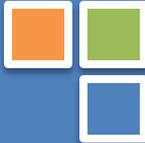


C# 입문 : 이론과 실습



제 7장 원폼 애플리케이션



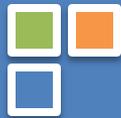
목차

- 윈폼 애플리케이션
- 폼 클래스
- 컨트롤 클래스



원폼 애플리케이션

- 통합 개발환경 시작하기
- 소스코드 보기
- 원폼 애플리케이션 작성하기



윈폼 애플리케이션

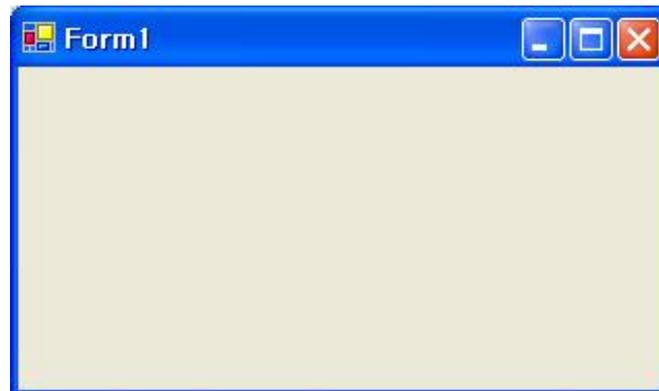
■ 윈폼 애플리케이션

- 컨트롤을 사용하여 프로그래머가 원하는 화면을 구성하고 이벤트가 발생했을 때 처리하고자 하는 작업을 이벤트 처리기에 기술하는 방식으로 프로그래밍된 프로그램.
- 윈도우 폼에 컨트롤 또는 컴포넌트를 배치하고 이벤트 처리기를 등록하여 사용자의 다양한 요구를 입력으로 받아 처리한 후 실행 결과를 응답해 주는 방식으로 작동.



윈도우 폼

- 윈도우 폼
 - 단순히 폼이라고 부름.
 - 운영체제에서 제공하는 기본적인 화면 단위인 창을 말하는 개념.
 - 사각형 모양의 작은 화면 영역을 의미
 - 사용자에게 정보를 제공하고 사용자가 입력하는 자료를 받음.





윈폼 애플리케이션의 핵심 클래스

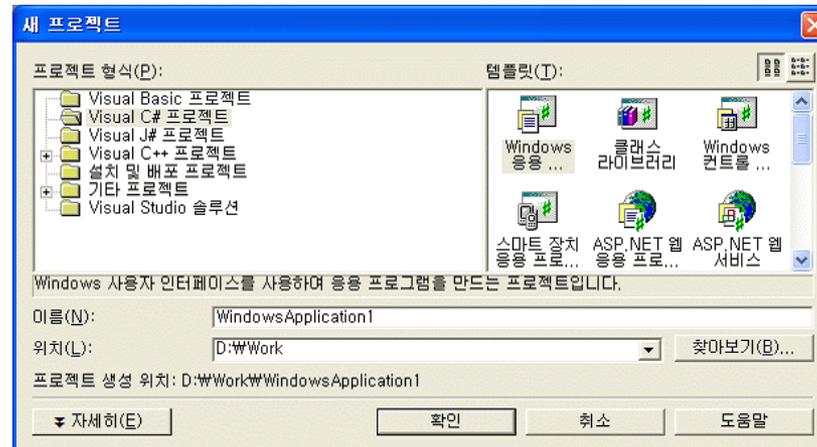
- 폼 클래스
 - 윈도우 폼을 나타내는 클래스.
 - System.Windows.Forms 네임스페이스 속함.
- 컴포넌트 클래스
 - 컨트롤 클래스의 베이스 클래스이며 화면에 직접적으로 나타나지 않으나 개념적인 부분을 나타내는 클래스.
- 컨트롤 클래스
 - 폼에 직접 표시되는 컨트롤을 위한 클래스.



통합 개발환경 사용하기 [1/4]

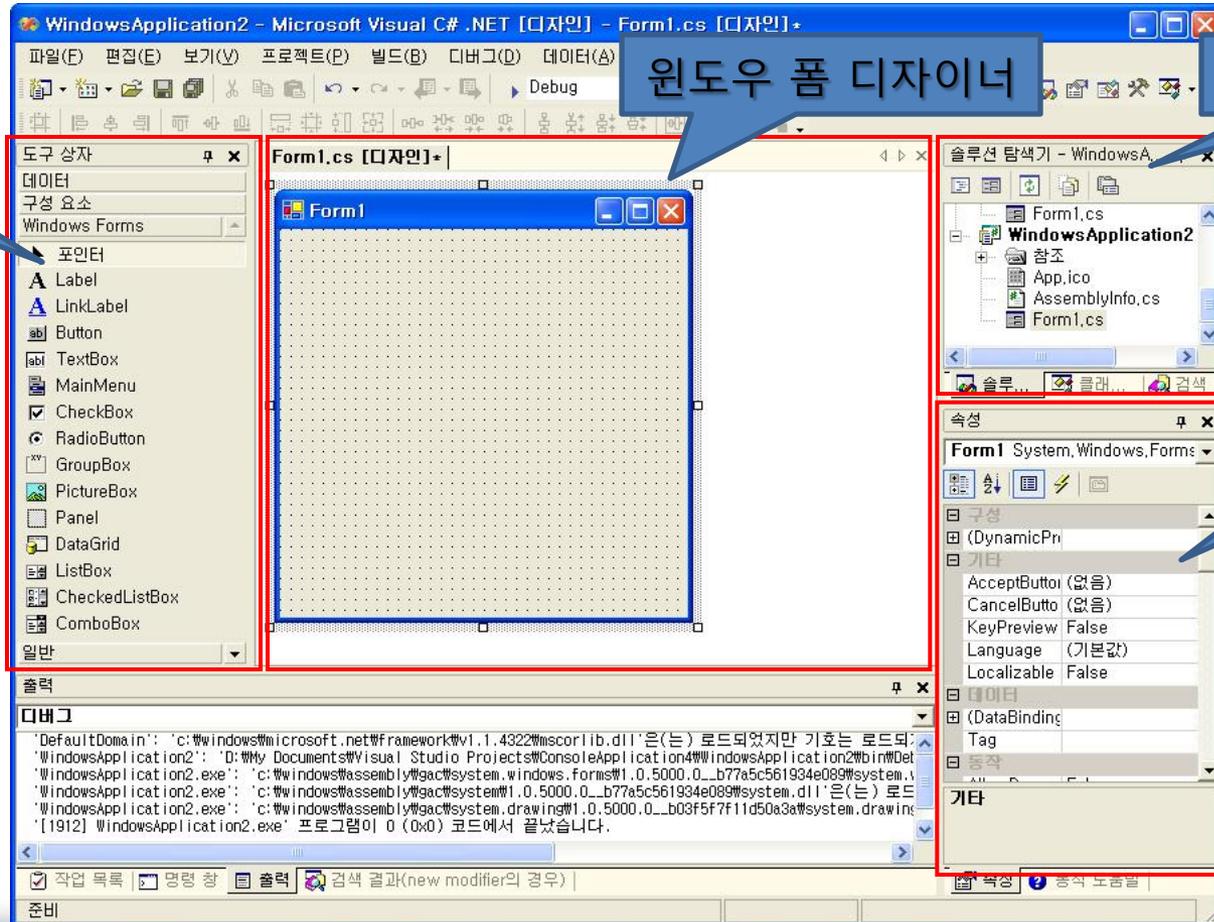
■ 1. 새 프로젝트 시작

- **【파일(F)】** ⇒ **【새로 만들기(N)】** ⇒ **【프로젝트(P)】** 를 선택.



- **【프로젝트 형식(P)】** ⇒ **【Visual C# 프로젝트】** 그리고 **【템플릿(T)】** ⇒ **【Windows 응용 프로그램】** 을 선택 .

통합 개발환경 사용하기 [2/4]

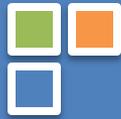


도구상자

윈도우 폼 디자이너

솔루션 탐색기

속성 브라우저



통합 개발환경 사용하기 [3/4]

- 윈도우 폼 디자이너
 - 폼의 크기를 조정하고 폼에 컨트롤을 배치하고 배치된 컨트롤의 위치와 크기를 수정할 수 있는 영역.
- 도구 상자
 - 폼에 배치할 수 있는 컨트롤과 컴포넌트의 목록을 보여주는 곳
- 속성 브라우저
 - 폼 디자이너에서 선택한 폼이나 컴포넌트의 프로퍼티와 이벤트 목록을 보여주는 곳
- 솔루션 탐색기
 - 응용 프로그램을 구성하고 있는 파일과 관련 항목을 보여주는 곳
 - 현재 응용 프로그램에 새 폼이나 파일을 추가하거나 또는 존재하는 파일이나 항목을 삭제할 때 사용



통합 개발환경 사용하기 [4/4]

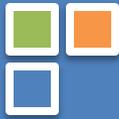
■ 속성 브라우저

The screenshot shows the Visual Studio Properties window for a **Text** control on **Form1**. The window is divided into several sections:

- 프로퍼티 (Properties):** The top section showing a tree view of properties. The **Text** control is selected, and its properties are listed below.
- 이벤트 (Events):** The middle section showing a list of events and their handlers.
- 디자이너에서 선택된 항목 (Item selected in designer):** A callout pointing to the **Text** control in the tree view.
- 선택된 항목 (Selected item):** A callout pointing to the **Text** control in the tree view.
- 항목별 정렬 (Item sorting):** A callout pointing to the sorting icons (list, A-Z, Z-A, icon, lightning bolt) in the tree view.
- 사전순 정렬 (Sort by name):** A callout pointing to the 'A-Z' sorting icon.
- 선택된 항목 설명 (Selected item description):** A callout pointing to the description text: "컨트롤에 포함된 텍스트입니다." (Text contained in the control).

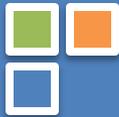
The Properties window shows the following properties for the **Text** control:

- 액세스 가능성 (Accessibility):** AccessibleC, AccessibleM, AccessibleF Default
- 창 스타일 (Appearance):** ControlBox True, HelpButton False
- 아이콘 (Icon):** Icon (아이콘), IsMdiContain False



소스코드 보기 [1/3]

- 생성된 프로젝트의 파일
 - App.ico : 응용 프로그램에서 사용되는 기본 아이콘에 대한 파일
 - AssemblyInfo.cs : 프로젝트가 생성하는 어셈블리를 설명하고 버전 관리 정보를 지정하는 데 사용하며, 애트리뷰트 정의를 포함하는 C# 소스파일
 - Form1.cs : 응용 프로그램의 C# 소스파일
 - Form1.resx : Form1.cs 파일과 관련된 리소스 정보를 저장하는 XML 파일
 - WindowsApplication1.csproj : Visual C# 프로젝트 파일
 - WindowsApplication1.csproj.user : 사용자 정의 설정을 포함하는 Visual C# 프로젝트 파일
 - WindowsApplication1.suo : 사용자 정의 정보를 포함하는 Visual C# 솔루션 파일



소스코드 보기 [2/3]

```
public class Form1 : System.Windows.Forms.Form {
    // 필드 항목 선언
    public Form1()                                // --- ②
    {
        InitializeComponent(); // 컴포넌트 초기화
        // 기타 멤버 초기화
    }
    protected override void Dispose(bool disposing) // --- ④
    {
        // 사용중인 리소스를 정리
    }
    private void InitializeComponent()             // --- ③
    {
        // 클래스에서 사용하는 컴포넌트 초기화
    }
    static void Main() {                          // --- ①
        Application.Run(new Form1());
    }
}
```



소스코드 보기 [3/3]

- ① Main() 메소드
 - 응용 프로그램의 시작점
 - Application 클래스의 Run() 메소드를 호출하여 응용 프로그램을 실행.

- ② 생성자
 - 폼에서 사용하는 각종 컴포넌트와 클래스의 멤버 초기화

- ③ InitializeComponent() 메소드
 - 폼에 있는 각종 컴포넌트들 초기화

- ④ Dispose() 메소드
 - 응용 프로그램이 종료될 때 실행
 - 응용 프로그램이 사용한 모든 리소스를 해제.



윈폼 애플리케이션 작성하기 [1/3]

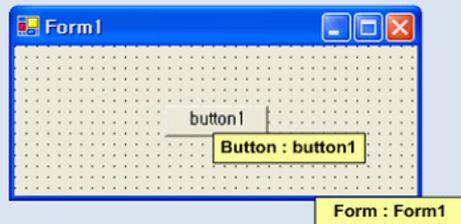
- 폼 설계
- 컴포넌트
- 프로퍼티
- 멤버
- 이벤트처리기
- 애플리케이션 실행



원폼 애플리케이션 작성하기 [2/3]

[예제 7.1 – DisplayStartDateApp.cs]

1) 폼 설계



2) 프로퍼티

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

3) 멤버

```
public Form1() {
    //...
    startDateTime = DateTime.Now;
}
DateTime startDateTime;
public DateTime GetStartDateTime() {
    return startDateTime;
}
```



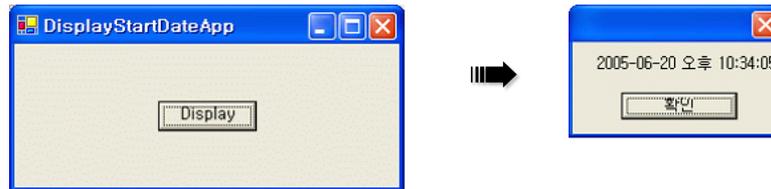
원폼 애플리케이션 작성하기 [3/3]

4) 이벤트 처리기

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

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

실행 방법 : 애플리케이션을 실행한 후, 폼에 있는 "Display" 버튼을 클릭한다.
실행 결과 :



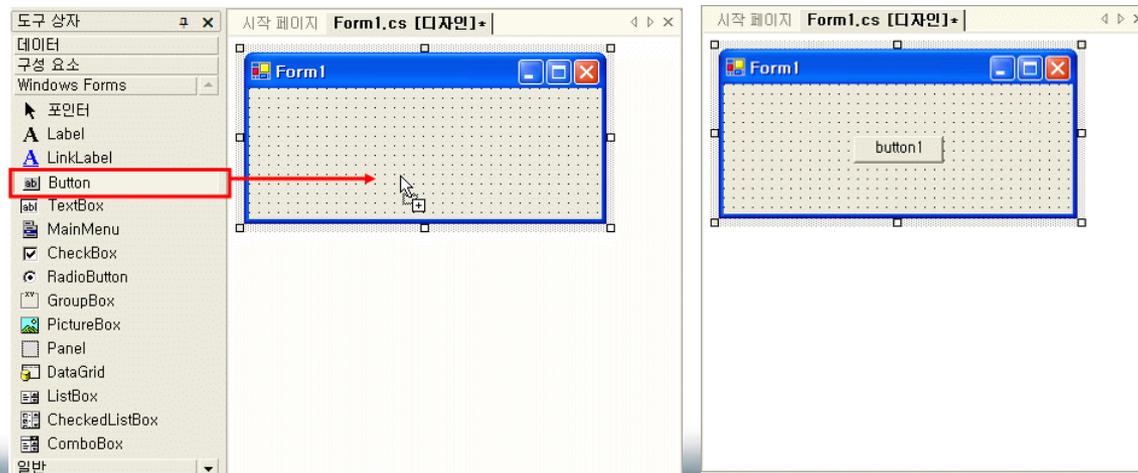


(1) 폼 설계

■ 폼 설계

- 폼에 컨트롤을 배치한 모습 표시.
- 폼에 배치되는 컨트롤의 종류와 배치된 컨트롤의 이름과 위치를 그림 형태로 표시.
- 노란색 텍스트 상자는 "ClassType : Name" 형식으로 표시
 - 컨트롤의 종류(클래스 이름)와 이름(객체 이름)을 나타냄.

■ 컨트롤의 추가





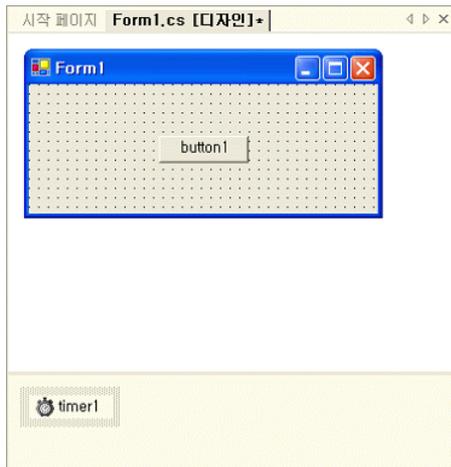
(2) 컴포넌트

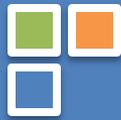
■ 컴포넌트 목록

컨트롤 : (Name)	프로퍼티	값
Timer : timer1	Images	FLGSKOR.ICO

■ 컴포넌트 추가

■ 컨트롤 추가와 유사





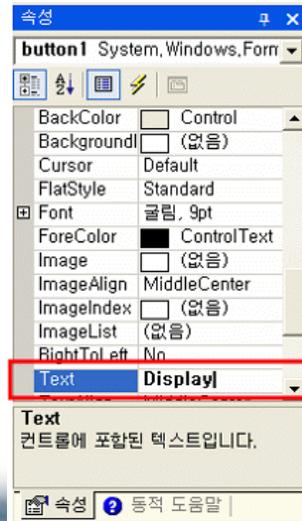
(3) 프로퍼티

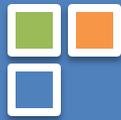
■ 프로퍼티 목록

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

■ 프로퍼티 값 변경

- 속성 브라우저를 이용.





(4) 멤버

■ 멤버

- 컴포넌트와 컨트롤을 제외한 클래스 멤버에 대한 선언과 초기화 등을 소스코드로 작성하는 곳.

■ 멤버 코드 추가

- 생성자에 초기화 부분 추가

```
public Form1(){  
    // Windows Form 디자이너 지원에 필요합니다.  
    //  
    InitializeComponent();  
    // TODO: InitializeComponent를 호출한 다음 생성자 코드를 추가합니다.  
    //  
    startDateTime = DateTime.Now;  
}
```

- 멤버 선언이나 메소드 등의 소스 코드 추가
 - 폼 클래스내의 적당한 곳에 추가.



(5) 이벤트 처리기 [1/2]

■ 이벤트 처리기

■ 통합 환경에서 생성하는 처리기의 이름

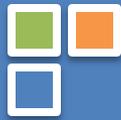
- <이벤트 처리기를 등록하려는 객체의 이름>_<이벤트 이름>

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

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

■ 매개변수

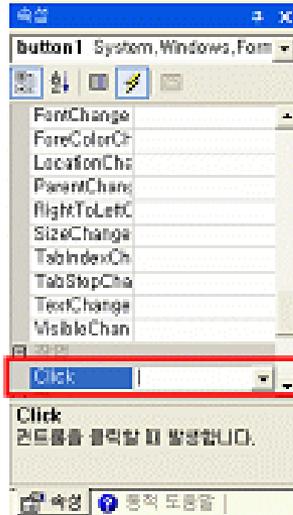
- 이벤트를 발생시킨 객체
- 이벤트에 관련된 정보를 가진 객체
 - EventArgs 클래스형이나 이의 파생 클래스형



(5) 이벤트 처리기 [2/2]

■ 이벤트 처리기 등록방법

- ① 이벤트 처리기를 등록하고자 하는 폼이나 컨트롤을 선택.
- ② 속성 브라우저에서 "이벤트" 버튼을 클릭.



- ③ 목록에서 처리기를 등록하고자 하는 이벤트를 더블 클릭
- ④ 소스코드 편집 창에서 이벤트 처리기의 내용을 입력



(6) 애플리케이션 실행

■ 컴파일과 실행

■ 컴파일하는 방법

- 메인 메뉴를 통하여 **【빌드(B)】** → **【솔루션 빌드(B)】**를 선택

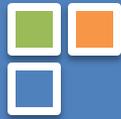
■ 컴파일한 후 바로 실행하는 방법

- **디버그(D)】** → **【시작(S)】**이나 **【디버그(D)】** → **【디버깅하지 않고 시작(G)】**을 선택

실행 방법 : 애플리케이션을 실행한 후, 폼에 있는 "Display" 버튼을 클릭한다.

실행 결과 :



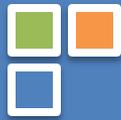


폼 클래스

■ 폼 클래스

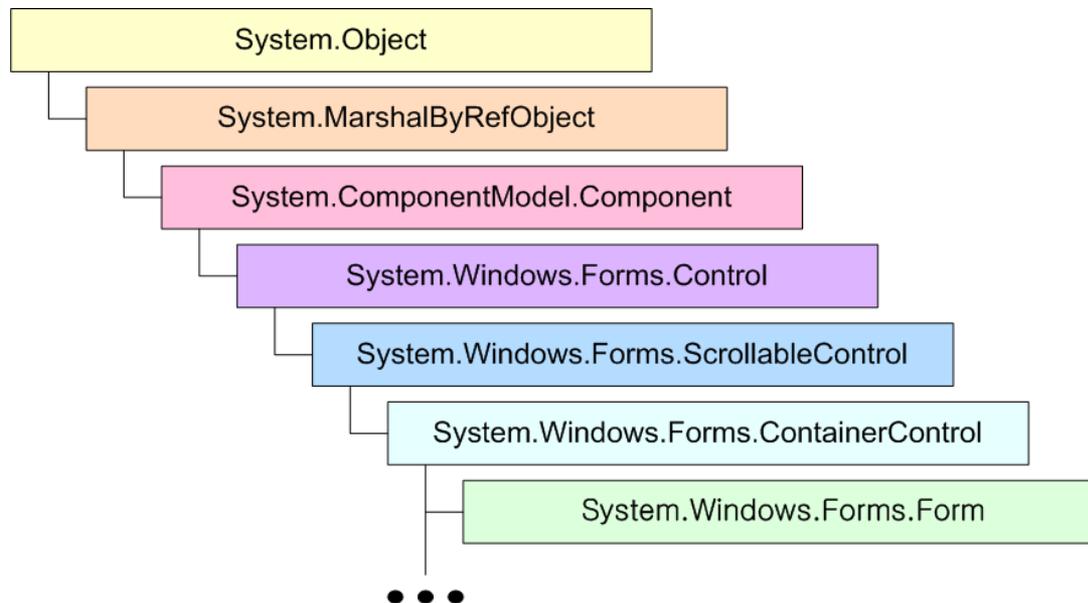
- Form 클래스를 나타냄.
- 폼의 외형을 설정하는 프로퍼티, 폼의 동작을 정의하는 메소드, 그리고 사용자와 상호작용을 처리하는 이벤트 등이 정의되어 있는 클래스
- 윈도우 폼을 다루는 핵심 클래스로 윈도우 폼을 사용하는 모든 클래스의 베이스 클래스.

```
using System.Windows.Forms;
public class UserDefinedForm : Form {
    // 새 클래스 멤버 정의
    // Form 클래스의 멤버 재정의
}
```



폼 클래스의 계층도

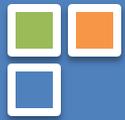
- 폼 클래스
 - 윈도우 폼을 사용하는 모든 클래스의 베이스 클래스
 - System.Windows.Forms 네임스페이스에 포함





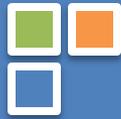
폼 클래스의 베이스 클래스

- Component 클래스
 - .NET 프레임워크에서 컴포넌트 기반 프로그래밍 기법에서의 컴포넌트 개념을 지원하는 클래스
 - System.ComponentModel 네임스페이스에 포함
 - 윈폼 애플리케이션뿐만 아니라 컴포넌트 개념이 필요한 다른 곳에서도 사용
 - 윈폼에서 제공하는 화면을 구성하는 여러 요소뿐만 아니라 화면에 표시되지 않는 요소도 컴포넌트로 표현.
- Control 클래스
 - 윈폼 애플리케이션에서 화면에 표시되는 구성요소를 나타내기 위해서 사용되는 컨트롤들의 베이스 클래스
 - 폼과 폼에 배치되는 여러 요소에서 공통적으로 필요한 멤버를 정의
- ScrollableControl 클래스
 - 스크롤 개념이 필요한 컨트롤을 정의하기 위해서 사용되는 베이스 클래스
- ContainerControl 클래스
 - 여러 컨트롤이나 컴포넌트를 포함할 수 있는 컨트롤에서의 포커스 관리를 하기 위한 베이스 클래스



폼 클래스의 멤버

- InitializeComponent() 메소드
 - 폼에 있는 각종 컴포넌트들을 초기화
 - 통합 개발환경이 관리하는 메소드
- Form 클래스의 자체 정의 멤버
- Control 클래스의 멤버



폼 클래스의 주요 프로퍼티

■ 주요 프로퍼티

프로퍼티	설명
FormBorderStyle	폼의 테두리 모양을 설정하거나 참조.
StartPosition	폼이 처음 나타나는 위치를 설정하거나 참조



FormBorderStyle 프로퍼티

■ 용도

- 폼의 테두리 모양을 설정하는 프로퍼티

■ 형식

- `public FormBorderStyle FormBorderStyle {get; set;}`

■ FormBorderStyle 열거형

기호상수	설명
None	테두리가 없음
FixedSingle	고정된 단일선 테두리
Fixed3D	고정된 3차원 테두리
FixedDialog	고정된 대화상자 스타일의 굵은 테두리
Sizable	크기를 조정할 수 있는 테두리 (기본값)
FixedToolWindow	크기를 조정할 수 없는 도구 창 테두리
SizableToolWindow	크기를 조정할 수 있는 도구 창 테두리



StartPosition 프로퍼티

■ 용도

- 윈도우 폼 응용 프로그램이 실행될 때 처음 나타나는 폼의 위치를 결정하는 프로퍼티

■ 형식

- `public FormStartPosition StartPosition {get; set;}`

■ FormStartPosition 열거형

기호상수	설명
CenterParent	폼이 해당 부모 폼의 범위 내에서 가운데에 맞춰진다
CenterScreen	폼이 현재 디스플레이의 가운데에 맞춰지며 크기는 해당 폼의 크기 내에서 지정된다.
Manual	폼의 위치는 Location 프로퍼티에 의해 결정된다
WindowsDefaultLocation	폼의 위치는 윈도우 운영 체제가 결정한다.
WindowsDefaultBounds	폼의 위치와 크기는 윈도우 운영 체제가 결정한다.



폼 클래스의 메소드

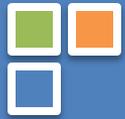
■ 주요 메소드

메소드	설명
Close()	폼을 닫는다.
Activate()	폼을 활성화 한다.
AddOwnedForm()	다른 폼을 현재 폼에 소유시킨다.
RemoveOwnedForm()	현재 폼에 소유된 폼을 제거한다.
SetDesktopBounds()	바탕 화면에서 폼의 범위를 설정한다.
SetDesktopLocation()	바탕 화면에서 폼의 위치를 설정한다.



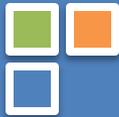
Close() 메소드

- 용도
 - 화면에 있는 폼을 닫기 위해서 사용
- 형식
 - `public void Close();`
- 설명
 - 메인 폼(main form)이 닫히는 경우에는 애플리케이션이 종료
 - 자식 폼(child form)이 닫히는 경우는 폼이 화면에서 사라짐.
 - Close() 메소드는 Closing 이벤트와 Closed 이벤트를 발생시킴.



Activate() 메소드

- 용도
 - 폼을 활성화하고 포커스를 주기 위해서 사용
- 형식
 - `public void Activate();`



SetDesktopBounds()/SetDesktopLocation() 메소드

■ 용도

■ SetDesktopBounds() 메소드

- 바탕화면에서 폼이 나타날 수 있는 범위를 설정할 때 사용

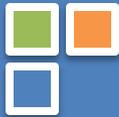
■ SetDesktopLocation() 메소드

- 바탕화면에서 폼의 위치를 정하기 위해서 사용

■ 형식

■ `void SetDesktopBounds(int x, int y, int width, int height);`

■ `void SetDesktopLocation(int x, int y);`



AddOwnedForm()/RemoveOwnedForm() 메소드

■ 용도

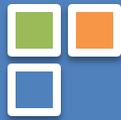
- 다른 폼을 소유하거나 소유된 폼을 제거할 때

■ 형식

- `public void AddOwnedForm(Form ownedForm);`
- `public void RemoveOwnedForm(Form ownedForm);`

■ 설명

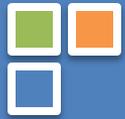
- 폼 Form1이 폼 Form2를 소유하고 있는 경우
 - `(Form1.AddOwnedForm(Form2))`
- Form1을 최소화하거나 닫는 경우에는 Form2도 같이 최소화되거나 닫힘.



폼 클래스의 이벤트

■ 주요 이벤트

이벤트	설명
Activated	폼이 활성화될 때 발생한다.
Closed	폼이 종료된 후 발행한다.
Closing	폼의 종료 과정 중에 발생한다.
Deactivate	폼이 비활성화될 때 발생한다.
Load	폼이 로드될 때 발생한다.



Load/Closed 이벤트 [1/3]

■ 용도

■ Load 이벤트

- 폼이 메모리로 로딩될 때 발생하는 이벤트
- 주로 폼에서 사용하는 자원을 할당하고 내부 구성요소를 동적으로 초기화하는데 사용

■ Closed 이벤트

- 폼이 닫히면 발생하는 이벤트
- Cancel 프로퍼티([표 7.6] 참조)의 값이 거짓일 때만 발생



Load/Closed 이벤트 [2/3]

[예제 7.2 – LoadClosedApp.cs]

1) 폼 설계



2) 프로퍼티

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

3) 이벤트 처리기

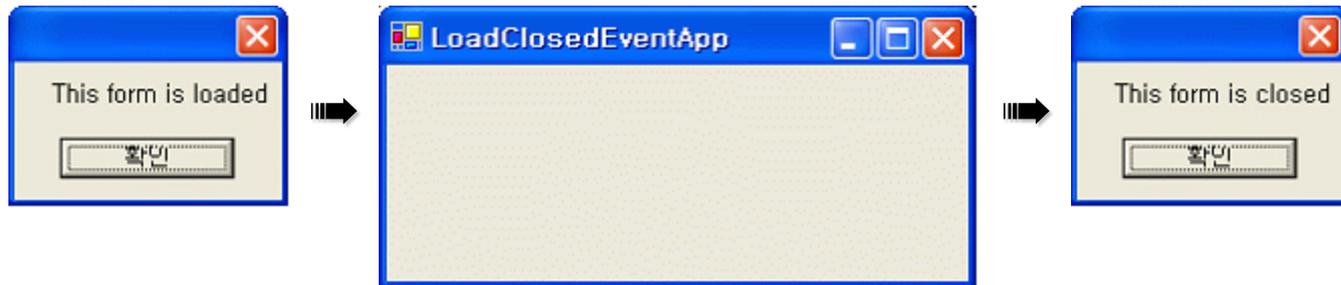
컨트롤 : (Name)	이벤트	메소드명
Form : Form1	Load	Form1_Load()
	Closed	Form1_Closed()

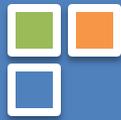
```
private void Form1_Load(object sender, EventArgs e) {
    MessageBox.Show("This form is loaded");
}
private void Form1_Closed(object sender, EventArgs e) {
    MessageBox.Show("This form is closed");
}
```



Load/Closed 이벤트 [3/3]

실행 방법 : 폼을 종료하면 폼의 종료를 알리는 메시지 상자가 나타난다.
실행 결과 :





Closing 이벤트 [1/3]

■ 용도

- 폼이 닫히는 도중에 발생

■ 매개변수

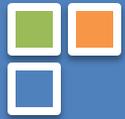
- EventArgs 형
 - Cancel 프로퍼티: 닫기 작업의 취소 여부를 설정.

■ 설명

- Cancel 프로퍼티는 닫기 작업의 취소 여부를 결정하기 위해 사용

■ 예

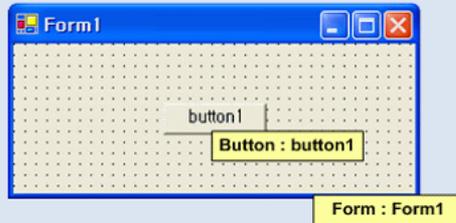
- 문서편집기에서 작업 중인 문서를 저장하지 않고 종료
- 현재 작업 중인 문서를 저장하고 종료할지, 저장하지 않고 종료할지, 또는 종료 작업을 취소할지 여부를 사용자에게 물어보는 대화상자를 표시
- 대화상자에서 종료 작업에 대한 취소를 선택하면 프로그램 종료 작업은 취소



Closing 이벤트 [2/3]

[예제 7.3 – PromptCloseApp.cs]

1) 폼 설계



2) 프로퍼티

컨트롤 : (Name)	프로퍼티	값
Form : Form1	Text	PromptCloseApp
Button : button1	Dock	Fill
	Text	Close

3) 이벤트 처리기

컨트롤 : (Name)	이벤트	메소드명
Form : Form1	Closing	Form1_Closing()
Button : button1	Click	button1_Click()

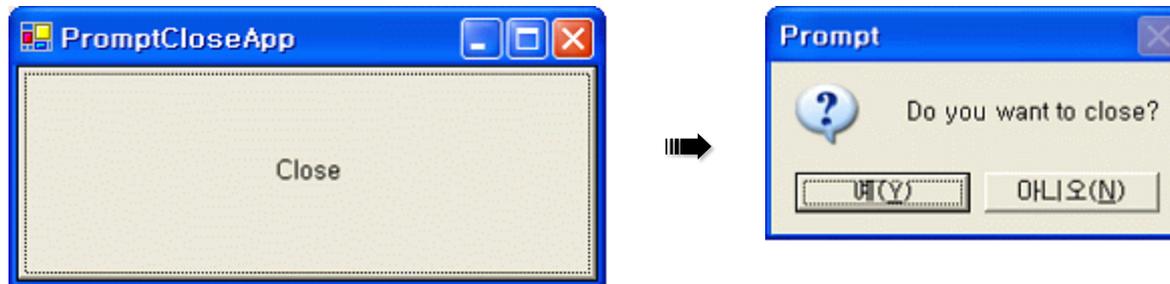


Closing 이벤트 [3/3]

```
private void Form1_Closing(object sender, CancelEventArgs e) {  
    if (MessageBox.Show("Do you want to close?", "Prompt", MessageBoxButtons.YesNo,  
        MessageBoxIcon.Question) == DialogResult.Yes)  
        e.Cancel = false;  
    else  
        e.Cancel = true;  
}  
private void button1_Click(object sender, EventArgs e) {  
    Close();  
}
```

실행 방법 : 폼의 바탕에 있는 버튼을 누르면, 폼의 닫기 여부를 물어보는 대화상자가 나타난다.

실행 결과 :





Activated/Deactivate 이벤트

- 용도
 - Activated 이벤트
 - 폼이 활성화된 직후 발생
 - Deactivate 이벤트
 - 폼이 비활성화될 때 발생

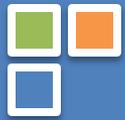


컨트롤 클래스

- 컨트롤 클래스
 - Control 클래스를 의미
 - 윈폼 애플리케이션에서 화면에 표시되어 사용자와 상호작용을 수행하는 컨트롤들을 위한 베이스 클래스
 - 폼과 폼에 배치되는 여러 컨트롤에서 공통적으로 필요한 멤버들이 정의

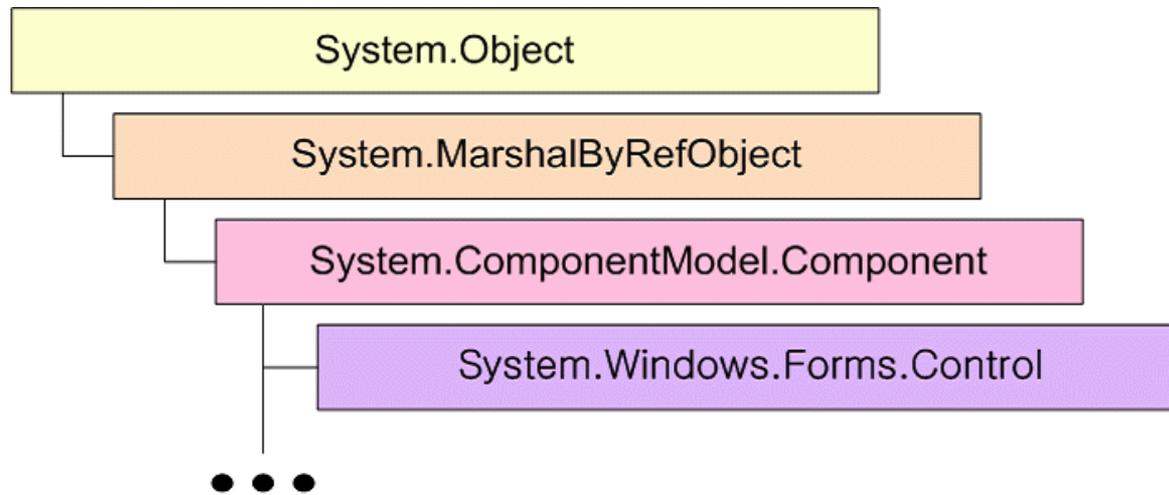
- 컴포넌트와 컨트롤
 - 컴포넌트: 화면에 자신을 그리지 않는 컴포넌트
 - 이미지 리스트나 타이머 등이 있음.
 - 폼에 표시되지 않는 컴포넌트는 Component 클래스에서 파생된 클래스

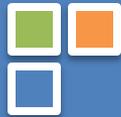
 - 컨트롤: 화면에 자신을 그릴 수 있는 컴포넌트
 - 버튼과 레이블, 체크박스 등이 있음.
 - 폼에 표시되는 컴포넌트인 컨트롤은 Control 클래스에서 파생된 클래스로 정의



컨트롤 클래스의 계층도

■ System.Windows.Forms 네임스페이스





컨트롤 클래스의 프로퍼티 [1/2]

프로퍼티	설명
Name	컨트롤의 이름으로 소스파일에서 참조할 때 사용.
Text	컨트롤에 표시되는 문자열을 설정하거나 참조.
Font	컨트롤에 표시되는 문자열의 글꼴을 설정.
ForeColor	컨트롤에 표시되는 문자열의 글자색을 설정.
BackColor	컨트롤의 배경색을 설정.
BackgroundImage	컨트롤의 배경 이미지를 설정.
Visible	컨트롤의 화면 표시 여부를 설정하거나 참조.
Enabled	컨트롤의 활성화 여부를 설정하거나 참조.
Size	컨트롤의 크기를 설정하거나 참조.
ClientSize	컨트롤에서 클라이언트 영역의 크기.
ClientRectangle	컨트롤에서 클라이언트 영역의 위치와 크기.
Dock	상위 컨트롤 내에서 컨트롤의 크기와 위치를 설정.



컨트롤 클래스의 프로퍼티 [2/2]

프로퍼티	설명
Width/Height	컨트롤의 폭과 높이를 설정하거나 참조.
Location	컨트롤의 위치를 설정하거나 참조.
Left/Top	컨트롤의 왼쪽과 위쪽 좌표를 설정하거나 참조.
Right/Bottom	컨트롤의 오른쪽/아래쪽 좌표.(읽기 전용 프로퍼티)
TabIndex	탭 키에 의한 포커스 이동 순서를 설정.
TabStop	탭 키에 의한 포커스 설정 가능 여부를 설정.
Parent	현재 컨트롤이 포함된 상위 컨트롤.
Tag	컨트롤에 연관된 객체를 설정하거나 참조.
Cursor	컨트롤에서 사용하는 커서를 설정하거나 참조.
ContextMenu	컨트롤에 해당하는 상황 메뉴를 설정.



Name 프로퍼티

- 용도
 - 컨트롤의 이름을 나타내는 프로퍼티
- 형식
 - `public string Name { get; set; }`



Text 프로퍼티

- 용도
 - 컨트롤에 표시되는 문자열을 설정하고자 할 때 사용하는 프로퍼티
- 형식
 - `public string Text { get; set; }`
- 설명
 - Form 클래스: 윈도우 폼의 제목 표시줄에 나타남.
 - Button 클래스: 버튼 위에 값이 표시
 - 클래스에 따라 다양한 방식으로 문자열이 표시됨.



Enabled/Visible 프로퍼티

■ 용도

■ Enabled 프로퍼티

- 컨트롤의 활성화 여부를 나타내는 프로퍼티
- 컨트롤이 비활성화되면, 컨트롤은 회색으로 변하며 키보드와 마우스의 입력에 반응하지 않음.

■ Visible 프로퍼티

- 컨트롤을 화면에 표시할지를 결정하는 프로퍼티
- 거짓으로 설정되면, 해당 컨트롤은 화면상에서 사라짐.

■ 형식

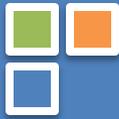
- `public bool Enabled { get; set; }`
- `public bool Visible { get; set; }`



Parent 프로퍼티

- 용도
 - 컨트롤이 다른 컨트롤에 포함되는 경우, 자신을 포함하고 있는 컨트롤을 나타내는 프로퍼티

- 형식
 - `public Control Parent {get; set;}`



Left/Right/Top/Bottom/Width/Height/Location 프로퍼티

■ 용도

- 컨트롤의 크기와 위치를 나타내기 위해서 사용되는 프로퍼티

■ 형식

- `public int Left { get; set; }`
- `public int Right { get; }`
- `public int Top { get; set; }`
- `public int Bottom { get; }`
- `public int Width { get; set; }`
- `public int Height { get; set; }`
- `public Point Location { get; set; }`

■ 설명

- 컨트롤이 다른 컨트롤에 포함되는 경우, 컨트롤을 포함하고 있는 상위 컨트롤 내에서의 상대적인 위치 값



Size/ClientSize/ClientRectangle 프로퍼티

■ 용도

- Size 프로퍼티: 컨트롤의 크기를 나타내는 프로퍼티
- ClientSize 프로퍼티: 클라이언트 영역의 크기를 나타내는 프로퍼티
- ClientRectangle 프로퍼티: 클라이언트 영역의 위치와 크기를 나타내는 프로퍼티

■ 형식

- `public Size Size {get; set;}`
- `public Size ClientSize {get; set;}`
- `public Rectangle ClientRectangle {get;}`

■ 설명

- 클라이언트 영역
 - 컨트롤 자신을 꾸미는데 필요한 영역을 제외한 나머지 영역을 의미
 - 예: 폼 클래스
 - Size 프로퍼티는 윈도우 폼의 전체 크기.
 - ClientSize 프로퍼티는 제목 표시줄을 제외한 순수한 클라이언트 영역의 크기.



Dock 프로퍼티

■ 용도

- 컨트롤을 포함하고 있는 상위 컨트롤 내에서 상대적인 위치와 크기로 설정하기 위해서 사용되는 프로퍼티

■ 형식

- `public DockStyle Dock { get; set; }`

■ DockStyle 열거형

기호상수	설명
None	컨트롤의 위치와 크기를 정하지 않음.
Top	컨트롤의 위치를 상위 컨트롤내의 위쪽에 배치.
Bottom	컨트롤의 위치를 상위 컨트롤내의 아래쪽에 배치.
Left	컨트롤의 위치를 상위 컨트롤내의 왼쪽에 배치.
Right	컨트롤의 위치를 상위 컨트롤내의 오른쪽에 배치.
Fill	컨트롤의 위치를 상위 컨트롤내의 나머지 공간에 배치.



Cursor 프로퍼티

■ 용도

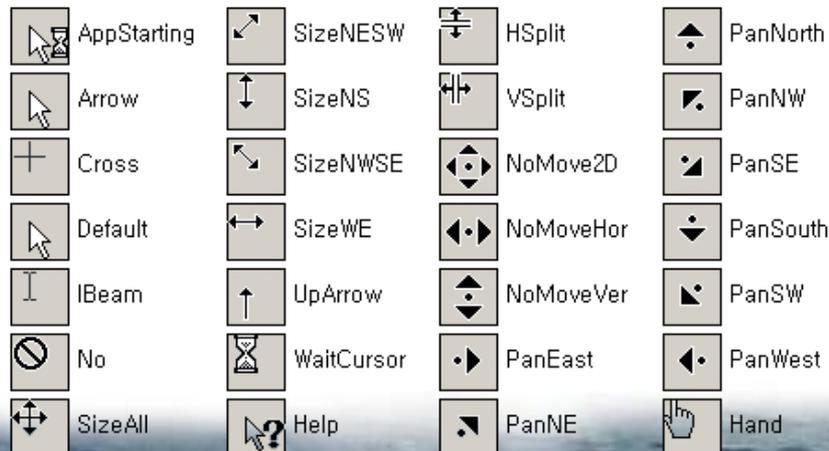
- 마우스의 커서 모양을 설정하는 프로퍼티

■ 형식

- `public Cursor Cursor {get; set;}`

■ 설명

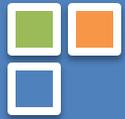
- Cursor 클래스는 커서를 표현하기 위해서 사용
- Cursors 클래스의 프로퍼티: 시스템에서 미리 정의된 커서를 나타냄.





Font 프로퍼티

- 용도
 - 컨트롤에서 사용하는 글꼴을 설정하는 프로퍼티
- 형식
 - `public Font Font { get; set; }`



BackColor/ForeColor 프로퍼티

- 용도
 - 컨트롤에서 사용하는 배경색과 전경색을 나타내는 프로퍼티

- 형식
 - `public Color BackColor { get; set; }`
 - `public Color ForeColor { get; set; }`



BackGroundImage 프로퍼티

■ 용도

- 컨트롤의 배경을 나타내는 프로퍼티

■ 형식

- `public Image BackGroundImage { get; set; }`

■ 설명

■ 폼 클래스

- 선택한 이미지의 크기가 컨트롤의 크기보다 크다면 이미지의 일부분만이 표시
- 이미지의 크기가 컨트롤의 크기보다 작다면 바둑판 형식으로 반복되어 이미지가 표시



ContextMenu 프로퍼티

■ 용도

- 컨트롤에서 마우스 오른쪽 버튼을 누를 때 나타나는 메뉴인 팝업 메뉴 (상황 메뉴)를 설정하는 프로퍼티

■ 형식

- `public ContextMenu ContextMenu {get; set;}`

■ 설명

- 상황 메뉴를 정의한 후 이 프로퍼티에 지정하면, 팝업 메뉴로 작동



TabIndex/TabStop 프로퍼티

■ 용도

■ TabIndex 프로퍼티

- 탭키를 통한 포커스를 이동할 때 포커스가 이동되는 순서를 설정하기 위해서 사용되는 프로퍼티
- 프로퍼티의 값이 작은 컨트롤에서 큰 컨트롤로 이동

■ TabStop 프로퍼티

- 탭키에 의해 포커스 이동이 필요 없을 때 값을 설정하는 프로퍼티
- 값이 참인 경우: 포커스 이동시 컨트롤이 포커스를 가짐.
- 값이 거짓인 경우: 컨트롤이 포커스를 얻지 못함.

■ 형식

- `public int TabIndex { get; set; }`
- `public bool TabStop { get; set; }`



Tag 프로퍼티

■ 용도

- 컨트롤에 필요한 값을 저장한 후 참조하기 위해서 사용되는 프로퍼티

■ 형식

- `public object Tag { get; set; }`

■ 설명

- 자료형이 `System.Object` 클래스형이기 때문에, .NET 프레임워크에서 사용하는 모든 값을 이 프로퍼티에 연결해 두었다가 필요할 때 사용할 수 있음.



컨트롤 클래스의 주요 메소드 [1/4]

메소드	설명
Show()	컨트롤을 화면에 표시한다.
Hide()	컨트롤을 화면에 표시하지 않는다.
Invalidate()	컨트롤의 영역을 무효화하여 다시 그려지도록 한다.
BringToFront()	컨트롤의 화면 표시 순서를 가장 앞으로 변경한다.
SendToBack()	컨트롤의 화면 표시 순서를 가장 뒤로 변경한다.
PointToClient()	화면상의 좌표를 컨트롤 내부의 상대 좌표로 변환한다.
PointToScreen()	컨트롤 내부의 좌표를 화면상의 좌표로 변환한다.
RectangleToClient()	사각형의 좌표를 컨트롤 내부의 상대 좌표로 변환한다.
RectangleToScreen()	사각형의 좌표를 전체 화면상의 좌표로 변환한다.
ResetXxx()	Xxx 이름을 가진 프로퍼티의 값을 초기값으로 설정한다.



컨트롤 클래스의 주요 메소드 [2/4]

■ Show() 메소드

- 컨트롤을 화면에 보이게 만드는 메소드로 Visible 프로퍼티의 값을 참으로 만듦.

- `public void Show();`

■ Hide() 메소드

- 컨트롤을 화면에서 사라지게 만드는 메소드로 Visible 프로퍼티의 값을 거짓으로 만듦.

- `public void Hide();`

■ Invalidate() 메소드

- 컨트롤의 영역을 무효화하여, 다시 화면에 그려지도록 만드는 메소드

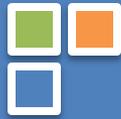
- `public void Invalidate();`

- Paint 이벤트를 발생시켜 Paint 이벤트의 처리기에서 폼이나 컨트롤에 새로운 내용을 그리는 작업을 하도록 함.



컨트롤 클래스의 주요 메소드 [3/4]

- BringToFront() 메소드
 - 해당 컨트롤을 맨 앞으로 이동시키는 메소드
 - `public void BringToFront();`
- SendToBack() 메소드
 - 해당 컨트롤을 맨 뒤로 이동시키는 메소드
 - `public void SendToBack();`
- PointToClient() 메소드
 - 전체 화면상의 좌표로 되어있는 한 점을 컨트롤의 Location 프로퍼티와의 상대 좌표로 변환하는 메소드
 - `public Point PointToClient(Point p);`
- PointToScreen() 메소드
 - 컨트롤 영역 안에 있는 한 점의 좌표를 전체 화면상의 좌표로 변환하는 메소드
 - `public Point PointToScreen(Point p);`



컨트롤 클래스의 주요 메소드 [4/4]

■ RectangleToClient() 메소드

- 전체 화면상의 영역을 이루는 사각형의 좌표를 컨트롤의 Location 프로퍼티와의 상대 좌표로 변환하는 메소드
 - `public Rectangle RectangleToClient(Rectangle r);`

■ RectangleToScreen() 메소드

- 클라이언트 영역의 상대 좌표로 되어있는 사각형을 전체 화면상의 좌표로 변환하는 메소드
 - `public Rectangle RectangleToScreen(Rectangle r);`

■ ResetXxx() 메소드

- Xxx와 일치하는 이름을 가진 프로퍼티의 값을 초기값으로 되돌리는 일을 하는 메소드
 - `public void ResetXxx();`
- Xxx 위치에 올 수 있는 프로퍼티
 - `BackColor, Bindings, Cursor, Font, ForeColor, ImeMode, RightToLeft, Text`



컨트롤 클래스의 주요 이벤트 [1/2]

■ 주요 이벤트

이벤트	설명
Click	컨트롤을 마우스로 클릭할 때 발생한다.
Move	컨트롤의 위치가 변경되었을 때 발생한다.
Paint	컨트롤을 다시 그려야 할 때 발생한다.
Resize	컨트롤의 크기가 변경되었을 때 발생한다.
XxxChanged	Xxx 이름을 가진 프로퍼티가 변경되었을 때 발생한다.



컨트롤 클래스의 주요 이벤트 [2/2]

- Paint 이벤트
 - 컨트롤을 다시 그려야 하는 경우에 발생하는 이벤트
 - PaintEventArgs 클래스형의 매개변수
 - ClipRectangle: 새로 그려야 하는 영역.
 - Graphics: 컨트롤을 그리는데 사용하는 Graphics 객체.
- Move 이벤트
 - 컨트롤의 위치가 변경되면 발생하는 이벤트
- Resize 이벤트
 - 컨트롤의 크기가 변경이 되면 발생하는 이벤트
- XxxChanged 이벤트
 - Xxx와 일치하는 이름을 가진 프로퍼티의 값이 변경되면 이벤트가 발생