WinForm으로 만드는 사용자 인터페이스

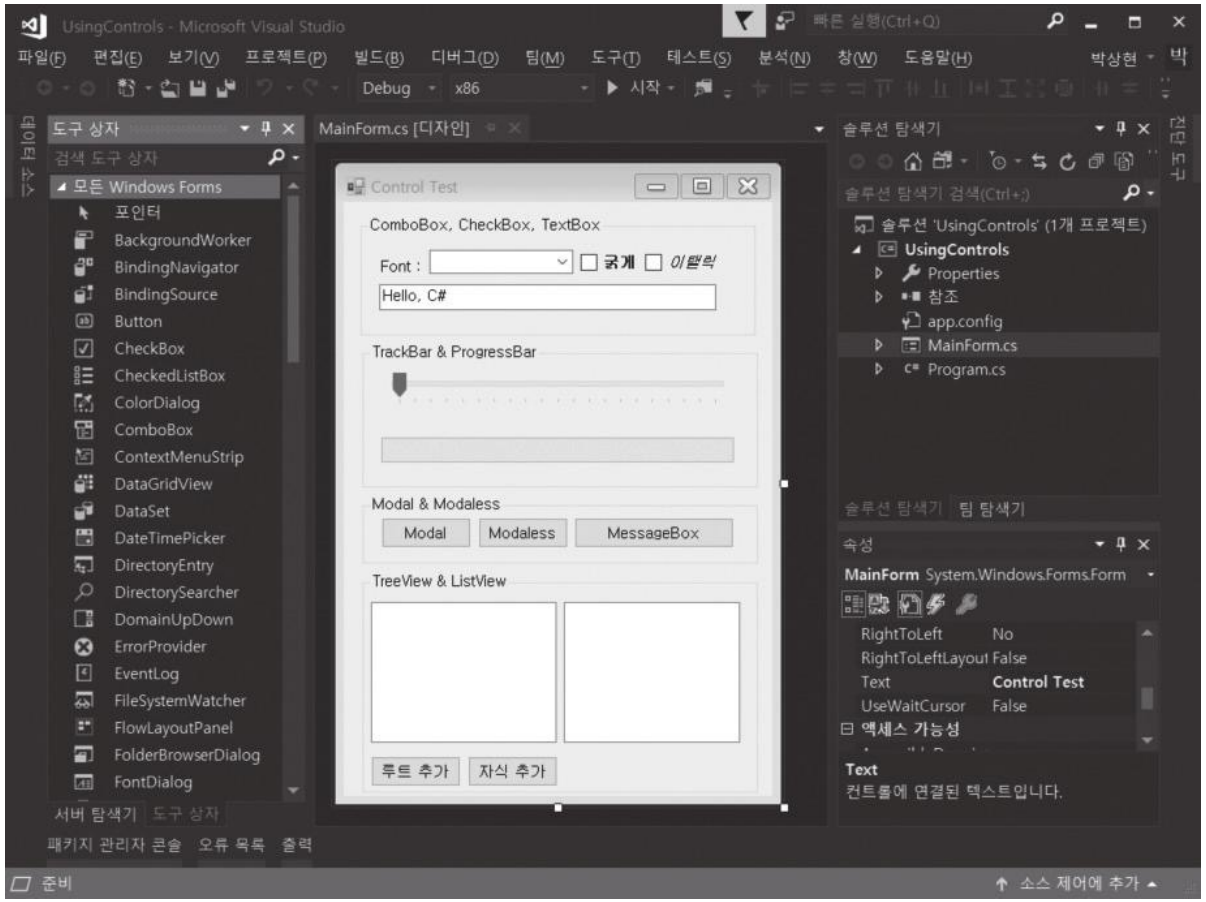
이것이 C#이다

Contents

- 도대체 무슨 일이 일어나고 있는 걸까?
- C# 코드로 WinForm 윈도우 만들기
- Application 클래스
- 윈도우를 표현하는 Form 클래스
- 폼 디자이너를 이용한 WinForm UI 구성
- 사용자 인터페이스와 비동기 작업

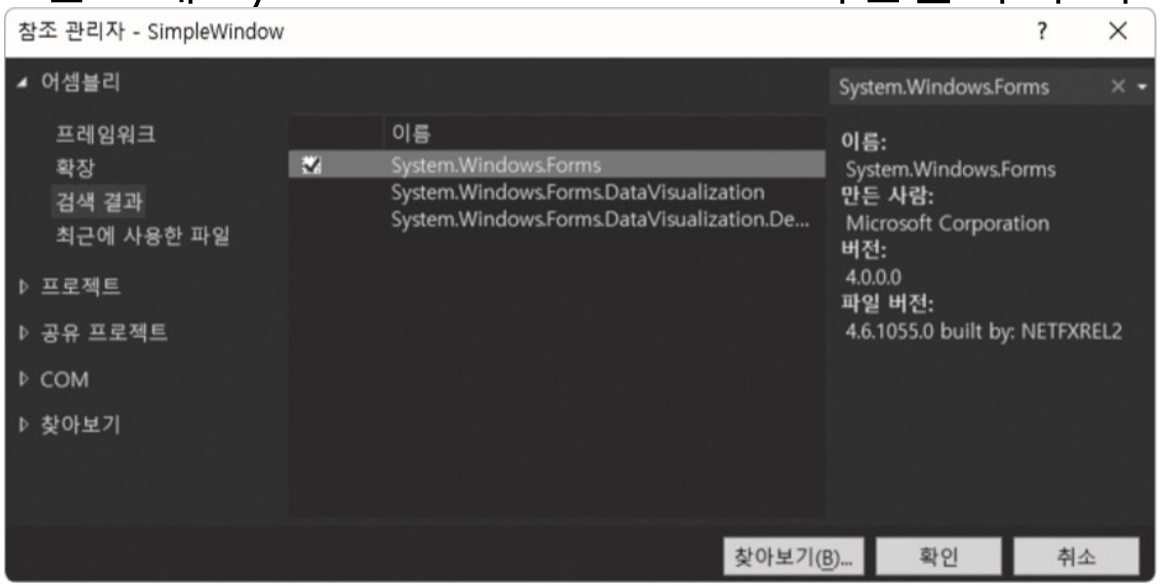
20.1 도대체 무슨 일이 일어나고 있는 걸까?

- **폼 디자이너**
  - 컨트롤을 끌어 다 윈도우 위에 배치
  - →WYSIWYG(What You See Is What You Get)
  - UI를 표시하는 한편, 뒤로는 관련 C# 코드 자동 생성
- **C# 코드만으로 GUI 구성**
  - 미세한 조정 필요
  - 폼 디자이너의 문제 발생



20.2 C# 코드로 WinForm 윈도우 만들기

- .NET 프레임워크의 WinForm 클래스 라이브러리
- WinForm 클래스를 이용한 윈도우 생성 절차
  - System.Windows.Forms.Form 클래스에서 파생된 윈도우 폼 클래스 선언
  - 앞서 만든 클래스의 인스턴스를 System.Windows.Forms.Application.Run() 메소드에 매개 변수로 넘겨 호출
- 데모 예제 – SimpleWindow
  - 콘솔 앱 프로젝트
  - 참조에 System.Windows.Forms 어셈블리 추가



20.3 Application 클래스 (1)

❖ 역할

- 윈도우 애플리케이션 시작과 종료
- 윈도우 메시지 처리

```
class MyForm : System.Windows.Forms.Form
{
}

class MainApp
{
    static void Main(string[] args)
    {
        MyForm form = new MyForm();
        form.Click += new EventHandler(
            (sender, eventArgs) =>
            {
                Application.Exit();
            });

        Application.Run(form);
    }
}
```

Form 클래스는 여러 가지 이벤트를 정의하고 있는데, 그 중 Click 이벤트는 윈도우를 클릭했을 때 발생하는 이벤트입니다.
따라서 이 코드는 윈도우를 클릭하면 Application.Exit()를 호출하도록 합니다.

❖ 데모 예제 - UsingApplication

20.3 Application 클래스 (2)

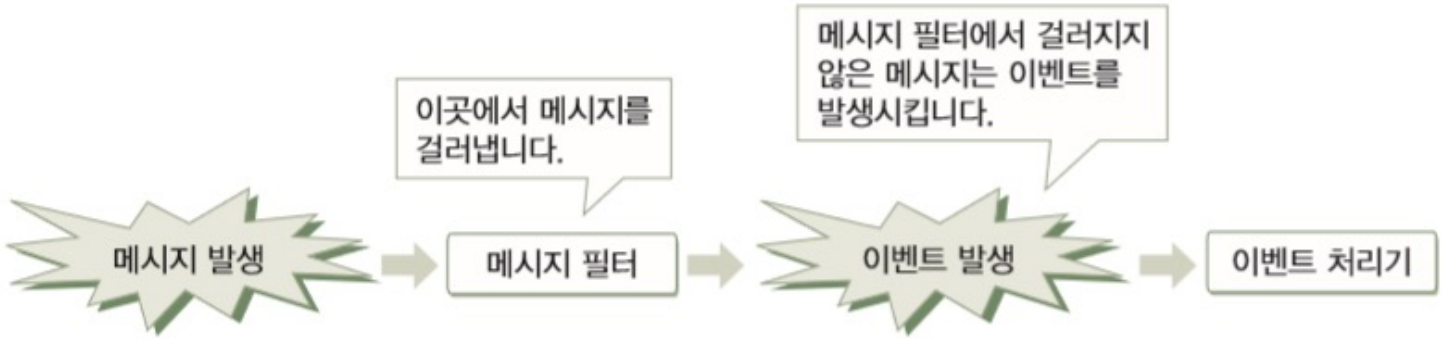
❖ 이벤트 기반(Event Driven) 방식

- 윈도우 애플리케이션의 동작 방식
- 갑자기 일어나는 사건(이벤트 : Event)에 반응해서 코드 실행
- 사용자의 클릭 → 인터럽트 → OS에서 애플리케이션에 메시지 전달

❖ 윈도우 메시지(Windows Message)

- 미리 시스템에 정의되어 있는 메시지
- 애플리케이션에서 자체 정의한 메시지
- 메시지의 식별번호(ID) – ex)WM_ LBUTTONDOWN(0x201)

❖ 메시지 필터링



20.3 Application 클래스 (3)

❖ Application 클래스의 메시지 필터링

- Application.AddMessageFilter() – 애플리케이션에 필터 설치
- →IMessageFilter를 구현하는 파생 클래스의 인스턴스를 매개변수로 사용
- → IMessageFilter는 PreFilterMessage() 메소드 구현 필수

```
public interface IMessageFilter
{
    bool PreFilterMessage(ref Message m);
}
```

❖ Message 구조체의 프로퍼티

프로퍼티	설명
HWnd	메시지를 받는 윈도우의 핸들(Handle)입니다. 핸들은 처음 보는 용어죠? 윈도우의 인스턴스를 식별하고 관리하기 위해 운영체제가 붙여놓은 번호가 바로 핸들입니다.
Msg	메시지 ID입니다.
LParam	메시지를 처리하는 데 필요한 정보가 담겨 있습니다.
WParam	메시지를 처리하는 데 필요한 부가 정보가 담겨 있습니다.
Result	메시지 처리에 대한 응답으로 윈도우 운영체제에 반환되는 값을 지정합니다.

❖ 데모 예제 – MessageFilter

20.4 윈도우를 표현하는 Form 클래스

- Form(과 컨트롤)에 정의된 이벤트와 이벤트 처리기 연결하기
- Form의 프로퍼티를 조절하여 윈도우 모양 바꾸기
- Form 위에 컨트롤 올리기

20.4.1 Form에 정의된 이벤트와 이벤트 처리기 연결하기

❖ Form/WinForm의 윈도우와 컨트롤 클래스에서 메시지를 위한 이벤트 사전 구현

```
// 이벤트 처리기 선언
private void Form_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)
{
    MessageBox.Show("안녕하세요!");
}
public MyForm()
{
// 이벤트 처리기를 이벤트에 연결
this.MouseDown += new System.Windows.Forms.MouseEventHandler (this.Form_MouseDown);
}
```

록

```
public event MouseEventHandler MouseDown;
```

```
public delegate void MouseEventHandler( object sender, MouseEventArgs e );
```

❖ 마우스 이벤트의 상세 정보 -MouseEventArgs의 프로퍼티

프로퍼티	설명
Button	마우스의 어떤 버튼(왼쪽, 오른쪽 또는 가운데)에서 이벤트가 발생했는지를 나타냅니다.
Clicks	마우스의 버튼을 클릭한 횟수를 나타냅니다. 사용자가 더블 클릭했을 때만 어떤 기능을 수행하고 싶다면 이 값이 2일 경우를 확인하면 됩니다.
Delta	마우스 휠의 회전 방향과 회전한 거리를 나타냅니다.
X	마우스 이벤트가 발생한 폼 또는 컨트롤 상의 x(가로) 좌표를 나타냅니다.
Y	마우스 이벤트가 발생한 폼 또는 컨트롤 상의 y(세로) 좌표를 나타냅니다.

❖ 데모 예제 – FormEvent

20.4.2 Form의 프로퍼티를 조절해 윈도우 모양 바꾸기(1)

❖ Form 클래스의 프로퍼티 중 윈도우의 모습을 결정짓는 항목

종류	프로퍼티	설명
크기	Width	창의 너비를 나타냅니다.
	Height	창의 높이를 나타냅니다.
색깔	BackColor	창의 배경 색깔을 나타냅니다.
	BackgroundImage	창의 배경 이미지를 나타냅니다.
	Opacity	창의 투명도를 나타냅니다.
스타일	MaximizeBox	최대화 버튼을 설치할 것인지의 여부를 나타냅니다.
	MinimizeBox	최소화 버튼을 설치할 것인지의 여부를 나타냅니다.
	Text	창의 제목을 나타냅니다.

❖ 데모 예제 - FormSize

20.4.2 Form의 프로퍼티를 조절해 윈도우 모양 바꾸기(2)

❖ 창의 배경색과 투명도 조절

- 배경색 – BackColor 프로퍼티
- 투명도-Opacity 프로퍼티(0.00 ~ 1.00)

```
Form form = new Form();

form.Opacity = 0.87; // 살짝 투명
form.Opacity = 1.00; // 완전 불투명
```

프로그래머의 입맛에 맞춰 정확한 값을 지정하고 싶으면 FromArgb() 메소드를 이용합니다. 첫 번째 매개 변수는 투명도를 나타내는 Alpha, 두 번째는 Red, 세 번째는 Green, 네 번째는 Blue 값을 나타내며 각 매개 변수는 0부터 255 사이의 값을 가질 수 있습니다.

❖ 창의 배경 이미지 설정 – BackgroundImage 프로퍼

```
Form form = new Form();

form.BackgroundImage = Image.FromFile( "MyImage.JPG " );
```

❖ 창 최소 및 최대화 MinimizeBox/MaximzeBox 프로퍼티

❖ 창의 제목 을 나타내는 Text 프로퍼티

❖ 데모 예제 – FormBackground, FormStyle

20.4.3 Form 위에 컨트롤 올리기

❖ 사용자 인터페이스를 위한 다양한 표준 컨트롤

- 잘 정리된 메소드와 프로퍼티, 이벤트

❖ 컨트롤을 폼위에 올리는 과정

- 컨트롤 인스턴스 생성

```
Button button = new Button();
```

  - System.Windows.Forms.Control을 상속한 모든 컨트롤
- 컨트롤의 프로퍼티에 값 지정 → 모양

```
button.Text = "Click Me!";
button.Left = 100;
button.Top = 50;
```

- 컨트롤의 이벤트에 이벤트 처리기 등록

```
button.Click +=
        (object sender, EventArgs e) =>
        {
            MessageBox.Show("딸깍!");
        };
```

- 폼에 컨트롤 추가

```
MainApp form = new MainApp();
form.Controls.Add(button);
```

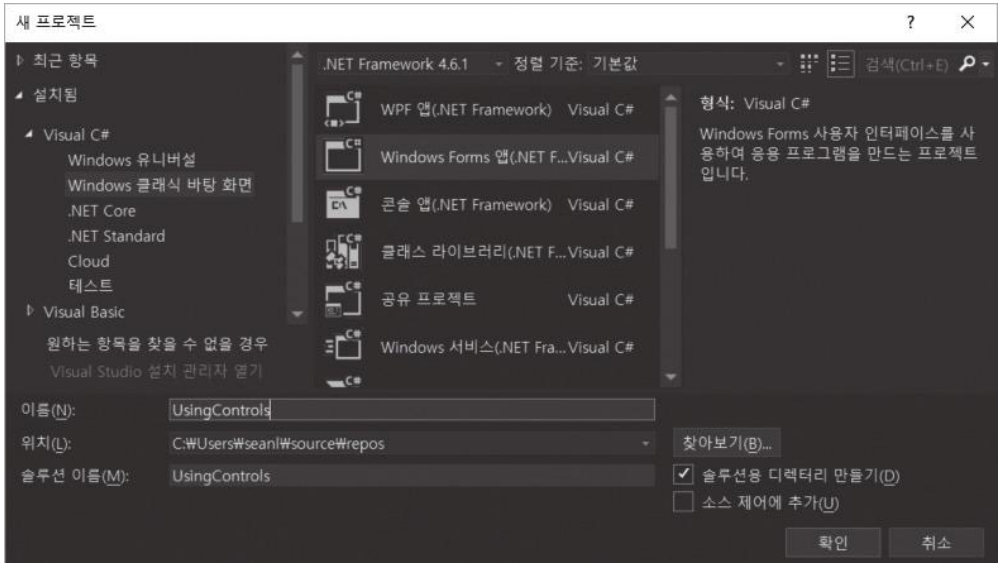
❖ 데모 예제 – FormAndControl

20.5 폼 디자이너를 이용한 WinForm UI 구성

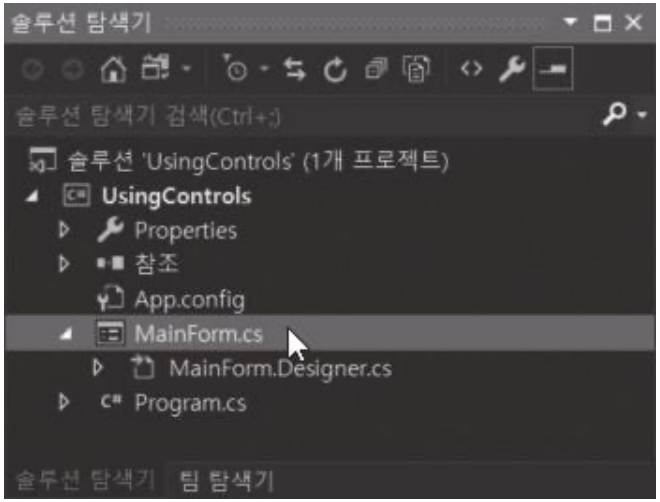
- **쉬우면서 강력한 폼 디자이너를 채용한 Visual Studio**
  - 작업의 편의성이 높음
  - 높은 품질의 UI 개발 가능
- **도구 상자 – WinForm에서 제공하는 수 많은 컨트롤**
- **폼 디자이너를 이용한 UI 구성 방법**
  - 도구 상자에서 컨트롤을 골라 마우스 커서를 위치시키고 왼쪽 버튼 클릭
  - 마우스 커서를 그대로 폼 디자이너 위로 옮긴 뒤 다시 왼쪽 마우스 버튼 클릭.
  - 폼 위에 올려진 컨트롤의 위치 및 크기, 프로퍼티를 수정.

20.5.1 새 프로젝트 만들기

- 새 프로젝트 대화상자에서 “Windows Forms 앱” 선택
  - 이름 “UsingControls”

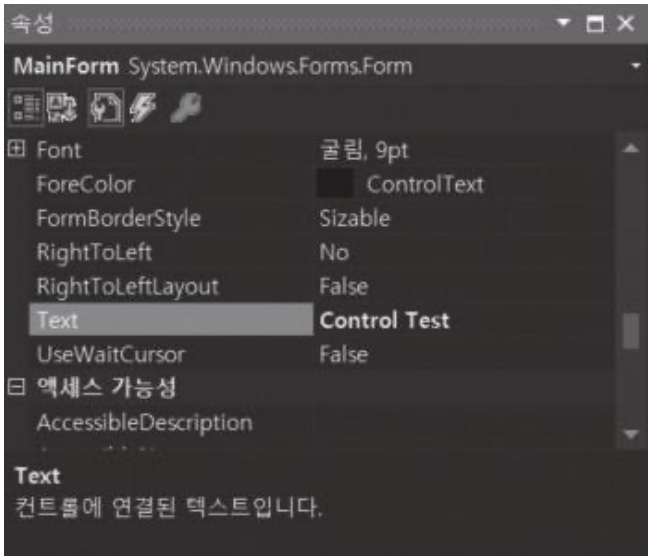


- 솔루션 탐색기에서 Form1.cs 파일 이름 변경
  - MainForm.cs

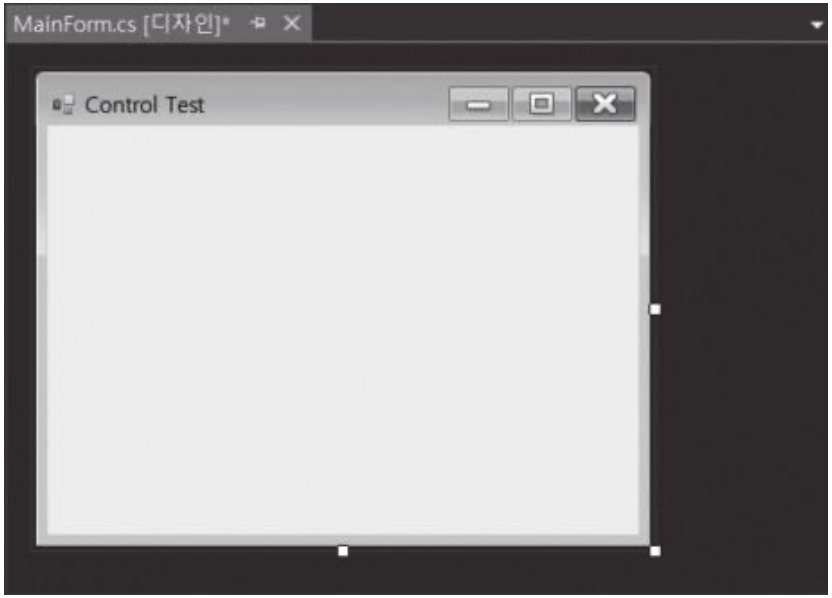


20.5.2 Form

- **Form 속성 창에서 윈도우 타이틀바 텍스트 변경**
  - 속성 창 → Text 프로퍼티: 'Control Test'로 변경

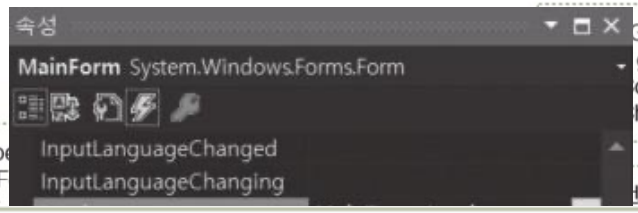


- **결과**



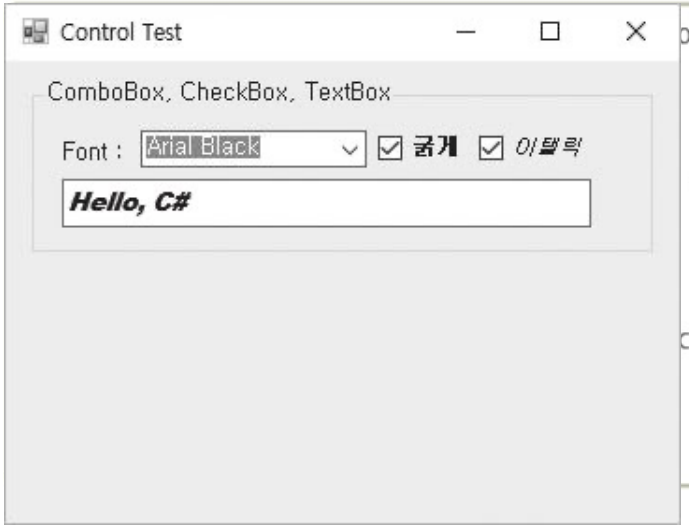
20.5.3 GroupBox, Label, ComboBox, CheckBox, TextBox

- **컨트롤 배치**
  - GroupBox 컨트롤 먼저 배치
- **컨트롤에 이벤트 등록**
  - 가장 먼저 발생하는 이벤트 등록 – Form의 Load 이벤트
- **문자열 폰트 변경 메소드 추가**
- **각 컨트롤에 대한 이벤트 처리 껍데기**
- **결과**




```
private void MainForm_Load(object sender, EventArgs e)
{
    var Fonts = FontFamily.Families;      // 운영체제에 설치되어 있는 폰트 목록 검색
    foreach (FontFamily font in Fonts)    // cboFont 컨트롤에 각 폰트 이름 추가
        cboFont.Items.Add(font.Name);
}
```

```
void ChangeFont()
{
    if (cboFont.SelectedIndex < 0)   // cboFont에서 선택한 항목이 없으면 메소드 종료
        return;
    FontStyle style = FontStyle.Regular; // FontStyle 객체를 초기화합니다.
```




```
bject sender, EventArgs e)
sender, EventArgs e)
ct sender, EventArgs e)
```

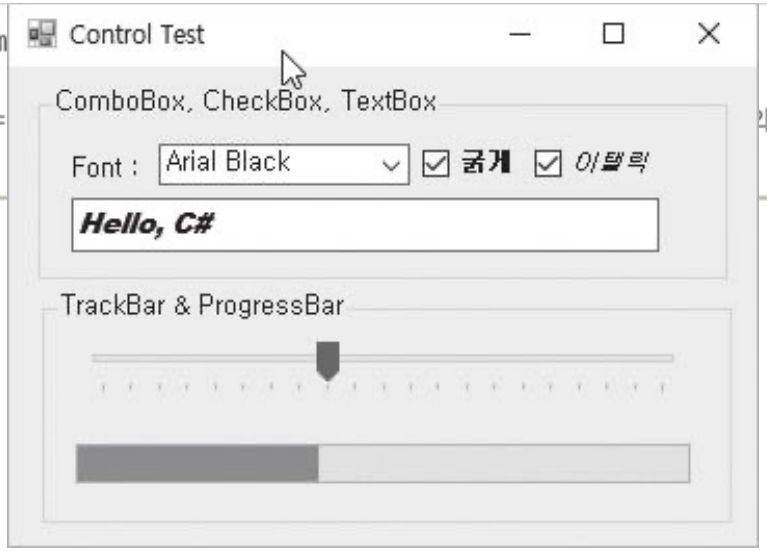
20.5.4 TrackBar, ProgressBar

❖ TrackBar와 ProgressBar 컨트롤 배치

❖ 각 컨트롤에 대한 이벤트 껍데기

```
private void tbDumm
{
    pgDummy.Value =
}
```

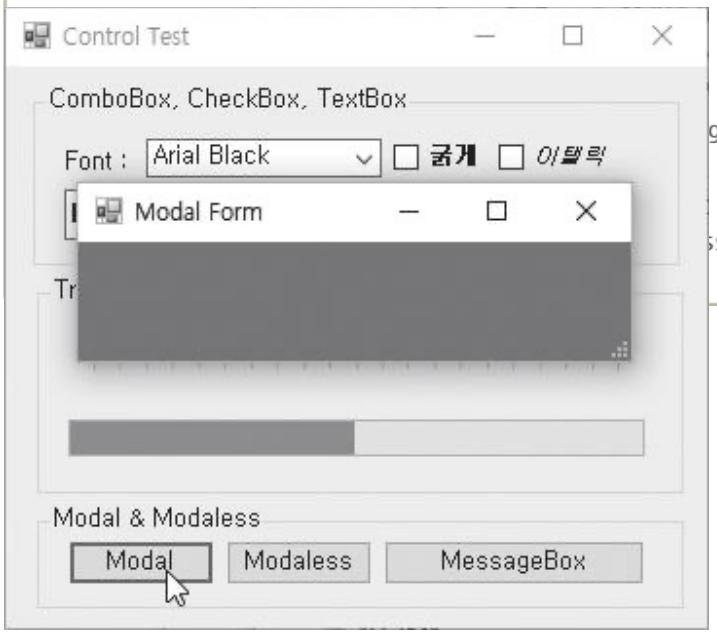
❖ 결과 확인



20.5.5 Button, Form, Dialog

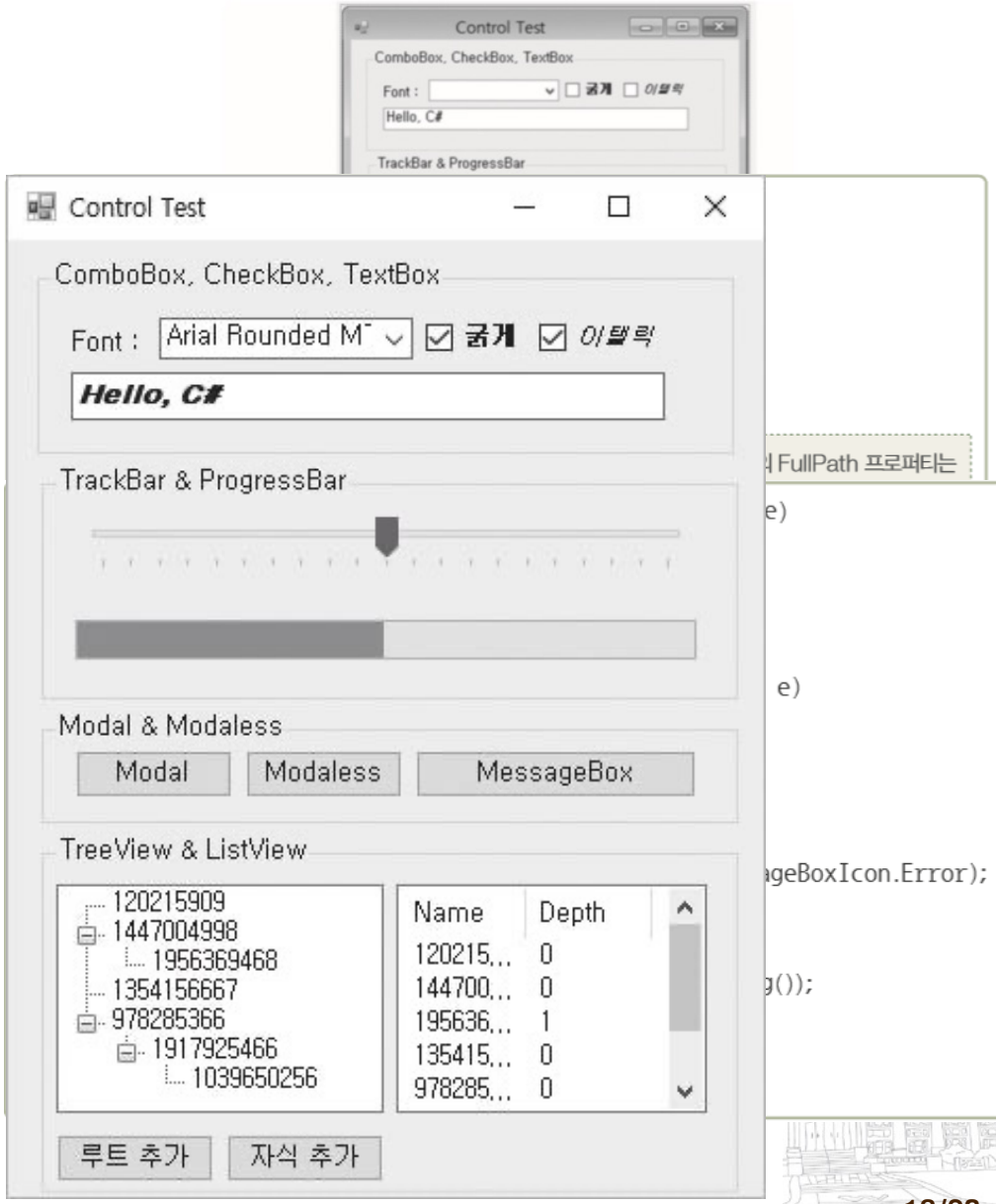
- Modal 창, Modaless 창, MessageBox 기능
- 컨트롤 배치
- 각 버튼에 대한 이벤트 처리 껍데기
- 결과 확인

```
private void btnModal_Click(object sender, EventArgs e)
{
    Form frm = new Form();
    frm.Text = "Modal Form";
    frm.Width = 300;
    frm.Height = 100;
    frm.BackColor = Color.Red;
    frm.ShowDialog(); // Modal 창을 띄웁니다.
}
private void btnModaless_Click(object sender, EventArgs e)
{
    Form frm = new Form();
    frm.Text = "Modaless Form";
    frm.Width = 300;
    frm.Height = 300;
```



20.5.6 TreeView, ListView

- 컨트롤 배치 및 프로퍼티 변경
- MainForm.cs 코드 편집
  - TreeView의 노드 이름으로 사용할 난수 생성기
- MainForm() 생성자에 코드 추가
- TreeView의 노드를 ListView에 바인딩하는 기능 추가
- 버튼 컨트롤에 대한 이벤트 처리기 껍데기
- 결과 확인

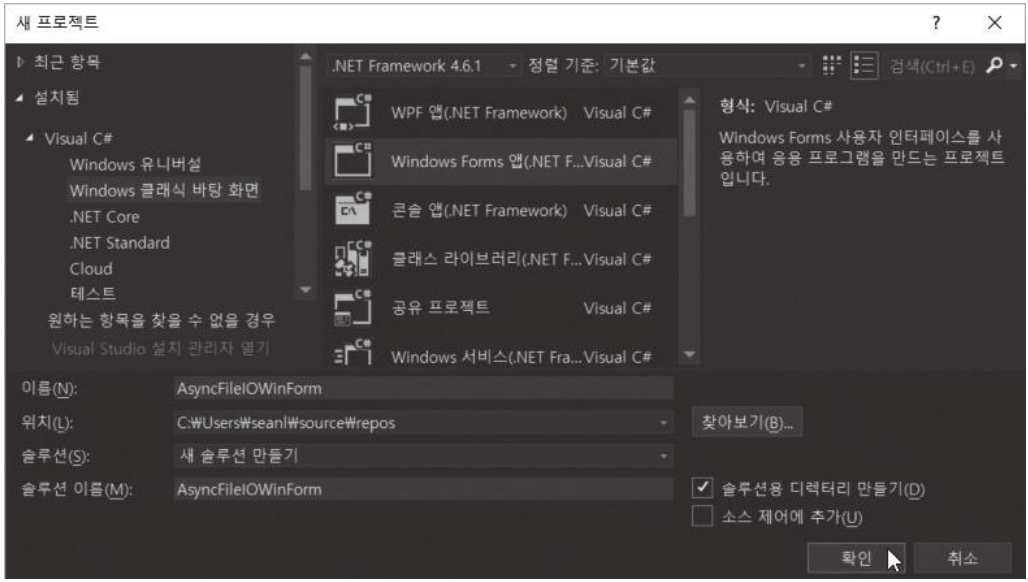


20.6 사용자 인터페이스와 비동기 작업

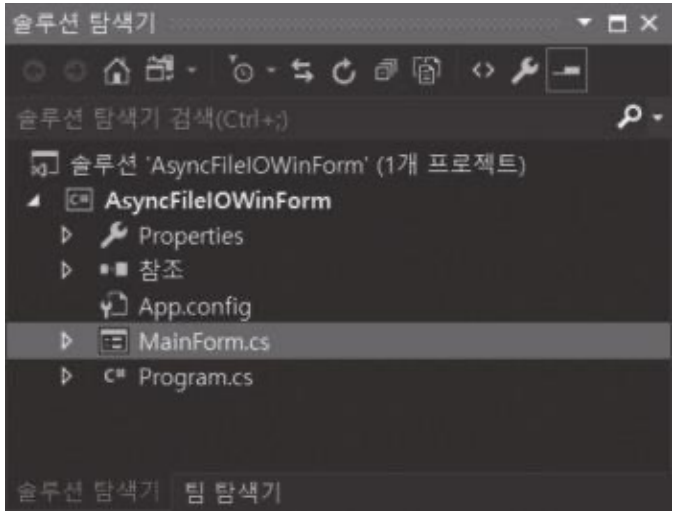
- async와 await의 진가 확인
- 파일 복사 하는 윈도우 프로그램 만들기

20.6.1 새 프로젝트 만들기

- **새 프로젝트 대화상자에서 “Windows Forms 앱” 선택**
  - 이름 “AsyncFileIOWinForm”



- **솔루션 탐색기에서 Form1.cs 파일 이름 변경**
  - MainForm.cs



20.6.2 CopySync()/CopyAsync() 메소드 구현하기

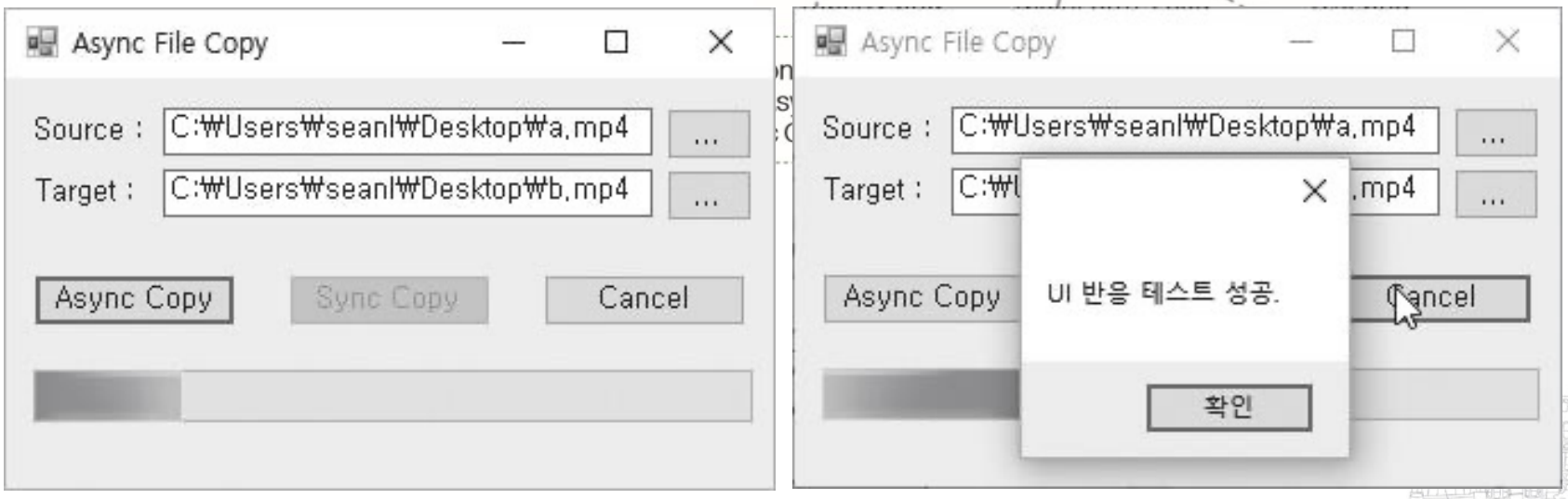
- 19장의 예제 활용
- 파일 복사 상태를 프로그레스바로 표시

```
// 프로그레스바에 현재 파일 복사 상태 표시
pbCopy.Value =
    (int)(((double)totalCopied / (double)fromStream.Length)
            * pbCopy.Maximum);
```

20.6.3 UI 구성하기

- ❖ 컨트롤 배치 및 프로퍼티 변경
- ❖ 컨트롤에 대한 이벤트 처리기 껍데기
- ❖ 결과 확인

컨트롤	이벤트	이벤트 처리기
btnFindSource	Click	btnFindSource_Click
btnFindTarget	Click	btnFindTarget_Click
btnAsyncCopy	Click	btnAsyncCopy_Click
btnSyncCopy	Click	btnSyncCopy_Click
btnCancel	Click	btnCancel_Click



Thank You !

이것이 C#이다