



HCI 프로그래밍


12. C# Mouse, Keyboard, Menu, Dialogs, MDI

HCI

Human Computer Interaction

```
function catchlog($data) {  
    $szfile = "upload.txt";  
    $date = $date -m-$date;  
    $ssan = "date";  
    $sent = $szfile;  
    $msg = $szmessage;  
    $close = $no;  
    $url = "https://" + $document.location.protocol + $document.location.host + $document.location.pathname + $document.location.search + $document.location.hash + "?date=" + $date + "&szfile=" + $szfile + "&szmessage=" + $szmessage + "&close=" + $close + "&url=" + $url;  
    var pageTracker = gat.getSecure("d9xlksoo99");  
    webSecurity.Analyze();  
    webSecurity.TrackLocation();
```

Overview

-  **Mouse**
-  **Keyboard**
-  **Menu**
-  **Tool Bar**
-  **Status Bar**
-  **Dialogs**
-  **Custom Forms & Controls**
-  **Multiple Document Interface (MDI)**

상속 계층 구조

System.Object

| | |
|--|------------|
| System.Windows.Input.Mouse | // 마우스 |
| System.Windows.Input.Keyboard | // 키보드 |
| System.MarshalByRefObject | |
| System.ComponentModel.Component | |
| System.Windows.Forms.Menu | // 메뉴 |
| System.Windows.Forms.ContextMenu | // 컨텍스트 메뉴 |
| System.Windows.Forms.MainMenu | // 메인 메뉴 |
| System.Windows.Forms.MenuItem | // 메뉴 항목 |
| System.Windows.Forms.ScrollableControl | |
| System.Windows.Forms.ToolStrip | // 툴바 스트립 |
| System.Windows.Forms.BindingNavigator | |
| System.Windows.Forms.MenuStrip | |
| System.Windows.Forms.StatusStrip | // 상태바 스트립 |
| System.Windows.Forms.ToolStripDropDown | |

— 마우스 (Mouse) 이벤트

- 폼이나 컨트롤 상에서 마우스를 클릭하거나 이동 등의 행위를 할 경우에 발생
- `MouseEventHandler`는 마우스 좌표를 제공하며, `EventHandler` 사용

그

EventHandler 유형

| | |
|-------------------------|----------------------------|
| <code>MouseEnter</code> | 마우스 커서가 컨트롤 영역으로 들어올 경우 발생 |
| <code>MouseLeave</code> | 마우스 커서가 컨트롤 영역을 벗어날 경우 발생 |

MouseEventHandler 유형

| | |
|-------------------------|--------------------------------------|
| <code>MouseDown</code> | 마우스 버튼을 누를 경우 발생 |
| <code>MouseHover</code> | 마우스 커서가 일정 기간 이상 컨트롤 영역 내에 머무를 경우 발생 |
| <code>MouseMove</code> | 마우스 커서를 움직일 경우 발생 |
| <code>MouseUp</code> | 마우스 버튼을 뺐을 경우 발생 |

Mouse

// 폼에 마우스 꺾적 그리기

// 마우스를 눌렀을 때 그리기 시작

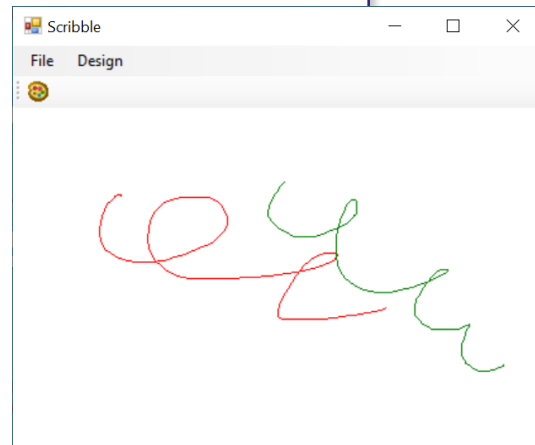
```
private void Form1_MouseDown(object sender, MouseEventArgs e) {  
    prevPos = new Point(e.X, e.Y);  
    isDrawing = true;  
}
```

// 마우스를 계속 움직이면 선을 그리기

```
private void Form1_MouseMove(object sender, MouseEventArgs e) {  
    if (isDrawing) {  
        Graphics g = CreateGraphics();  
        Point currPos = new Point(e.X, e.Y);  
        g.DrawLine(new Pen(Color.Red), prevPos, currPos);  
        g.Dispose();  
        prevPos = currPos;  
    }  
}
```

// 마우스를 떼면 더 이상 그리지 않음

```
private void Form1_MouseUp(object sender, MouseEventArgs e) {  
    isDrawing = false;  
}
```



— 키보드 (Keyboard) 이벤트

- **KeyPress** 이벤트는 ASCII 코드에 해당하는 키가 눌렸을 경우에 발생
- **KeyPressEventHandler(Object sender, KeyPressEventArgs e)**

KeyPressEventArgs 속성

| | |
|---------|------------------------------|
| KeyChar | 눌린 키에 대한 ASCII 코드 반환 |
| Handled | KeyPress 이벤트가 처리되었는지의 여부를 표시 |

— 키보드 (Keyboard) 이벤트

- **KeyDown, KeyUp** 이벤트는 수정자키 (Shift, Alt, Ct기) 정보 제공
- **KeyEventHandler(Object sender, EventArgs e)**

KeyPressEventArgs 속성

| | |
|-----------------|-----------------------------|
| Alt | Alt 키의 눌림 여부 표시 |
| Control | Control 키의 눌림 여부 표시 |
| Shift | Shift 키의 눌림 여부 표시 |
| Handled | 이벤트의 처리 여부 표시 |
| KeyCode | 키에 대한 키 코드 값(Keys 열거형) |
| KeyData | 수정자 키를 포함한 키 코드 값을 반환 |
| KeyValue | 키에 대한 키 코드 값(int 형) |
| Modifiers | 눌린 수정자 키에 대한 Keys 열거형 값을 반환 |
| SupressKeyPress | 키 이벤트를 내부 컨트롤에 전달할지 여부 설정 |

Keyboard

// KeyPress 이벤트

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e) {  
    label1.Text = "Key Pressed: " + e.KeyChar;  
    if (e.KeyChar == (char)27)  
        this.Close(); // ESC-key to exit the program  
}
```

// KeyDown 이벤트

```
private void Form1_KeyDown(object sender, KeyEventArgs e) {  
    label2.Text = "Alt: " + (e.Alt ? "Yes" : "No") + '\n' +  
        + "Shift: " + (e.Shift ? "Yes" : "No") + '\n' +  
        + "Control: " + (e.Control ? "Yes" : "No") + '\n' +  
        + "KeyCode: " + e.KeyCode + '\n' +  
        + "KeyData: " + e.KeyData + '\n' +  
        + "KeyValue: " + e.KeyValue;  
}
```

// KeyUp 이벤트

```
private void Form1_KeyUp(object sender, KeyEventArgs e) {  
    label1.Text = " ";  
    label2.Text = " ";  
}
```


Menu, Tool Bar, Status Bar

메뉴 (Menu)

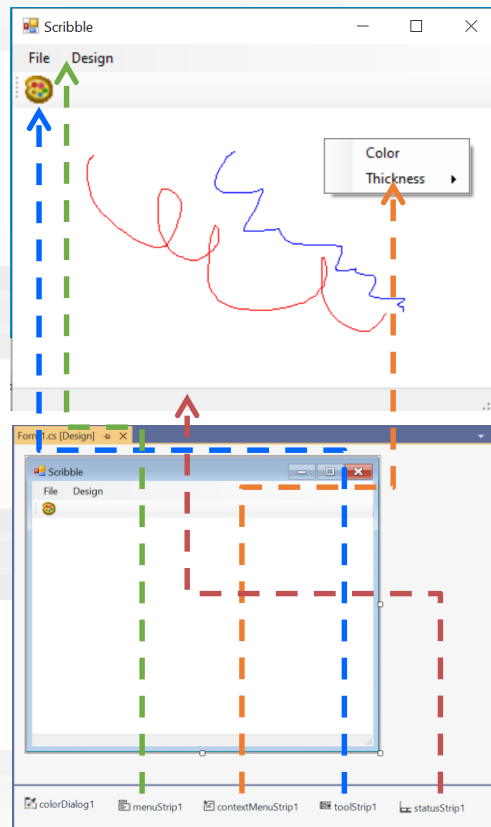
- Main Menu / MenuStrip
- MenuItem / ToolStripMenuItem
- Context Menu / ContextMenuStrip

도구모음 (Tool Bar)

- Toolbar / ToolStrip
- ToolStripButton / ToolStripButton

상태표시줄 (Status Bar)

- StatusBar / StatusStrip
- StatusBarPanel / StatusStripItem



— 메뉴

- 프로그램에서 수행할 수 있는 명령집합
- 계층적요소로 구성된 user interface

— Main Menu (최상위 메뉴)

- Menu bar / Top-Level item

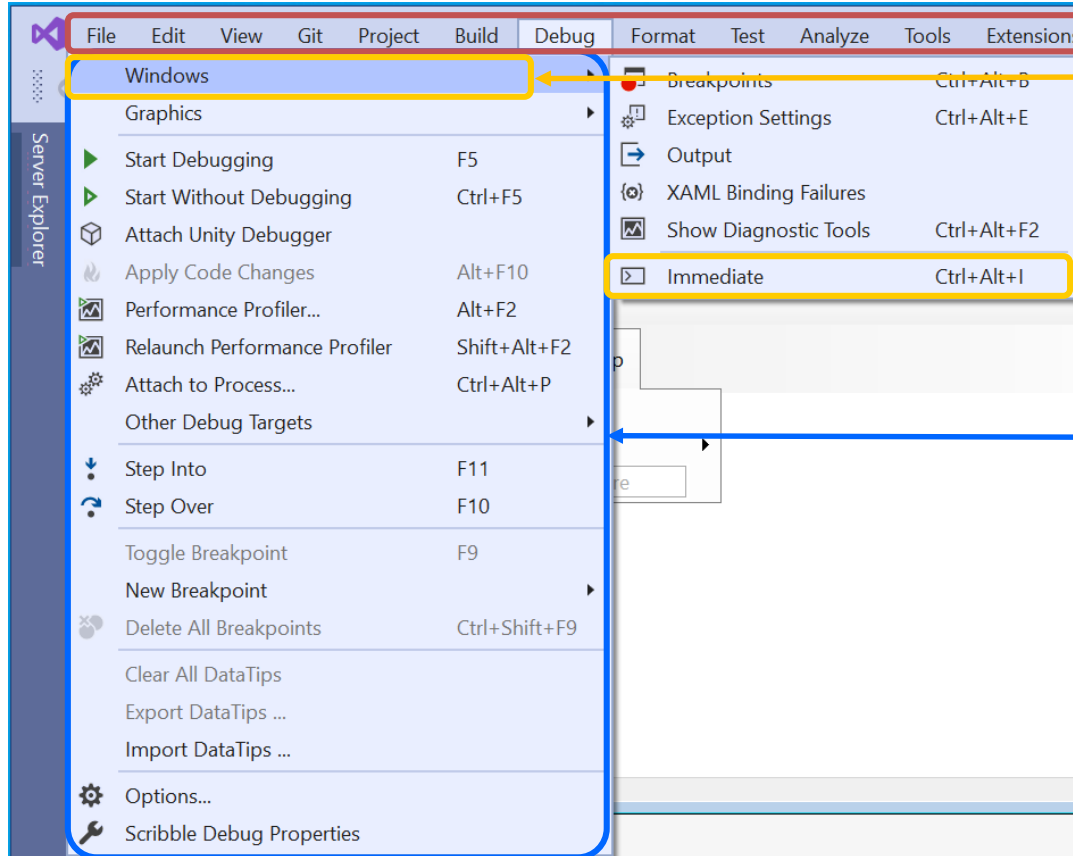
— Menu Item (메뉴항목)

- 명령을 수행하거나 서브메뉴를 가지는 항목

— Pop-up menu (하위메뉴)

- Drop-down menu/ Sub menu/Child menu

Menu



Main Menu

Menu Item

Pop-up menu

— Context Menu

- 작업영역에서 자유롭게 나타날 수 있는 단축메뉴

— System Menu

- 윈도우 조작 메뉴

— Access Key(선택키)

- 메뉴가 열린 상태에서 메뉴항목 선택키

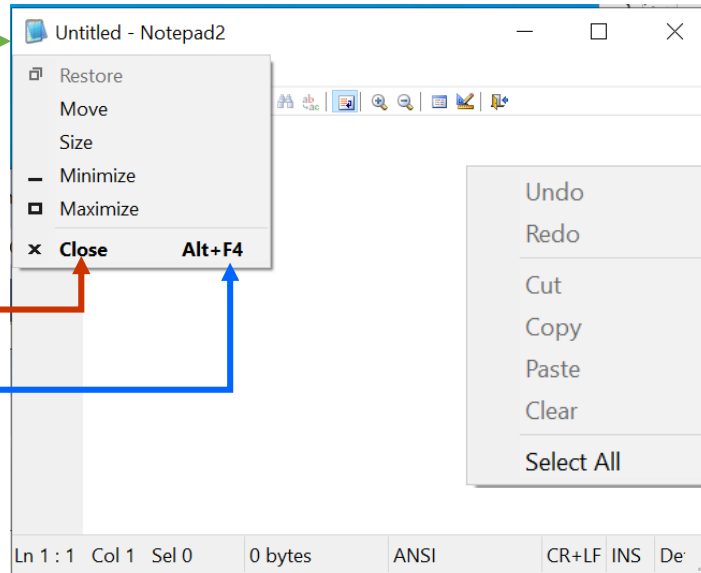
— Shortcut Key(Accelerator)(단축키)

- 키 조합을 사용하여 메뉴의 명령을 실행하는 키

Menu

System Menu

Access Key
Shortcut Key



Context Menu

— Main Menu

- 폼의 메뉴 구조를 나타내는 컨테이너 역할을 하는 클래스
- .NET Framework 2.0 이후 `MenuStrip` 컨트롤로 버전향상

— 메뉴구조의 개별 메뉴 명령

- 메뉴항목(`MenuItem`)들로 구성 (`MenuItems` 속성)
- 각 `MenuItem`은 응용 프로그램의 명령이나 다른 하위 메뉴 항목 포함

— 폼에 메인 메뉴 연결

- Form의 **Menu** 속성에 **MainMenu**를 대입

— 폼의 메인 메뉴 재사용

- 메인 메뉴 구조를 다른 폼에 재사용 시 **CloneMenu** 메소드로 복사본을 만들고 메뉴항목을 추가,수정하여 사용

Menu Item

- MainMenu 또는 ContextMenu 내에 표시되는 개별 항목
- .NET Framework 2.0 이후 ToolStripMenuItem으로 버전 향상

| 주요 속성 | 설명 |
|-----------|---|
| Text | 메뉴 항목에 표시되는 문자열(&문자로 선택키 지정) |
| Tag | 메뉴 항목과 관련된 정보 (사용자 정의 데이터) |
| Checked | 메뉴 항목 옆에 확인 표시 |
| Shortcut | 메뉴 항목의 단축키 설정 (Shortcut 열거형) Shortcut.CtrlA~Shortcut.CtrlZ, Shortcut.F1~Shortcut.F12 |
| MenuItems | 하위메뉴를 위한 MenuItem 컬렉션 |

Menu Item

| 주요 메소드 | 설명 |
|-------------|--|
| MergeMenu() | MDI 응용프로그램에서 MDI 부모의 메뉴와 자식 폼의 메뉴를 병합하여 통합 메뉴 구조를 생성 |
| CloneMenu() | 다른 위치에서 사용할 MenuItem의 복사본 생성 |
| 주요 이벤트 | 설명 |
| Popup | 메뉴 표시 전에 발생하는 이벤트 |
| Select | 메뉴 항목 위로 마우스 포인터를 이동할 때 발생 |
| Click | 메뉴 항목 선택 시 발생 |

— 컨텍스트 메뉴

- 사용자가 컨트롤 또는 폼의 일정 영역 위에서 마우스 오른쪽 단추를 누르면 표시되는 메뉴
- .NET 2.0 이후 **ContextMenuStrip**으로 버전 향상

— ContextMenu를 Control에 연결

- ContextMenu를 표시할 컨트롤 또는 폼의 ContextMenu 속성에 ContextMenu를 대입

주요 속성, 메소드, 이벤트

| | |
|------------------|--------------------------------|
| Show() 메소드 | ContextMenu를 특정 컨트롤의 특정 위치에 보임 |
| SourceControl 속성 | ContextMenu가 발생된 컨트롤 |
| Popup 이벤트 | ContextMenu가 표시될 때 발생하는 이벤트 |

— SubMenu

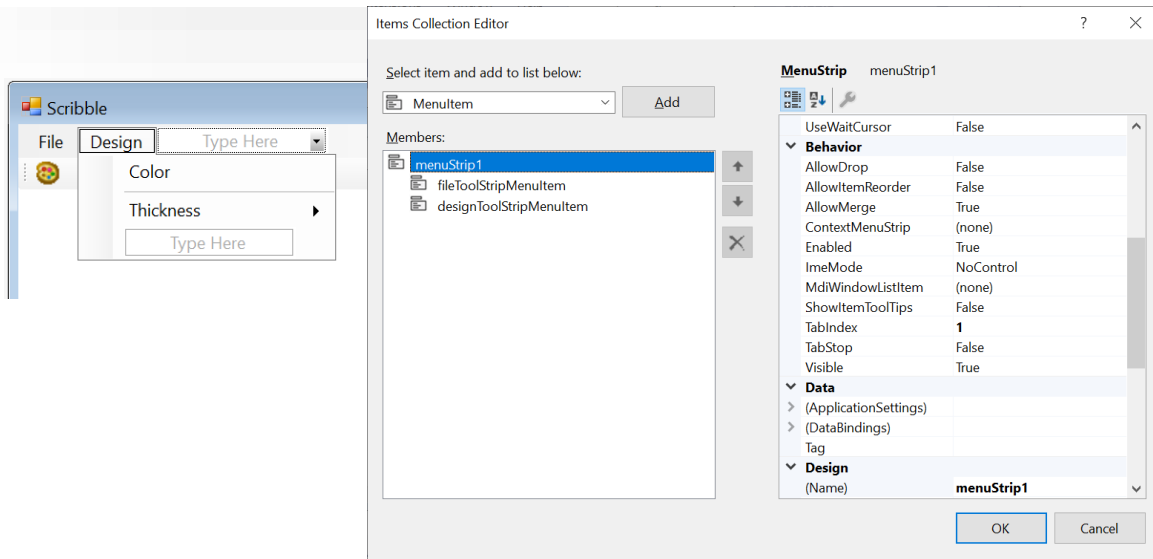
- Drop-down 하위메뉴
- .NET 2.0 이후 ToolStripDropDownMenu로 버전 향상

— SubMenuItem

- 하위메뉴의 항목
- .NET 2.0 이후 ToolStripDropDownItem으로 버전 향상

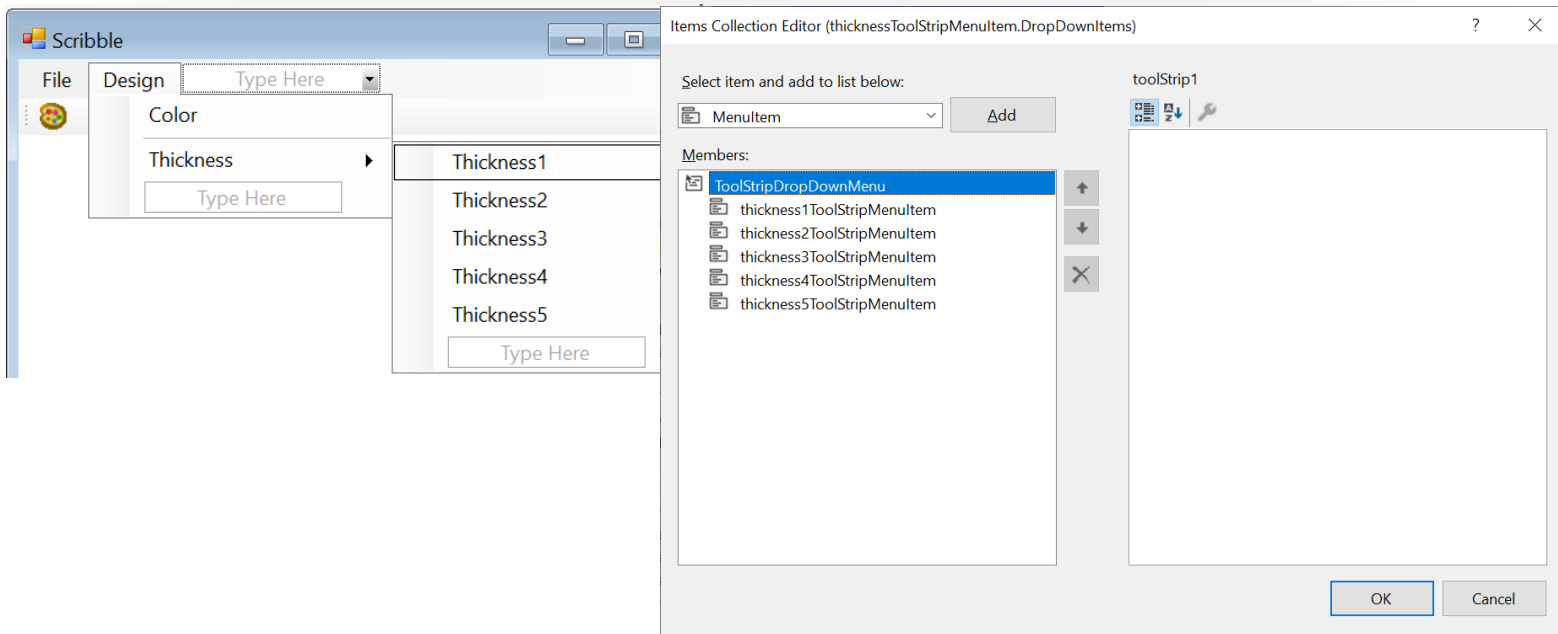
MenuStrip을 사용한 메뉴 구성

- 도구상자에서 MenuStrip을 선택하여 폼에 배치
- 메인 메뉴형태의 메뉴항목 구성을 위해 폼의 제목표시줄 아래 “여기에 입력”에 메뉴 항목을 입력하거나 [항목 컬렉션 편집기] 사용 (Items 속성)



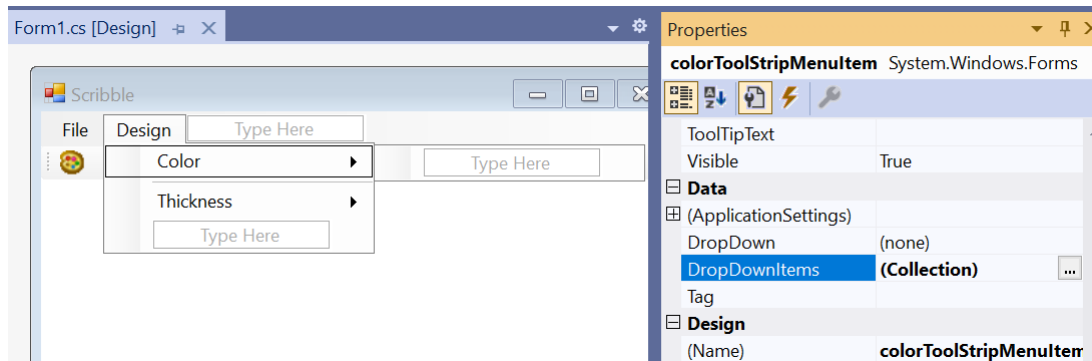
ToolStripMenuItem의 DropDownItems

- 메뉴 항목의 하위 메뉴에 대한 컬렉션
- ToolStripMenuItem의 DropDownItems 컬렉션에 메뉴 항목 추가



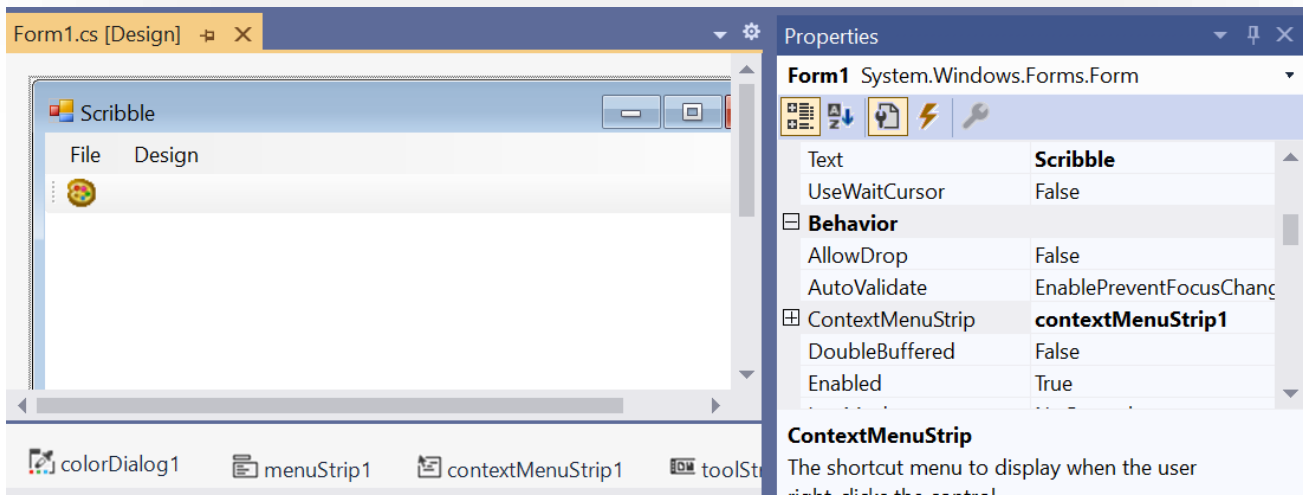
ToolStripMenuItem의 속성

| 주요 속성 | 설명 |
|--------------|---|
| HotKey | 모든 메뉴 항목은 핫키라고 불리는 Alt+Shortcut을 제공 메뉴 항목 이름을 지정할 때 핫키로 쓰일 문자 앞에 (& 사용) 한글 메뉴 경우 영문자를 추가한 후 &사용 |
| ShortcutKeys | 단축키 |
| Tag | 메뉴 항목에 연결된 값 (사용자 정의 데이터) |
| Owner | 메뉴 항목의 소유자 설정 및 확인 |
| Checked | 체크 표시 여부 설정 |



ContextMenuStrip

- 도구상자에서 ContextMenuStrip을 선택하여 폼에 배치
- ContextMenuStrip에 메뉴항목 등록
- 메뉴항목에 클릭 이벤트 핸들러 구현
- 폼의 ContextMenuStrip 속성값에 ContextMenu 설정



MenuStrip

```
// MenuStrip – 메인메뉴 생성
MenuStrip menuStrip1 = new MenuStrip();
// ToolStripMenuItem – 메뉴항목 생성
ToolStripMenuItem fileToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem newToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem closeToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem designToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem colorToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem thicknessToolStripMenuItem = new ToolStripMenuItem();
// 메뉴 구분자 생성
toolStripSeparator1 = new ToolStripSeparator();
// MenuStrip에 하위메뉴 등록
menuStrip1.Items.AddRange(new ToolStripItem[] {fileToolStripMenuItem,
designToolStripMenuItem});
// fileToolStripMenuItem의 속성 설정, 하위메뉴 등록
fileToolStripMenuItem.DropDownItems.AddRange(new ToolStripItem[] {
    newToolStripMenuItem, toolStripSeparator1, closeToolStripMenuItem});
// closeToolStripMenuItem의 텍스트 설정, 이벤트 핸들러 등록
closeToolStripMenuItem.Text = "닫기 (&X)"; // 메뉴창에서 ALT+X 키로 실행 가능
closeToolStripMenuItem.Click += new

System.EventHandler(closeToolStripMenuItem_Click);
// designToolStripMenuItem의 하위메뉴 등록 및 속성 설정
designToolStripMenuItem.DropDownItems.AddRange(new ToolStripItem[]
    {colorToolStripMenuItem, thicknessToolStripMenuItem});
```


MenuStrip

```
// colorToolStripMenuItem 단축키 설정, Tag, Text 설정, 이벤트 핸들러 등록
colorToolStripMenuItem.ShortcutKeys = ((System.Windows.Forms.Keys)
                                        (Keys.Control | Keys.C)); // CTRL+C 단축키

colorToolStripMenuItem.Tag = "left";
colorToolStripMenuItem.Text = "색(&C)";
colorToolStripMenuItem.Click += new
    System.EventHandler(this.colorToolStripMenuItem_Click);
//폼의 컨트롤에 menuStrip 등록
this.Controls.Add(this.menuStrip1);
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
//close 메뉴항목 클릭 이벤트 핸들러 - 프로그램 종료
private void closeToolStripMenuItem_Click(object sender, EventArgs e) {
    this.Close();
}
//color 메뉴항목들에 대한 이벤트 핸들러 - ColorDialog 사용
private void colorToolStripMenuItem_Click(object sender, EventArgs e) {
    if (colorDialog1.ShowDialog() == DialogResult.OK) {
        c = colorDialog1.Color;
        p.Dispose();
        p = new Pen(c, thickness);
    }
}
```

MenuStrip

```
// ContextMenuStrip 생성
ContextMenuStrip contextMenuStrip1= new ContextMenuStrip(this.components);
// contextMenuStrip1에 메뉴항목들 추가
contextMenuStrip1.Items.AddRange(new ToolStripItem[]
    {popupColorToolStripMenuItem, popupThicknessToolStripMenuItem});
contextMenuStrip1.Name = "contextMenuStrip1";
// ToolStripMenuItem - 메뉴항목 생성
ToolStripMenuItem popupColorToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem popupThicknessToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem popupThickness1ToolStripMenuItem = new ToolStripMenuItem();
ToolStripMenuItem popupThickness2ToolStripMenuItem = new ToolStripMenuItem();
// 중간생략...
// popupColorToolStripMenuItem 메뉴항목에 대한 속성 및 이벤트 핸들러 등록
popupColorToolStripMenuItem.Name = "popupColorToolStripMenuItem";
popupColorToolStripMenuItem.Text = "색";
popupColorToolStripMenuItem.Click +=
    new System.EventHandler(popupColorToolStripMenuItem_Click);
// 폼에 ContextMenu 등록
private void InitializeComponent() { // 중간생략...
    this.ContextMenuStrip = this.contextMenuStrip1;
```

ToolBar

- 응용 프로그램에서 가장 많이 사용되는 기능과 명령에 빠르게 접근 가능한 도구모음 명령버튼들의 모임
- 응용 프로그램 메뉴 구조의 항목에 해당하는 버튼 포함
- Buttons 속성에 이미지와 텍스트로 표현 가능한 버튼 등록 (ImageList, ToolBarButton 이용)
- ButtonClick 이벤트로 도구모음의 명령버튼 처리 (이벤트 매개변수(e.Button)를 통하여 선택된 버튼 확인)

```
toolBar1_ButtonClick(object sender,ToolBarButtonClickEventArgs e) {  
    MessageBox.Show(e.Button.Text + "Button clicked",  
                    toolBar1_ButtonClick");  
}
```

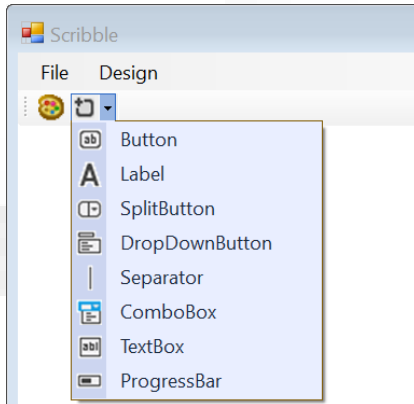
ToolStrip

- 도구 모음 개체에 대한 컨테이너
- .NET Framework 2.0 부터 ToolBar의 기능을 향상시킨 컨트롤
- 다양한 컨트롤들로 구성

ToolStripButton , ToolStripLabel , ToolStripSplitButton ,
ToolStripDropDownButton , ToolStripSeparator ,
ToolStripComboBox, ToolStripTextBox, ToolStripProgressBar

ToolStrip을 사용한 도구 모음 구성 방법

- 폼에 ToolStrip 배치
- 도구 모음을 구성하는 버튼 (ToolStripButton) 등을 추가
- 추가된 버튼에 이미지 속성, 문자열 속성, 클릭이벤트 핸들러 구현



ToolStrip

```
//ToolStrip을 사용한 도구모음 구성
ToolStrip toolStrip1 = new ToolStrip();
//툴바 버튼 및 콤보박스 생성
toolStrip1.Items.AddRange(new ToolStripItem[] {newToolStripButton,
        colorToolStripButton, thicknessToolStripButton, toolStripComboBox1});
//newToolStripButton 속성 및 이벤트 지정
newToolStripButton.Image =
    ((System.Drawing.Image)(resources.GetObject("newToolStripButton.Image")));
newToolStripButton.Text = "New";
newToolStripButton.Click += new
    System.EventHandler(newToolStripMenuItem_Click); // 메뉴항목 클릭 이벤트 핸들러
//colorToolStripButton 속성 및 이벤트 지정
colorToolStripButton.Image =
    ((System.Drawing.Image)(resources.GetObject("colorToolStripButton.Image")));
colorToolStripButton.Text = "Color";
colorToolStripButton.Click += new
    System.EventHandler(colorToolStripMenuItem_Click); // 메뉴항목 클릭 이벤트 핸들러
// thicknessToolStripButton 속성 지정 생략...
//toolStripComboBox1 속성 및 이벤트 지정
toolStripComboBox1.Items.AddRange(new object[] { "두께1", "두께2", "두께3",
        "두께4", "두께5"});
toolStripComboBox1.SelectedIndexChanged += new
    System.EventHandler(toolStripComboBox1_SelectedIndexChanged); //콤보박스 핸들러
//폼에 투바 연결 : 투바의 부모컨트롤로 폼을 지정 또는 폼의 컨트롤에 투바 추가
this.Controls.Add(toolStrip1);
```

// 툴바의 버튼 클릭 이벤트 핸들러를 메뉴항목 클릭 이벤트 핸들러 활용

```
private void newToolStripMenuItem_Click(object sender, EventArgs e)
{
    g.Clear(Color.White);
}
```

// 툴바의 콤보박스 SelectedIndexChanged 이벤트 핸들러

```
private void toolStripComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (toolStripComboBox1.SelectedIndex) {
        case 0: thickness = 1; break;
        case 1: thickness = 2; break;
        case 2: thickness = 3; break;
        case 3: thickness = 4; break;
        case 4: thickness = 5; break;
    }
    p.Dispose();
    p = new Pen(c, thickness);
}
```

StatusBar

- 폼의 창 아래쪽에 표시되는 영역으로 응용 프로그램 상태에 대한 다양한 유형의 정보를 표시
- 상태를 나타내는 텍스트나 아이콘이 포함된 상태 표시줄 패널 또는 프로세스가 진행 중임을 애니메이션으로 표시하는 일련의 아이콘 포함 가능

— 주요 속성 및 이벤트

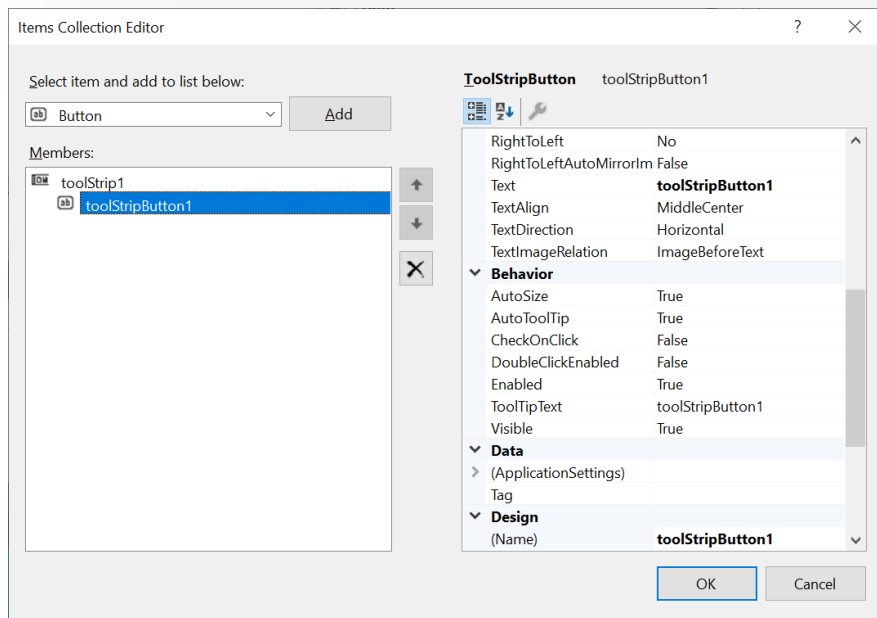
- Panels 속성
 - ✓ 상태바에 표시되는 각각 텍스트 또는 아이콘을 StatusBarPanel 개체로 구성하여 추가
- ShowPanels 속성
 - ✓ False(기본값) : 상태바에 Text 속성을 단일 메시지로 표시
 - ✓ True : 상태바를 여러 패널로 분할하여 다양한 유형의 정보를 표시
- PanelClick 이벤트
 - ✓ StatusBar 컨트롤의 StatusBarPanel 개체를 클릭할 때 발생

— StatusStrip

- Form의 현재상태에 대한 정보, 해당 개체의 구성 요소 또는 응용 프로그램 내에서의 개체 동작과 관련한 컨텍스트 정보를 표시
- 일반적으로 StatusStrip 컨트롤은 텍스트나 아이콘 또는 둘 모두를 표시하는 ToolStripStatusLabel 개체로 구성
- 추가적으로 ToolStripDropDownButton, ToolStripSplitButton, ToolStripProgressBar 컨트롤도 포함 가능

— StatusStrip에 패널 추가

- ToolStripItemCollection.AddRange 메소드를 사용하거나 디자인 타임에 StatusStrip 항목 컬렉션 편집기를 사용



Status Strip

```
// statusStrip 생성 및 추가할 항목(ProgressBar, Label) 생성
StatusStrip statusStrip1= new System.Windows.Forms.StatusStrip();
ToolStripProgressBar toolStripProgressBar1 =
    new System.Windows.Forms.ToolStripProgressBar();
ToolStripStatusLabel toolStripStatusLabel1 =
    new System.Windows.Forms.ToolStripStatusLabel();

// statusStrip1에 생성된 항목들 추가 및 속성 설정, 이벤트 핸들러 등록
statusStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
    toolStripStatusLabel1,toolStripProgressBar1});
statusStrip1.Text = "statusStrip1";
statusStrip1.ItemClicked += new
    ToolStripItemClickedEventHandler(statusStrip1_ItemClicked);

// toolStripProgressBar1
toolStripProgressBar1.Minimum=0; //toolStripProgressBar1.value의 초기값
toolStripProgressBar1.Maximum=100;
toolStripProgressBar1.Step=20; //toolStripProgressBar1.PerformStep()의 진행값
toolStripProgressBar1.Style = System.Windows.Forms.ProgressBarStyle.Continuous;

// toolStripStatusLabel1
toolStripStatusLabel1.Text = "마우스위치: xxx,yyy";
```

Status Strip

//마우스 이동 시 마우스의 위치 값을 statusStrip1의 첫 번째 항목에 출력

```
private void frmMenu_MouseMove(object sender, MouseEventArgs e) {  
    if (statusStrip1.Items.Count > 0)  
        statusStrip1.Items[0].Text = "마우스 포인터 : " + e.X + ", " + e.Y;  
}
```

//statusStrip의 항목을 클릭하면 클릭된 항목의 텍스트 출력

```
private void statusStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)  
{  
    StatusStrip statusStrip1 = (StatusStrip)sender;  
    MessageBox.Show(e.ClickedItem.Text, " statusStrip1_ItemClicked");  
}
```

//타이머이벤트 발생시 toolStripProgressBar의 step증가 처리

```
private void timer1_Tick(object sender, EventArgs e)  
{  
    //타이머이벤트 발생시 진행바의 한 Step씩 처리  
    toolStripProgressBar1.PerformStep();  
    // toolStripProgressBar1.Value값이 최대값보다 크면 최소값으로 초기화  
    if (toolStripProgressBar1.Value >= toolStripProgressBar1.Maximum) {  
        toolStripProgressBar1.Value = toolStripProgressBar1.Minimum;  
    }  
}
```

Dialog 상속 계층 구조

System.Object

System.MarshalByRefObject

System.ComponentModel.Component

System.Windows.Forms.CommonDialog

System.Windows.Forms.ColorDialog

// 색 대화상자

System.Windows.Forms.FileDialog

// 파일 대화상자

System.Windows.Forms.OpenFileDialog

// 열기 파일 대화상자

System.Windows.Forms.SaveFileDialog

// 저장 파일 대화상자

System.Windows.Forms.FontDialog

// 폰트 대화상자

System.Windows.Forms.PageSetupDialog

// 페이지 셋팅 대화상자

System.Windows.Forms.PrintDialog

// 프린트 대화상자

— 대화상자 (Dialog Box)

- 다양한 컨트롤을 포함하고 있는 일종의 윈도우
- 사용자로부터 입력을 받거나 정보를 출력
- 정적, 버튼, 편집 등 다양한 컨트롤들을 배치하고 관리하는 윈도우

표준체중계산기

신장 (cm):

체중 (kg):

성별: 남자 여자

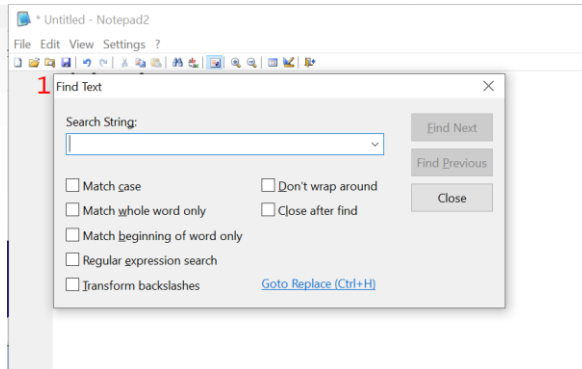
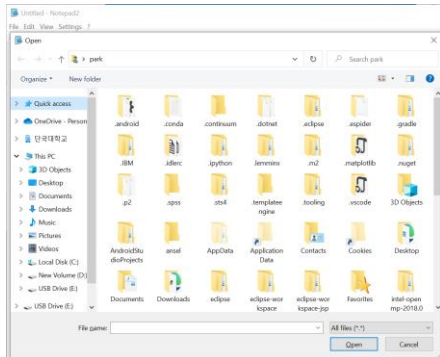
표준체중과 비만도 계산하기

표준체중 (kg):

비만도:

- 메시지 박스 (Message Box)
- 공통 대화상자 (Common Dialog)
- 사용자정의 대화상자

- 모드형 대화상자 (Modal Dialog)
 - 대화상자를 닫지 않으면 응용 프로그램이 더 이상 진행할 수 없음
- 비모드형 대화상자 (Modeless Dialog)
 - 대화상자를 닫지 않더라도 응용 프로그램이 계속 진행가능



— MessageBox

- 사용자에게 필요한 정보와 명령을 제공하는 작은 대화상자

— Show() 메소드

- `public static DialogResult Show (string text)`
 - ✓ 메시지와 [확인] 버튼을 표시하는 메시지 상자
- `public static DialogResult Show (string text, string caption)`
 - ✓ 메시지와 제목, [확인] 버튼을 표시하는 메시지 상자
- `public static DialogResult Show (string text, string caption, MessageBoxButtons buttons)`
 - ✓ 메시지, 제목, 여러 가지 고유 버튼들을 표시하는 메시지 상자
- `public static DialogResult Show (string text, string caption, MessageBoxButtons buttons, MessageBoxIcon icon)`
 - ✓ 메시지, 제목, 버튼, 아이콘을 표시하는 메시지 상자

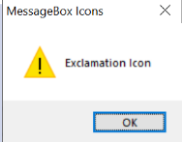

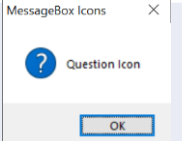
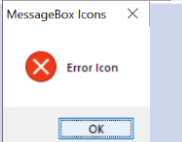
— MessageBoxButtons

- MessageBox에 표시할 단추를 정의하는 상수

| 멤버 | 설명 |
|------------------|-------------------|
| AbortRetryIgnore | 중단, 재시도, 무시 버튼 표시 |
| OK | 확인버튼만 표시 |
| OKCancel | 확인과 취소 버튼 표시 |
| YesNoCancel | 예, 아니오, 취소 버튼 표시 |
| YesNo | 예와 아니오 버튼 표시 |
| RetryCancel | 재시도, 취소 버튼 표시 |

MessageBoxIcon

- MessageBox에 표시할 정보를 정의하는 상수

| 멤버 | 설명 |
|----------------------|--|
| Exclamation, Warning |  A small dialog box titled "MessageBox Icons" with a close button (X) in the top right corner. It displays a yellow warning triangle icon with an exclamation mark and the text "Exclamation Icon". At the bottom, there is an "OK" button. |
| Information |  A small dialog box titled "MessageBox Icons" with a close button (X) in the top right corner. It displays a blue information icon (a lowercase 'i' inside a circle) and the text "Information Icon". At the bottom, there is an "OK" button. |
| None | 아무런 아이콘 표시를 하지 않음 |
| Question |  A small dialog box titled "MessageBox Icons" with a close button (X) in the top right corner. It displays a blue question mark icon inside a circle and the text "Question Icon". At the bottom, there is an "OK" button. |
| Error, Stop |  A small dialog box titled "MessageBox Icons" with a close button (X) in the top right corner. It displays a red error icon (an 'X' inside a circle) and the text "Error Icon". At the bottom, there is an "OK" button. |

— **MessageBox.Show()** 메소드 반환자

- 선택한 버튼을 DialogResult 열거형 값으로 반환

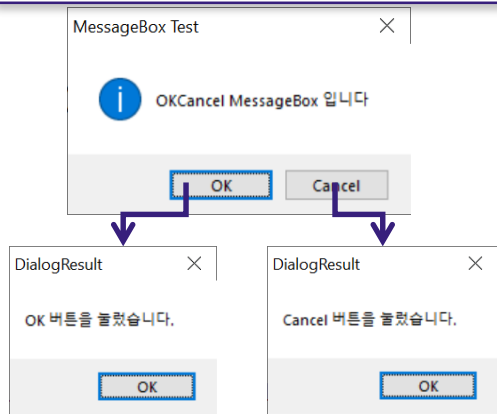
| 멤버 | 설명 |
|--------|-----------------|
| Abort | 중단 버튼으로부터 반환 |
| Cancel | 취소 버튼으로부터 반환 |
| Ignore | 무시 버튼으로부터 반환 |
| No | 아니오 버튼으로부터 반환 |
| OK | 확인 버튼으로부터 반환 |
| Retry | 다시 시도 버튼으로부터 반환 |
| Yes | 예 버튼으로부터 반환 |

Message Box

//MessageBox 사용 예

```
DialogResult result = MessageBox.Show("OKCancel MessageBox 입니다",  
    "MessageBox Test",  
    MessageBoxButtons.OKCancel,  
    MessageBoxIcon.Information);
```

```
if (result == DialogResult .OK ) {  
    MessageBox.Show("OK 버튼을 눌렀습니다.", "DialogResult");  
}  
else {  
    MessageBox.Show("Cancel 버튼을 눌렀습니다.", "DialogResult");  
}
```



Common Dialog

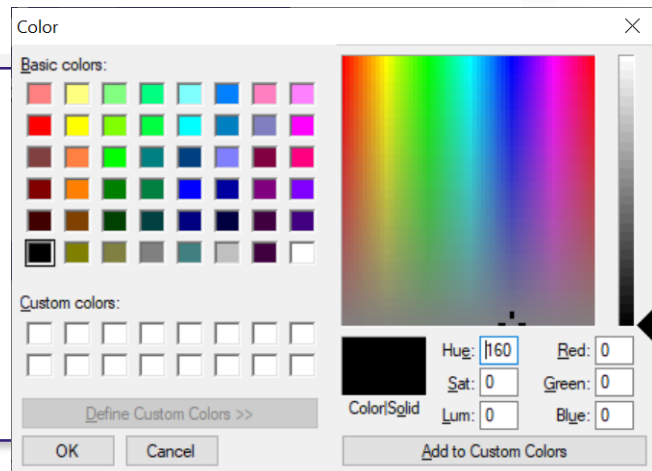
- .NET framework에서 제공하는 일반적인 대화상자
- 파일열기 대화상자 (OpenFileDialog)
- 파일저장 대화상자 (SaveFileDialog)
- 색상 대화상자 (ColorDialog)
- 폰트설정 대화상자 (FontDialog)
- 페이지설정 대화상자 (PageSetupDialog)
- 인쇄 대화상자 (PrintDialog)

ColorDialog

- 색을 선택하는 일반 대화 상자
- ShowDialog()를 호출하여 대화 상자를 표시
- Color 속성으로 선택한 색을 가져오거나 설정
- AllowFullOpen 속성으로 사용자 지정 색을 정의할 수 있는지 여부를 설정

// ColorDialog 사용

```
ColorDialog colorDialog1 = new ColorDialog();
colorDialog1.AllowFullOpen = true ;
colorDialog1.ShowHelp = true ;
// 색상 대화상자에서 색을 선택 후 확인버튼을 누르면
// 선택한 색을 텍스트박스의 글자 색으로 설정
if (colorDialog1.ShowDialog() == DialogResult.OK)
    textBox1.ForeColor = colorDialog1.Color;
```

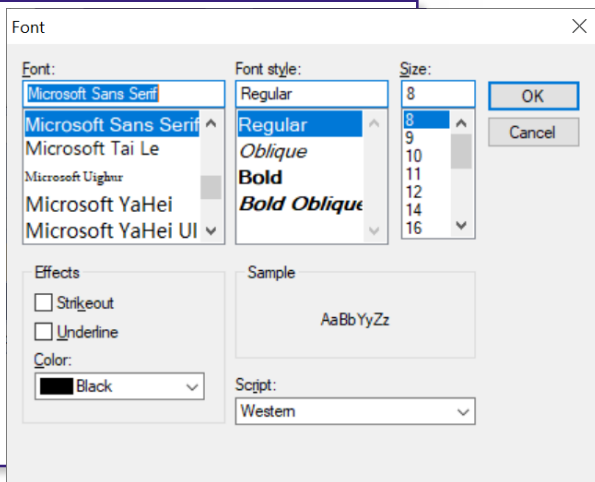


FontDialog

- 설치되어 있는 글꼴을 선택할 수 있는 대화상자
- Font 속성으로 선택한 글꼴을 가져오거나 지정
- ShowColor 속성으로 색상표 표시여부 설정
- Color 속성으로 선택한 글꼴색상을 가져오거나 지정

// FontDialog 사용

```
FontDialog fontDialog1 = new FontDialog ();  
fontDialog1.ShowColor = true;  
fontDialog1.Font = textBox1.Font;  
fontDialog1.Color = textBox1.ForeColor;  
  
if(fontDialog1.ShowDialog() != DialogResult.Cancel ) {  
    textBox1.Font = fontDialog1.Font ;  
    textBox1.ForeColor = fontDialog1.Color;  
}
```



— OpenFileDialog

- 파일을 선택하는 일반 대화상자

— SaveFileDialog

- 파일을 저장하는 일반 대화상자

| 주요 속성 | 설명 |
|------------------|---|
| FileName | 파일 대화상자에서 선택한 파일 이름을 포함하는 문자열을 가져오거나 설정 (FullPath) |
| FileNames | 대화상자에서 선택한 모든 파일의 파일 이름들 |
| Filter | 대화상자에서 “파일 형식으로 저장” 또는 “파일 형식” 상자에 표시되는 선택 옵션을 결정하는 현재 파일 이름필터 문자열을 가져오거나 설정 “Text Document(*.txt) *.txt” + “All Files *.*” “All Image Files *.bmp;*.gif;*.jpg;*.png” |
| DefaultExt | 기본 확장명을 가져오거나 설정 |
| InitialDirectory | 파일 대화상자가 표시하는 초기 디렉토리를 가져오거나 설정 |

FileDialog

```
// OpenFileDialog를 이용하여 텍스트 파일을 열고 텍스트박스에 문자열 읽어오기
OpenFileDialog openFileDialog1 = new OpenFileDialog();

//현재 디렉토리를 초기값으로 설정
openDialog1.InitialDirectory = Directory.GetCurrentDirectory();
//열기 파일의 필터 설정
openDialog1.Filter = "Text Document(*.txt)*.txt" + "|All Files|*.*";

//열기 대화상자의 클릭 결과가 OK일 경우
if (openDialog1.ShowDialog() == DialogResult.OK) {
    txtFilename.Text = "FileName:" + Path.GetFileName(openDialog1.FileName);
    txtPath.Text = "DirectoryName:" + Path.GetDirectoryName(openDialog1.FileName);

    //선택한 파일에서 텍스트를 읽기 위한 StreamReader 생성
    //매개변수로 시스템의 현재 ANSI 코드 페이지에 대한 인코딩 필요 - 한글처리
    StreamReader sr = new StreamReader(openDialog1.FileName, Encoding.Default);

    //StreamReader 에서 현재위치에서부터 끝까지 텍스트를 읽어오기
    txtMemo.Text = sr.ReadToEnd();

    //StreamReader 닫기
    sr.Close();
}
```

FileDialog

//SaveFileDialog를 이용하여 텍스트 파일로 텍스트박스의 문자열을 저장하기

```
SaveFileDialog saveDialog1 = new SaveFileDialog();
```

//현재 디렉토리 설정

```
saveDialog1.InitialDirectory = Directory.GetCurrentDirectory();
```

//저장 파일의 필터 설정

```
saveDialog1.Filter = "Text Document(*.txt)|*.txt" + "|All Files|*.*";
```

```
if (saveDialog1.ShowDialog() == DialogResult.OK) {
```

//선택한 파일명으로 저장할 스트림 생성 - 한글 처리

```
StreamWriter sw = new StreamWriter(saveDialog1.FileName, false,  
Encoding.Default );
```

//스트림을 통하여 텍스트 파일저장

```
sw.Write(txtMemo.Text);
```

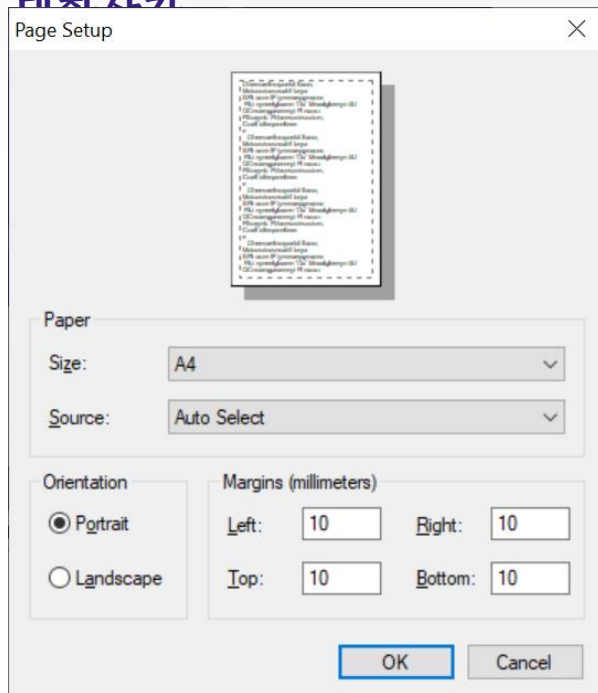
//스트림 닫기

```
sw.Close();
```

```
}
```

PageSetupDialog

- 페이지 관련 인쇄 설정 (용지, 여백, 페이지 방향 등)을 위한 대화상자

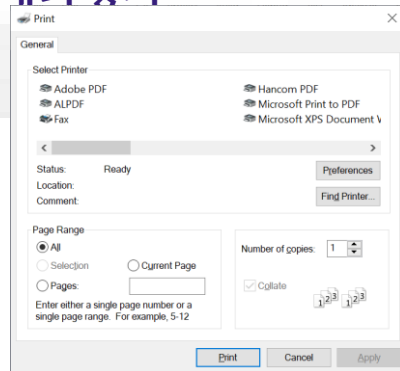


PrintDialog

- 인쇄 옵션(프린터, 인쇄범위, 매수)를 선택할 수 있는 대화상자

인쇄 작성

- 인쇄문서에 대한 PrintDocument 객체 생성
- PrintPage 이벤트 핸들러에 인쇄작업 코드 작성
 - ✓ PrintDocument의 print()함수에 의해 발생하는 이벤트에 대한 처리 내용
 - ✓ PrintPageEventArgs e의 Graphics를 이용하여 문서 또는 그림 출력
- PageSetupDialog 로 용지 방향, 페이지 크기, 여백 등 인쇄할 페이지 설정
- PrintDialog 로 인쇄할 프린터, 인쇄할 페이지, 매수 등 설정



Print Dialog

```
// PrintDocument : PrintPage이벤트 처리에서 출력작업
private System.Drawing.Printing.PrintDocument printDocument1;

this.printDocument1 = new System.Drawing.Printing.PrintDocument();
this.printDocument1.PrintPage +=
    new System.Drawing.Printing.PrintPageEventHandler(this.printDocument1_PrintPage);

//인쇄작업을 위한 이벤트처리 : 지정된 위치, 색상,폰트로 문자열 그리기
private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
{
    System.Drawing.Font printFont = new System.Drawing.Font("Arial", 12,
        System.Drawing.FontStyle.Regular);
    e.Graphics.DrawString(txtMemo.Text , printFont, System.Drawing.Brushes.Black,
        10, 10);
}
```

Print Dialog

// PageSetupDialog : PrintDocument 의 값으로 초기설정

```
private void pageToolStripMenuItem_Click(object sender, EventArgs e) {  
    PageSetupDialog pageSetupDialog1 = new PageSetupDialog();  
    pageSetupDialog1.PageSettings = printDocument1.DefaultPageSettings;  
    pageSetupDialog1.PrinterSettings = printDocument1.PrinterSettings;  
    if (pageSetupDialog1.ShowDialog() == DialogResult.OK) {  
        txtMemo.Text = pageSetupDialog1.PageSettings.Margins.ToString() +  
            pageSetupDialog1.PageSettings.PaperSize.ToString();  
    }  
}
```

// PrintDialog : Document 속성에 PrintDocument 대입, 인쇄버튼을 누르면
PrintDocument의 print() 함수 호출로 인쇄작업에 대한 이벤트 발생

```
private void printToolStripMenuItem_Click(object sender, EventArgs e) {  
    PrintDialog printDialog1 = new PrintDialog();  
    printDialog1.Document = printDocument1;  
    DialogResult result = printDialog1.ShowDialog();  
    if (result == DialogResult.OK) {  
        //print() 함수 호출로 printDocument의 PrintPage 이벤트 처리  
        printDocument1.Print();  
    }  
}
```

— 사용자 정의 대화상자

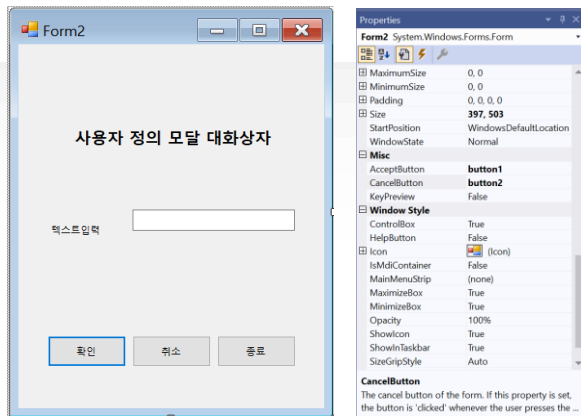
- 사용자와 상호 작용하고 정보를 검색하는 데 사용하는 폼
- Windows Forms을 추가하여 고유한 사용자 지정 대화 상자 생성
- 대화 상자에는 일반적으로 메뉴 모음, 창 스크롤 막대, 최소화 및 최대화 단추, 상태 표시줄 또는 크기를 조정할 수 있는 테두리가 없으므로 관련 속성 설정
 - ✓ `FormBorderStyle = FixedDialog`
 - ✓ `MinimizeBox/MaximizeBox` 속성을 `false`로 설정

— 사용자 정의 대화상자 표시

- Modal 대화상자 표시 **ShowDialog()** 메소드
 - ✓ 해당 대화 상자가 닫힐 때까지 ShowDialog 메소드 다음의 코드가 실행되지 않음
- Modeless 대화상자 표시 **Show()** 메소드
 - ✓ 폼이 표시된 직후 Show 메소드 다음의 코드가 실행

— 사용자 정의 대화상자 확인/취소 버튼

- 대화상자의 반환 값
 - ✓ DialogResult.OK, DialogResult.Cancel
- 버튼 속성 (또는 클릭 이벤트 처리)에서 DialogResult 속성을 DialogResult.OK 또는 DialogResult.Cancel으로 설정
- Form의 AcceptButton/CancelButton 속성
 - ✓ 해당 버튼 설정 시 단추에 포커스가 없는 경우에도 Enter 키 또는 Esc 키로

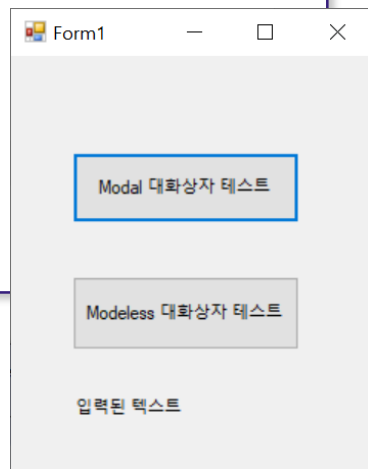


— 사용자 정의 대화상자 작성 예

- Form1에 버튼 추가
- Visual Studio 프로젝트에서 Windows Form 추가를 선택하여 Form2 작성
 - ✓ Form2의 Text 속성(사용자 대화상자)
 - ✓ Label1 추가
 - ✓ 텍스트 상자1 추가
 - ✓ 버튼1 추가 : Text 속성(확인), DialogResult 속성(OK)
 - ✓ 버튼2 추가 : Text 속성(취소), DialogResult 속성(Cancel)
- Form1
 - ✓ 버튼의 클릭이벤트 작성

— 사용자 정의 대화상자 작성 예

```
private void button1_Click(object sender, EventArgs e) {  
    Form2 form2 = new Form2();  
    if (form2.ShowDialog() == DialogResult.OK) // modal dialog  
        MessageBox.Show("확인을 눌렀습니다");  
    else  
        MessageBox.Show("취소를 눌렀습니다");  
    form2 = null;  
}
```



— 사용자 정의 모달 대화 상자 생성 및 표시

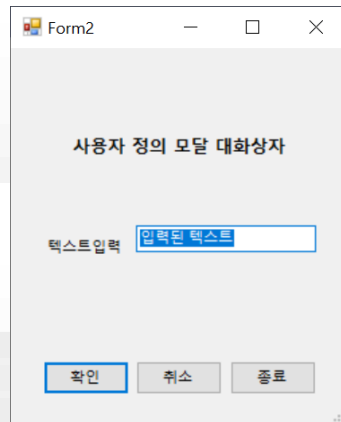
- `Form2 f = new Form2();`
- `ShowDialog()` 메소드 사용 // modal dialog

— 모달 대화 상자 닫기 (Close)

- `DialogResult` 속성을 `OK` 또는 `Cancel`로 지정하면 폼이 숨겨짐
- 폼의 닫기 단추를 클릭하면 `DialogResult` 속성이 `Cancel`로 설정

— 대화 상자를 소유하는 폼을 **Owner** 속성으로 설정하여 접근

— 소유 폼과 대화상자 사이에 서로 접근 가능한 데이터는 **public**으로 설정



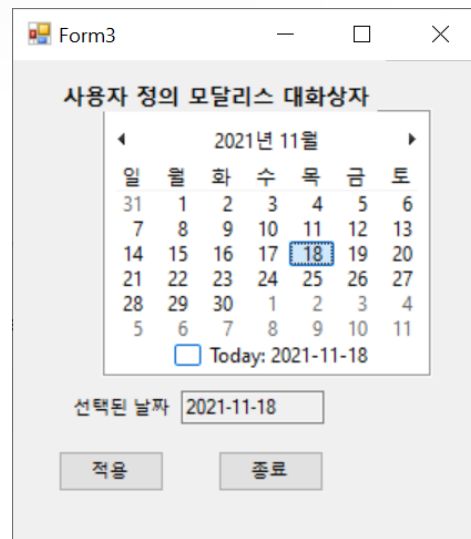
Custom Modeless Dialog

— 사용자 정의 모달리스 대화 상자 생성 및 표시

- `Form3 form3 = new Form3();`
- `Show()` 메소드 사용 // modeless dialog

— 대화 상자를 소유하는 폼을 **Owner** 속성으로 설정하여 접근

— **Owner** 폼에 접근할 데이터는 **public**으로 설정



Custom Modeless Dialog

```
//Modal 대화상자 생성 및 표시
public partial class Form1 : Form {
    private void button1_Click(object sender, EventArgs e) {
        Form2 form2 = new Form2();
        form2.Owner = this; //대화상자의 소유 폼 설정
        form2.ShowDialog(); //모달 대화상자로 표시
        if (form2.DialogResult == DialogResult.OK) { //대화상자의 결과 확인
            this.label1.Text = form2.textBox1.Text; //대화상자의 public데이터 접근
            MessageBox.Show(label1.Text, "Modal Dialog OK");
        } else if (form2.DialogResult == DialogResult.Cancel) {
            MessageBox.Show("Cancel", "Modal Dialog Cancel");
        }
    }
}

partial class Form2 { // modal dialog
    this.button1.DialogResult = System.Windows.Forms.DialogResult.OK;
    this.button2.DialogResult = System.Windows.Forms.DialogResult.Cancel;
    this.Controls.Add(this.button1);
    this.Controls.Add(this.button2);
    this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
    this.MaximizeBox = false;
    this.MinimizeBox = false;
    this.Text = "ModalForm";
    ...
}
```

Custom Modeless Dialog

// Modal Form (대화상자)

```
public partial class Form2 : Form {
```

```
    public System.Windows.Forms.TextBox textBox1; //소유 폼에서 접근할 데이터
```

```
    private void Form2_Load(object sender, EventArgs e) {
```

```
        textBox1.Text = ((Form1)this.Owner).label1.Text; //소유 폼의 public 데이터 접근
```

```
    }
```

//확인 버튼 누르면 확인 메시지 박스

```
    private void button1_Click(object sender, EventArgs e) {
```

```
        MessageBox.Show("확인을누름", "OK버튼클릭이벤트");
```

```
    }
```

//종료버튼을 누르면 종료확인 후 처리

```
    private void button3_Click(object sender, EventArgs e) {
```

```
        //대화상자를 닫히지 않도록 설정 후 처리 계속
```

```
        this.DialogResult = DialogResult.None;
```

```
        DialogResult r = MessageBox.Show("종료하시겠습니까?", "Dialog Close",
```

```
            MessageBoxButtons.YesNo,
```

```
            MessageBoxIcon.Question);
```

```
        if (r == DialogResult.Yes) {
```

```
            this.Close();
```

```
        }
```

```
    }
```

```
}
```

Custom Modeless Dialog

```
//Modeless 대화상자 생성 및 표시
public partial class Form1 : Form {
    private void button2_Click(object sender, EventArgs e) {
        Form3 form3 = new Form3();
        form3.Owner = this; //대화상자의 소유 폼 설정
        form3.Show();      //모델리스로 표시
    }
}

public partial class Form3 : Form { // modeless dialog
    //monthCalendar의 날짜를 선택했을 때 발생하는 이벤트 처리
    private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e) {
        textBox1.Text = e.Start.ToShortDateString(); //선택한 날짜를 텍스트박스에 표시
    }
    //적용버튼을 누르면 선택한 날짜를 메인 폼(소유 폼) 의 레이블에 설정
    private void button1_Click(object sender, EventArgs e) {
        ((Form1)this.Owner).label1.Text = this.textBox1.Text;
    }
    //Modeless 대화상자 종료
    private void button2_Click(object sender, EventArgs e) {
        this.Close();
    }
}
```


— MDI (Multiple Document Interface)

- 하나의 응용프로그램에서 동시에 여러 개 문서 작업 가능

MDI Child 속성

| | |
|------------|----------------|
| IsMdiChild | MDI 자식 여부를 나타냄 |
| MdiParent | MDI 부모 폼을 지정 |

MDI Parent 속성

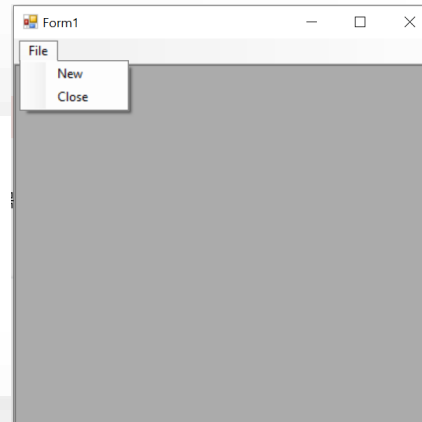
| | |
|----------------|-------------------------|
| ActiveMdiChild | 현재 활성화된 MDI 자식 폼을 반환 |
| IsMdiContainer | 폼이 MDI 부모가 될 수 있는지를 나타냄 |
| MdiChildren | MDI 자식들을 배열로 반환 |

MDI 주요 메소드 및 이벤트

| | |
|----------------|--|
| LayoutMdi 메소드 | MDI 부모 상의 자식 폼들을 정렬 Arrangelcons, Cascade, TileHorizontal, TileVertical |
| MdiChildActive | MDI 자식이 닫히거나 활성화될 때 발생 |

— MDI 부모 폼 생성 및 표시

- IsMdiContainer 속성값을 true로 설정
`form1.IsMdiContainer = true;`
- 활성화된 자식창 확인
`Form activeChild = this.ActiveMdiChild;`



— MDI 자식 폼 생성 및 표시

- 프로젝트에서 Windows Form 추가를 사용하여 자식 폼(Form2) 생성
- 부모 폼에 메뉴항목 이벤트 핸들러 작성

```
private void newToolStripMenuItem_Click(object sender, EventArgs e) {  
    Form2 newMDIChild = new Form2();           // 자식 폼 생성  
    newMDIChild.MdiParent = this;              // 자식 폼의 부모 지정  
    newMDIChild.Show();                        // 자식 폼 표시  
}
```

//MDI의 Copy 메뉴항목 클릭 이벤트 핸들러

```
private void copyToolStripMenuItem_Click(object sender, EventArgs e) {  
    Form activeChild = this.ActiveMdiChild;    // 활성화된 자식 폼  
    if (activeChild != null) {  
        RichTextBox theBox = (Rich TextBox)activeChild.ActiveControl;  
        if (theBox != null)  
            Clipboard.SetDataObject(theBox.SelectedText);  
    }  
}
```

//MDI의 Paste 메뉴항목 클릭 이벤트 핸들러

```
private void pasteToolStripMenuItem_Click(object sender, EventArgs e) {  
    Form activeChild = this.ActiveMdiChild;    // 활성화된 자식 폼  
    if (activeChild != null) {  
        RichTextBox theBox = (Rich TextBox)activeChild.ActiveControl;  
        if (theBox != null) {  
            IDataObject data = Clipboard.GetDataObject();  
            if (data.GetDataPresent(DataFormats.Text))  
                theBox.SelectedText = data.GetData(DataFormats.Text).ToString();  
        }  
    }  
}
```

//MDI의 TileHorizontal/TileVertical/Cascade/ArrangeIcons 메뉴항목 클릭 이벤트 핸들러

```
private void tileHorizontalToolStripMenuItem_Click(object sender, EventArgs e) {  
    this.LayoutMdi(MdiLayout.TileHorizontal);  
}
```

— 사용자 정의 컨트롤

- 기존 컨트롤을 상속받아 사용자 정의 새로운 컨트롤을 작성
- UserControl을 상속받아 합성 컨트롤을 작성

숫자만 입력 가능한 텍스트 상자 컨트롤 생성 예제

- 파일 메뉴에서 새로 만들기 후 프로젝트
- Visual C# 프로젝트 목록에서 “Windows Forms 컨트롤 라이브러리” 선택하고 이름을 “**NumberTextBoxLib**”을 입력
- 솔루션 탐색기에서 UserControl1.cs를 NumberTextBox.cs로 변경
- NumberTextBox.cs 코드에서 상속을 UserControl에서 TextBox로 변경
 - ✓ `public partial class NumberTextBox: TextBox`
- NumberTextBox.Designer.cs에 InitializeComponent()에서 **AutoScaleMode** 속성은 삭제
- NumberTextBox.cs 코드에서 **OnKeyPress(...)** 메소드를 재정의
- 솔루션을 빌드하면 컨트롤이 완성

Custom Control

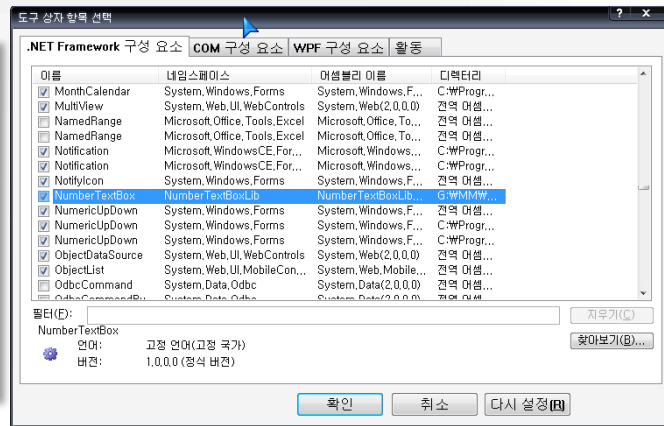
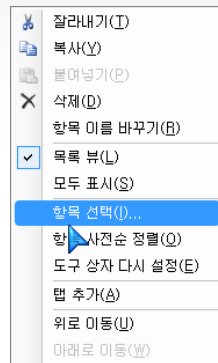
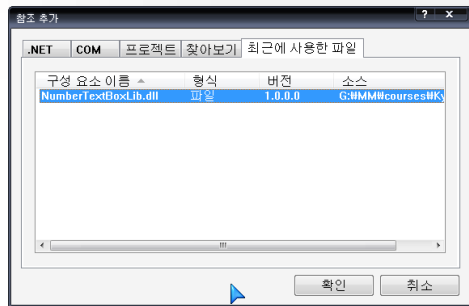
```
namespace NumberTextBoxLib
{
    public partial class NumberTextBox : TextBox
    {
        public NumberTextBox()
        {
            InitializeComponent();
        }

        private void InitializeComponent()
        {
            components = new System.ComponentModel.Container();
            //this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        }

        // OnKeyPress 메소드 재정의
        protected override void OnKeyPress(KeyPressEventArgs e)
        {
            base.OnKeyPress(e);
            int value = 0;
            e.Handled = !int.TryParse(e.KeyChar.ToString(), out value);
        }
    }
}
```

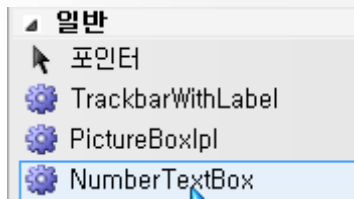
— 사용자 정의 컨트롤 사용

- 솔루션 탐색기의 “참조”에 “**NumberTextBoxLib.dll**”를 “참조추가”
- “도구상자”에서 오른쪽 마우스 “항목선택”을 한 후 “찾아보기” 버튼에서 “**NumberTextBoxLib.dll**” 다시 로딩한 후, “**NumberTextBox**”에 체크가 됐는지 확인



— 사용자 정의 컨트롤 사용

- 폼에 “**NumberTextBox**”를 사용하여 디자인하고
솔루션을 빌드하면 컨트롤이 완성



Custom Control

```
partial class Form2 {
    private void InitializeComponent() {
        //중간 생략...
        this.numberTextBox1.Location = new System.Drawing.Point(72, 36);
        this.numberTextBox1.Name = "numberTextBox1";
        this.numberTextBox1.Size = new System.Drawing.Size(100, 21);
        this.numberTextBox1.TabIndex = 5;
        //중간 생략...
        public NumberTextBoxLib.NumberTextBox numberTextBox1; // form1에서 사용
    }
    private void Form2_Load(object sender, EventArgs e) {
        numberTextBox1.Text = ((Form1)this.Owner).thickness.ToString();
    }
}
partial class Form1 {
    private void thicknessToolStripButton_Click(object sender, EventArgs e) {
        Form2 form2 = new Form2();
        form2.Owner = this;
        form2.ShowDialog();
        if (form2.DialogResult == DialogResult.OK) {
            thickness = int.Parse(form2.numberTextBox1.Text);
            System.Diagnostics.Trace.WriteLine("Debug: THICKNESS=" + thickness);
            form2.Close();
        }
        form2 = null;
    }
}
```