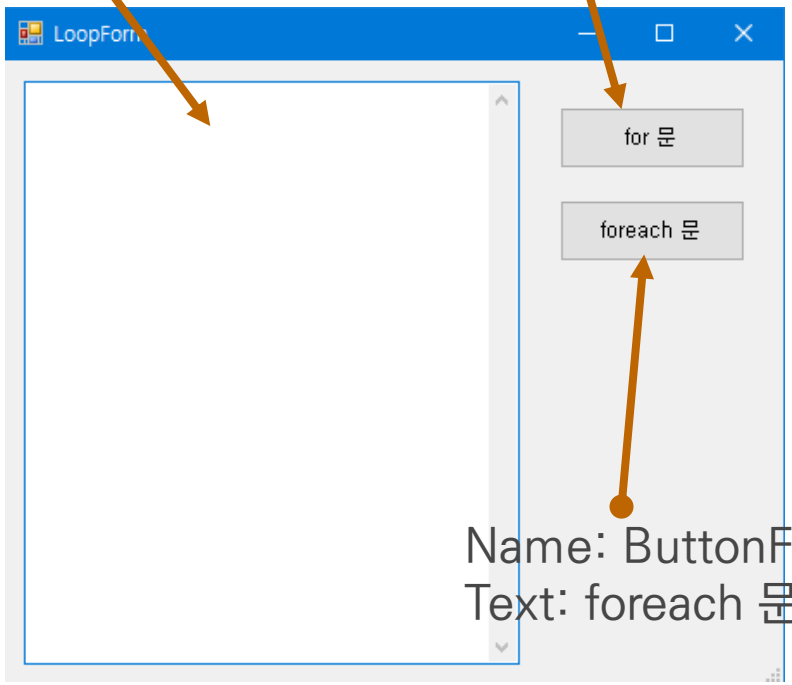
```
private void btnifResult_Click(object sender, EventArgs e)
{
    ifTest();
}

private void btnswitchResult_Click(object sender, EventArgs e)
{
    switchTest();
}
```

WinForm기반의 if conditional

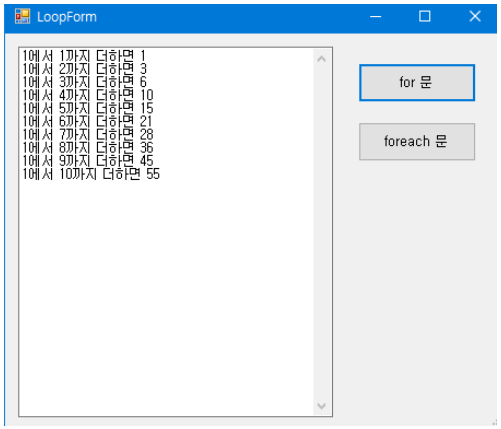
➢ 화면 디자인 구성

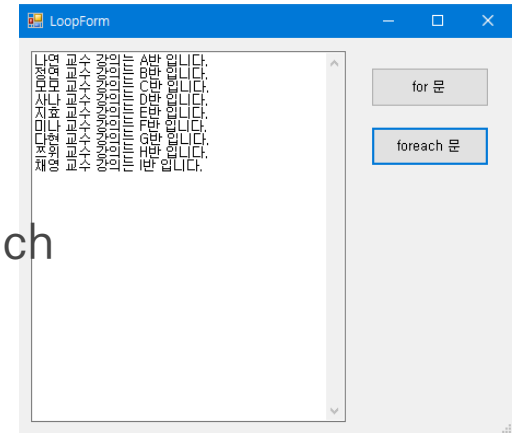
Name: textBoxResult
Text: MulitLine 체크

Name: ButtonFor
Text: for문



Name: ButtonForeach
Text: foreach 문





WinForm기반의 if conditional

➢ 코드 구성 : ButtonFor_Click()

```
public partial class LoopForm : Form {
    public LoopForm()
    {
        InitializeComponent();
    }
    private void ButtonFor_Click(object sender, EventArgs e)
    {
        textBoxResult.Text = string.Empty;
        StringBuilder sb = new StringBuilder();
        int iResult = 0;

        for (int i = 1; i <= 10; i++)
        {
            iResult = iResult + i;
            sb.Append(String.Format("1에서 {0}까지 더하면 {1} \r\n", i, iResult));
        }
        textBoxResult.Text = sb.ToString();
    }
```

WinForm기반의 for, foreach

➢ 코드 구성 : ButtonForeach_Click()

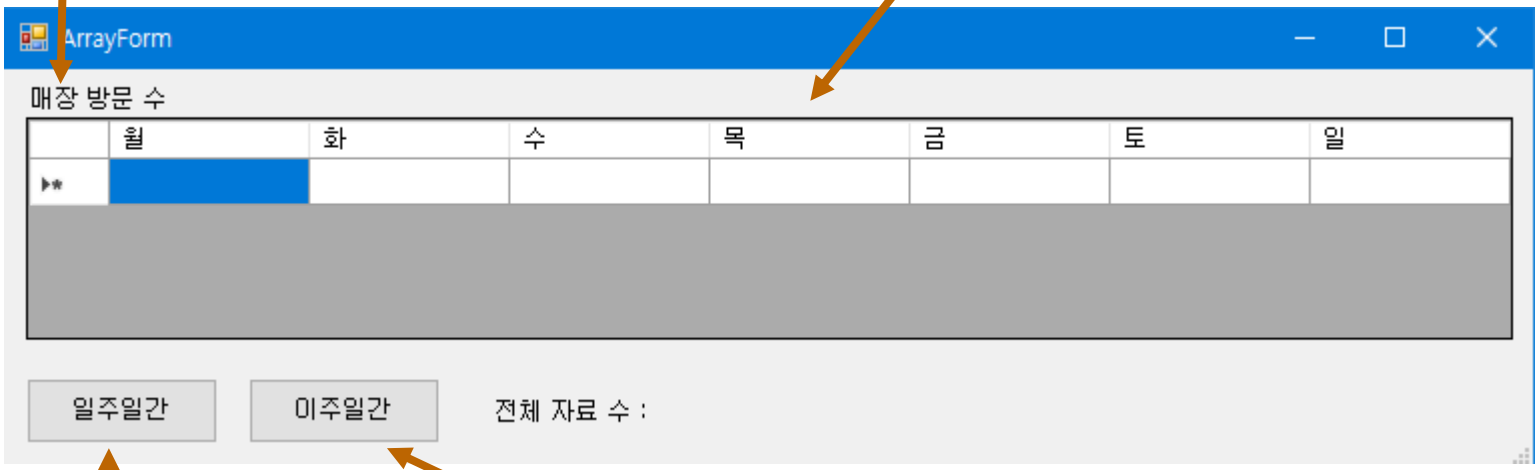
```
private void ButtonForeach_Click(object sender, EventArgs e)
{
    textBoxResult.Text = String.Empty;
    StringBuilder sb = new StringBuilder();

    int i = 65;
    string[] strArray = { "나연", "정연", "모모", "사나", "지효", "미나", "다현", "쯔위", "채영" };
    foreach (string str in strArray)
    {
        sb.Append(String.Format("{0} 교수 강의는 {1}반 입니다. \r\n", str, (char)i++));
    }
    textBoxResult.Text = sb.ToString().Trim();
}
```

WinForm기반의 Array, DataGridView

➢ 화면 디자인 구성

Text: 매장 방문 수

Name: dgDay
Text: –



Name: button1
Text: 일주일간

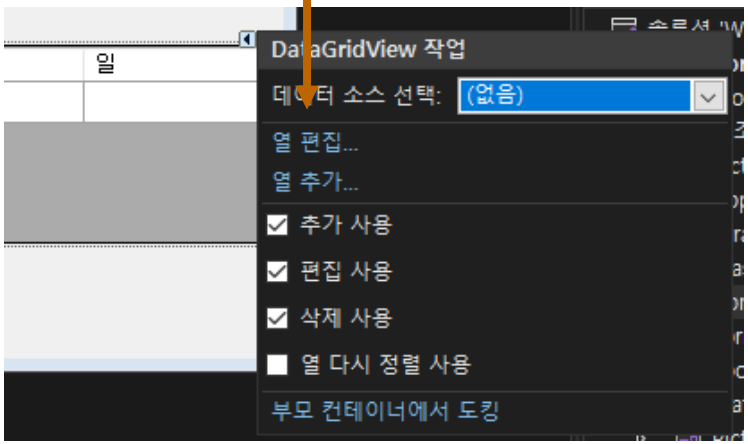
Name: button2
Text: 이주일간

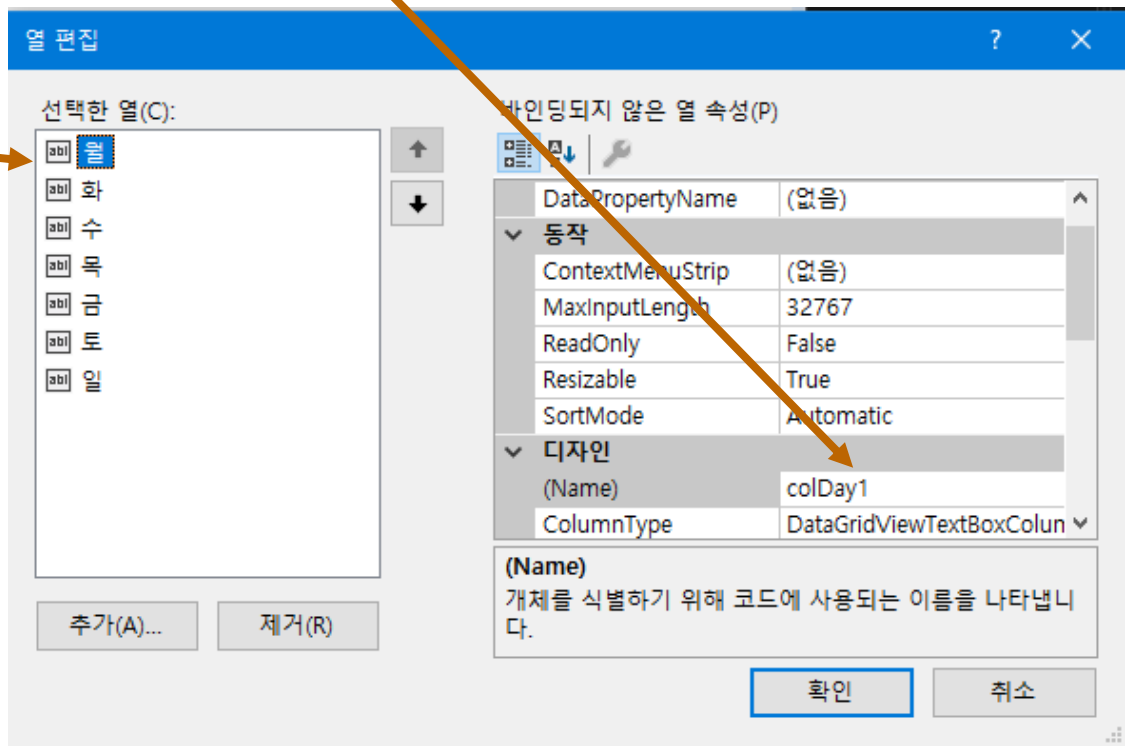
WinForm기반의 Array, DataGridView

➢ 화면 디자인 구성

Name: colDay1, colDay2, colDay3, … colDay7
HeaderText: 월, 화, 수, … 일

열 편집





WinForm기반의 Array, DataGridView

➢ 코드 구성: button1_Click

```
private void button1_Click(object sender, EventArgs e)
{
    int[] iTest = { 10, 5, 30, 4, 15, 22, 18 };

    string iname;
    for(int i = 0; i < iTest.Length; i++)
    {
        iname = "colDay" + string.Format("{0}", i + 1);
        dgDay[iname, 0].Value = iTest[i];
    }
}
```

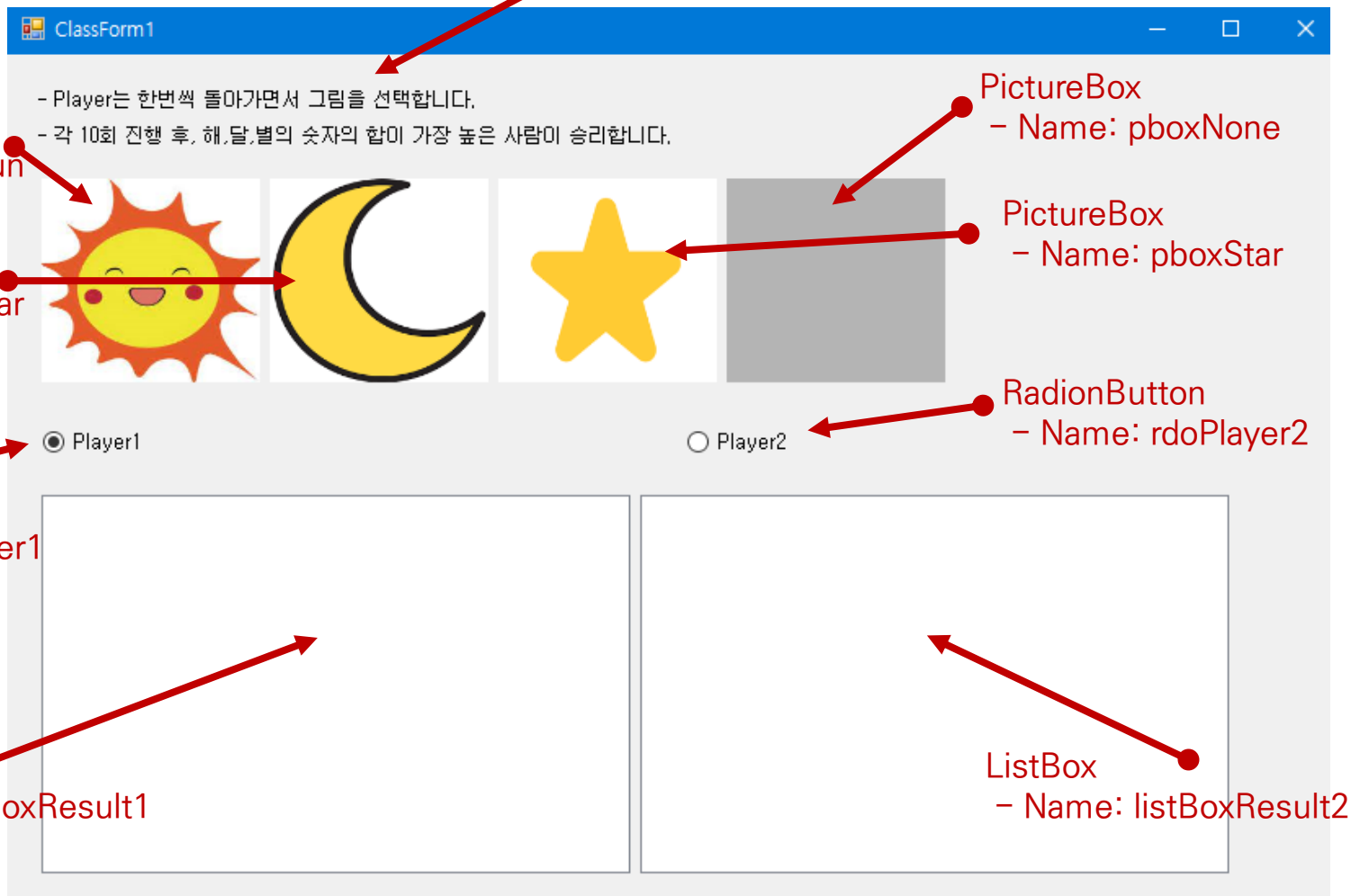
WinForm기반의 Array, DataGridView

➢ 코드 구성: button2_Click

```
private void button2_Click(object sender, EventArgs e)
{
    int[,] iTest = { { 10, 5,  30, 4,  15, 22, 18 },
                     { 11, 15, 25, 14, 7,  5,  29 } };

    dgDay.Rows.Add();
    string iname;

    for (int i = 0; i < iTest.GetLength(1); i++)
    {
        iname = "colDay" + string.Format("{0}", i + 1);
        dgDay[iname, 0].Value = iTest[0, i];
        dgDay[iname, 1].Value = iTest[1, i];
    }
}
```

WinForm기반의 Struct, Class

➢ 화면 디자인 구성: 해, 달, 별 점수 게임



WinForm기반의 struct, class

➢ 코드 구성 : struct structPlayer

```
public partial class ClassForm1 : Form {
    struct structPlayer {
        public int iCount; //플레이어가 몇회진행

        public int iSun;
        public int iMoon;
        public int iStar;

        public int iCardSum; //해,달,별 더한값

        //값들을 더해서 반환
        public int CardSun(int iSun, int iMoon, int iStart) => iSun + iMoon + iStart;

        public string ResultText() => string.Format("{0}회) 해:{1}, 달:{2}, 별:{3} => 합계는 {4}입니다."
            iCount, iSun, iMoon, iStar, iCardSum);
    }
```

WinForm기반의 struct, class

➢ 코드 구성 : struct structPlayer

```
structPlayer strPlayer1;
structPlayer strPlayer2;

Random rd = new Random();

public ClassForm1()
{
    InitializeComponent();
}
```

```
private void pboxSun_Click(object sender, EventArgs e)
{
    int iNumber = rd.Next(1, 21);
    if(rdoPlayer1.Checked)
        strPlayer1.iSun = iNumber;
    else
        strPlayer2.iSun = iNumber;

    Result();
}
```

```
private void pboxMoon_Click(object sender, EventArgs e)
{
    int iNumber = rd.Next(1, 21);
    if (rdoPlayer1.Checked)
        strPlayer1.iMoon = iNumber;
    else
        strPlayer2.iMoon = iNumber;

    Result();
}
```

WinForm기반의 struct, class

➢ 코드 구성 : struct structPlayer

```
structPlayer strPlayer1;
structPlayer strPlayer2;

Random rd = new Random();

public ClassForm1()
{
    InitializeComponent();
}
```

```
private void pboxStar_Click(object sender, EventArgs e)
{
    int iNumber = rd.Next(1, 21);
    if (rdoPlayer1.Checked)
        strPlayer1.iStar = iNumber;
    else
        strPlayer2.iStar = iNumber;

    Result();
}
```

```
private void pboxNone_Click(object sender, EventArgs e)
{
    Result();
}
```

WinForm기반의 struct, class

➢ 코드 구성 : struct structPlayer

```
private void Result()
{
    string strResult = string.Empty;
    if (rdoPlayer1.Checked) {
        strPlayer1.iCount++;
        strPlayer1.iCardSum = strPlayer1.CardSun(strPlayer1.iSun, strPlayer1.iMoon, strPlayer1.iStar);
        strResult = strPlayer1.ResultText();

        listBoxResult1.Items.Add(strResult);
    }
    else {
        strPlayer2.iCount++;
        strPlayer2.iCardSum = strPlayer2.CardSun(strPlayer2.iSun, strPlayer2.iMoon, strPlayer2.iStar);
        strResult = strPlayer2.ResultText();

        listBoxResult2.Items.Add(strResult);
    }
```

```
private void iCheckChange()
{
    if(rdoPlayer1.Checked)
        rdoPlayer2.Checked = true;
    else
        rdoPlayer1.Checked = true;
}
```

WinForm기반의 struct, class

➢ 코드 구성 : struct structPlayer

```
iCheckChange();

if (strPlayer1.iCount >= 5 && strPlayer2.iCount >= 5)
{
    if (strPlayer1.iCardSum > strPlayer2.iCardSum)
        MessageBox.Show("Player1이 이겼습니다.");
    else if (strPlayer1.iCardSum > strPlayer2.iCardSum)
        MessageBox.Show("Player2가 이겼습니다.");
    else
        MessageBox.Show("비겼습니다.");
}
```

```
private void iCheckChange()
{
    if(rdoPlayer1.Checked)
        rdoPlayer2.Checked = true;
    else
        rdoPlayer1.Checked = true;
}
```

간단한 WinForms 프로그램

Q & A