Windows Forms

Клакович Л.М.

Програмування з Windows.Forms

- **`Windows.Forms`** – технології .NET для швидкої і ефективної розробки клієнтських програм
- Структура WinForms-програми
- Огляд ієрархії класів **`System.Windows.Forms`**
- **Controls**- компоненти з візуальним представленням

Windows Forms

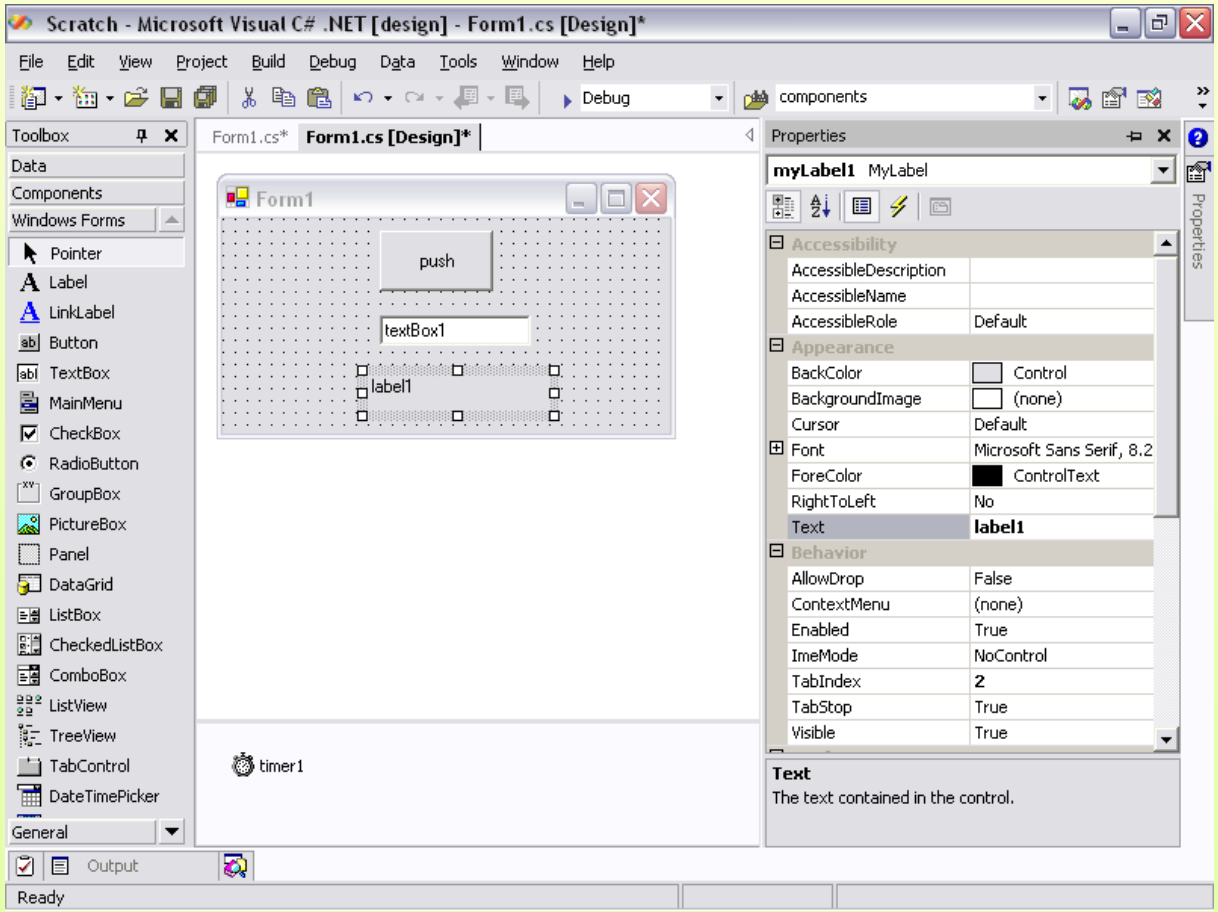
- Windows Forms є класами для побудови UIs
  - Форми (forms), контроли (controls), діалоги (dialogs), etc.
  - Частина бібліотеки .NET Framework
  - Основні класи в просторі імен **System.Windows.Forms** приховують виклики **Win32 API**

- Використовуються для побудови:
  - графічних програм для Windows (WA)
  - клієнтських частин в клієнт-серверних програмах

Visual Studio

- **Visual Studio .NET забезпечує середовище розробки**

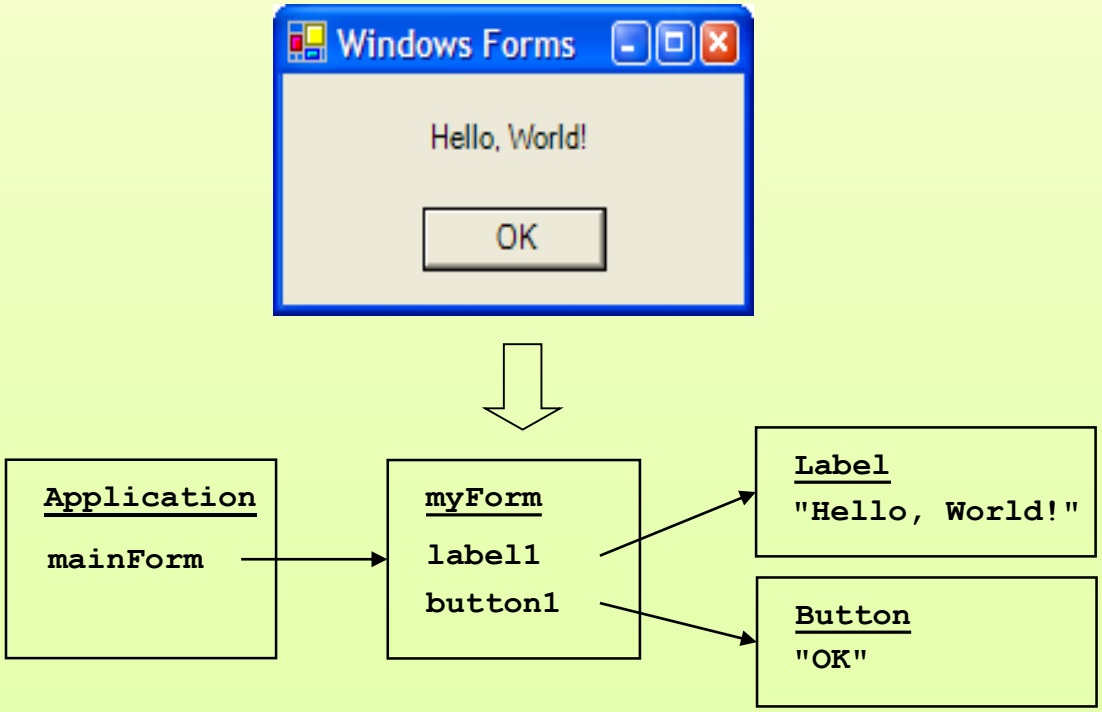
  - початкові шаблони, майстри, утиліти, drag-and-drop редагування, властивості, grid, etc.



Структура WA

Windows Forms application складається з 3 основних частин:

- application
- форма (Form)
- елементи керування (controls) і компоненти на формах



Визначення форми

```
class MyForm : Form
{
  public MyForm()
  {
    this.ShowInTaskbar = false;
    this.Location      = new Point(10, 10);
    this.Size          = new Size(100, 100);
  }
}
```

Задання властивостей

start application

```
class MyApp
{
  public static void Main()
  {
    MyForm form = new MyForm();

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

Клас **Application**

`public sealed class Application`

- **містить набір статичних методів і властивостей для керування поведінкою WA**
- **визначає набір подій рівня всієї програми.**
  - Початок та закінчення програми, простій центрального процесора
  - Метод **Run** запускає процес обробки повідомлень Windows
  - Властивості надають інформацію про WA (**CompanyName**, **StartupPath**)
  - Всі члени є статичні. Клас `Application` не наслідується.

`Application` Public Properties

Назва	Опис
CommonAppDataPath	Gets the path for the application data that is shared among all users.
CommonAppDataRegistry	Gets the registry key for the application data that is shared among all users.
CompanyName	Gets the company name associated with the application.
CurrentCulture	Gets or sets the culture information for the current thread.
CurrentInputLanguage	Gets or sets the current input language for the current thread.
ExecutablePath	Gets the path for the executable file that started the application, including the executable name.
LocalUserAppDataPath	Gets the path for the application data of a local, non-roaming user.
MessageLoop	Gets a value indicating whether a message loop exists on this thread.
OpenForms	Gets a collection of open forms owned by the application.
ProductName	Gets the product name associated with this application.
ProductVersion	Gets the product version associated with this application.
StartupPath	Gets the path for the executable file that started the application, not including the executable name.
UserAppDataPath	Gets the path for the application data of a user.
UseWaitCursor	Gets or sets whether the wait cursor is used for all open forms of the application.

`Application` Public Methods

Назва	Опис
AddMessageFilter **RemoveMessageFilter** **FilterMessage**	**Для перехоплення повідомлень з допомогою певного фільтру.**
DoEvents	Обробка всіх повідомлень з черги.
EnableVisualStyles	Забезпечує доступ до візуальних стилів.
Exit	**Завершення роботи**
ExitThread	**Завершення обробки повідомлень для даного потоку і закриває всі вікна, якими володіє цей потік.**
OleRequired	Ініціалізація бібліотеки OLE.
Restart	Закриття програми і старт її нової версії.
Run	**Запускає стандартний цикл роботи з повідомленнями для даного потоку.**
SetSuspendState	Suspends or hibernates the system, or requests that the system be suspended or hibernated.
SetUnhandledExceptionMode	Інструкція програмі як діяти у випадку неперехоплених вийнятків.

`Application` Public Events

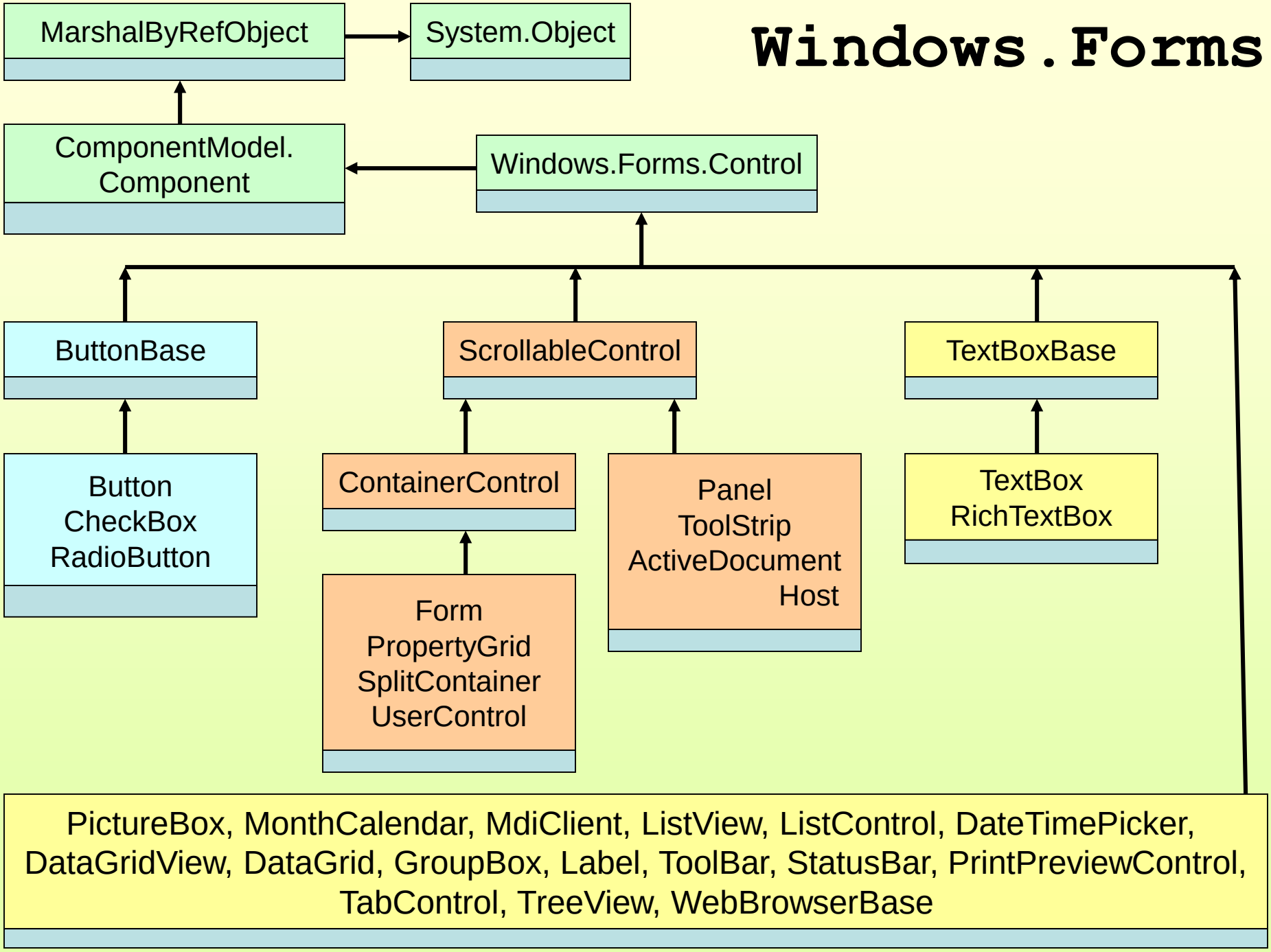
Назва	Опис
ApplicationExit	**Настає в момент закриття програми**.
EnterThreadModal	Occurs when the application is about to enter a modal state.
Idle	Настає в той момент, коли всі поточні повідомлення в черзі оброблені і програма переходить в режим бездіяльності
LeaveThreadModal	Occurs when the application is about to leave a modal state.
ThreadException	Occurs when an untrapped thread exception is thrown.
ThreadExit	Виникає при завершенні роботи потоку в програмі. Якщо роботу завершує головний потік, то подія виникає до події ApplicationExit

Форма

`System.Windows.Forms.Form`

- представляє вікно, забезпечуючи відповідні сервіси
- використовується як базовий для створення власних форм
- наслідує можливості багатьох базових класів

Windows.Forms



Клас `Component`

- **Містить реалізацію інтерфейсу IComponent**, яка включає властивість **Site**. Інтерфейс **ISite** повертає набір методів, що дають змогу елементу керування взаємодіяти з контейнером (напр. формою)
- **Містить подію Disposed,** забезпечує реалізацію методу **Dispose().** Викликається автоматично тоді, коли компонента вже не потрібна (напр. при закритті вікна). Основне призначення – звільнення ресурсів.

Клас `Control`

`Control` – компонента з візуальним представленням

Визначає загальні риси для всіх типів, що відносяться до елементів графічного інтерфейсу:

- Розмір і розташування
  - **`Height,Width,Top,Bottom,Left,Right,Size,Location`**
  - **`Bounds, ClientSize`**
  - **`Dock, Anchor`**
- Зовнішній вигляд
  - **`Font, Text`**
  - **`BackColor, ForeColor`**
  - **`BackgroundImage, BackgroundImageLayout`**
- Взаємодія
  - з користувачем (перехоплення введення з клавіатури і мишки)
  - з операційною системою (отримання HWND)
    - **`Handle, IsHandleCreated`**
    - **`protected virtual void WndProc ( ref Message m )`**

Налаштування елементів керування

- **`GetStyle(), SetStyle()`** – для отримання (встановлення) стилю, як значення переліченння **`ControlStyles`** (аналог заповнення структури WNDCLASSEX)

```
SetStyle(ControlStyles.ResizeRedraw,true)
```

- **`Hide(),Show()`** – керують станом властивості **`Visible`**
- **`Invalidate()`** змушує обновити власне зображення (недійсну область), шляхом відправки відповідного повідомлення в чергу.
- **`Refresh()`** обновлює ел. керування і всі дочірні ел. к.

Організація контролів

- відображення на видимій поверхні іншого контрола, який фігурує в

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

- набір асоційованих контролів

```
public ControlCollection Controls { get; }
```

```
this.Controls.Add(this.button1);
```

- тип `ControlCollection` реалізує `IList, ICollection, IEnumerable`

```
this.Controls.Contains(this.button1);
```

- порядок переміщень за допомогою `Tab` по асоційованих контролах, для яких `TabStop==true`

```
public int TabIndex { get; set; }
```

Клас `Form`

```
public class Form : ContainerControl
```

Типи вікон

- **SDI (Single Document Interface)**
- **MDI (Multiple Document Interface)**

  ```
  IsMdiContainer == true
  ```
- **MDI-child**: встановти `MdiParent`

  ```
  public Form MdiParent { get; set; }
  ```
- **Діалогові вікна** (модальні)

Супровід форм

- Події за час існування об'єкта форми
  - виклик конструктора
  - **`Load`** (форма існує, але ще невидима)
  - **`Activated`** (форма видима і доступна як поточна)
  - **`Closing`** (спроба закриття, CancelEventArgs.Cancel=true - зупиняє закриття)
  - **`Closed`** (після закриття форми, звільнення ресурсів)
  - **`Deactivated`**

Form properties

- **Form містить багато властивостей**
  - Використовуються для зміни вигляду і поведінки

```
public class Form : ContainerControl
{
  public IButtonControl AcceptButton { get; set; }
  public IButtonControl CancelButton { get; set; }
  public bool           HelpButton   { get; set; }

  public Icon   Icon { get; set; }
  public String Text { get; set; }

  public Size MaximumSize { get; set; }
  public Size MinimumSize { get; set; }

  public MainMenu Menu { get; set; }

  public bool ShowInTaskbar { get; set; }

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

Форма як контейнер контролів

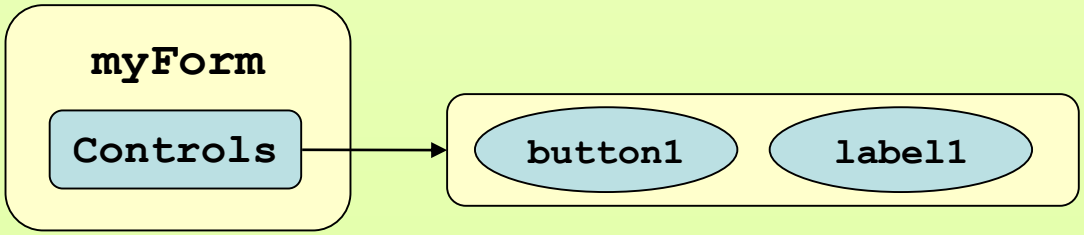
- **Форма керує колекцією контролів**
  - властивість `Controls`

Всі форми наслідують **Колекцію ел. К.**

Додати до колекції

```
class MyForm : Form
{
  Button button1 = new Button();
  Label  label1  = new Label ();

  public MyForm()
  {
    ...
    this.Controls.Add(button1);
    this.Controls.Add(label1 );
  }
  ...
}
```



Form border

- **Форми можуть виглядати як вікна, діалоги або інструменти**
  - Використовуючи властивість **`FormBorderStyle`** для встановлення або отримання стилю рамки.

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

Fixed3D, FixedDialog, FixedSingle, FixedToolWindow, None, Sizable, SizableToolWindow

Показ форм

- `public void Show ();` (`Visible ==true`)
- `public void Hide ();` (`Visible ==false`)

- `public DialogResult ShowDialog ();`

Member name	Description
Abort	The dialog box return value is **Abort** (usually sent from a button labeled Abort).
Cancel	The dialog box return value is **Cancel** (usually sent from a button labeled Cancel).
Ignore	The dialog box return value is **Ignore** (usually sent from a button labeled Ignore).
No	The dialog box return value is **No** (usually sent from a button labeled No).
None	**Nothing** is returned from the dialog box. This means that the modal dialog continues running.
OK	The dialog box return value is **OK** (usually sent from a button labeled OK).
Retry	The dialog box return value is **Retry** (usually sent from a button labeled Retry).
Yes	The dialog box return value is **Yes** (usually sent from a button labeled Yes).

class Phone

```
public partial class Phone : Form
{
    public Phone()
    {
        InitializeComponent();
    }
    public String PhoneNumber
    {
        get { return textBox1.Text; }
        set { textBox1.Text = value; }
    }

}
```

Phone.InitializeComponent()

```
    // Phone
this.AutoScaleDimensions =
           new System.Drawing.SizeF(6F,13F);
this.AutoScaleMode =
           System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(322, 243);
this.ControlBox = false;
this.Controls.Add(this.label1);
this.Controls.Add(this.textBox1);
this.Controls.Add(this.btnCancel);
this.Controls.Add(this.btnOK);
this.FormBorderStyle =
     System.Windows.Forms.FormBorderStyle.FixedDialog;
this.MaximizeBox = false;
this.MinimizeBox = false;
this.Name = "Phone";
this.ShowInTaskbar = false;
this.Text = "Phone Number";
```

Phone.InitializeComponent()

```
// btnOK
  this.btnOK.DialogResult = System.Windows.Forms.DialogResult.OK;
  this.btnOK.Location = new System.Drawing.Point(75, 194);
  this.btnOK.Name = "btnOK";
  this.btnOK.Size = new System.Drawing.Size(75, 23);
  this.btnOK.TabIndex = 1;
  this.btnOK.Text = "OK";
  this.btnOK.UseVisualStyleBackColor = true;
// btnCancel
  this.btnCancel.DialogResult =
              System.Windows.Forms.DialogResult.Cancel;
  this.btnCancel.Location = new System.Drawing.Point(180, 194);
  this.btnCancel.Name = "btnCancel";
  this.btnCancel.Size = new System.Drawing.Size(75, 23);
  this.btnCancel.TabIndex = 2;
  this.btnCancel.Text = "Cancel";
  this.btnCancel.UseVisualStyleBackColor = true;
```

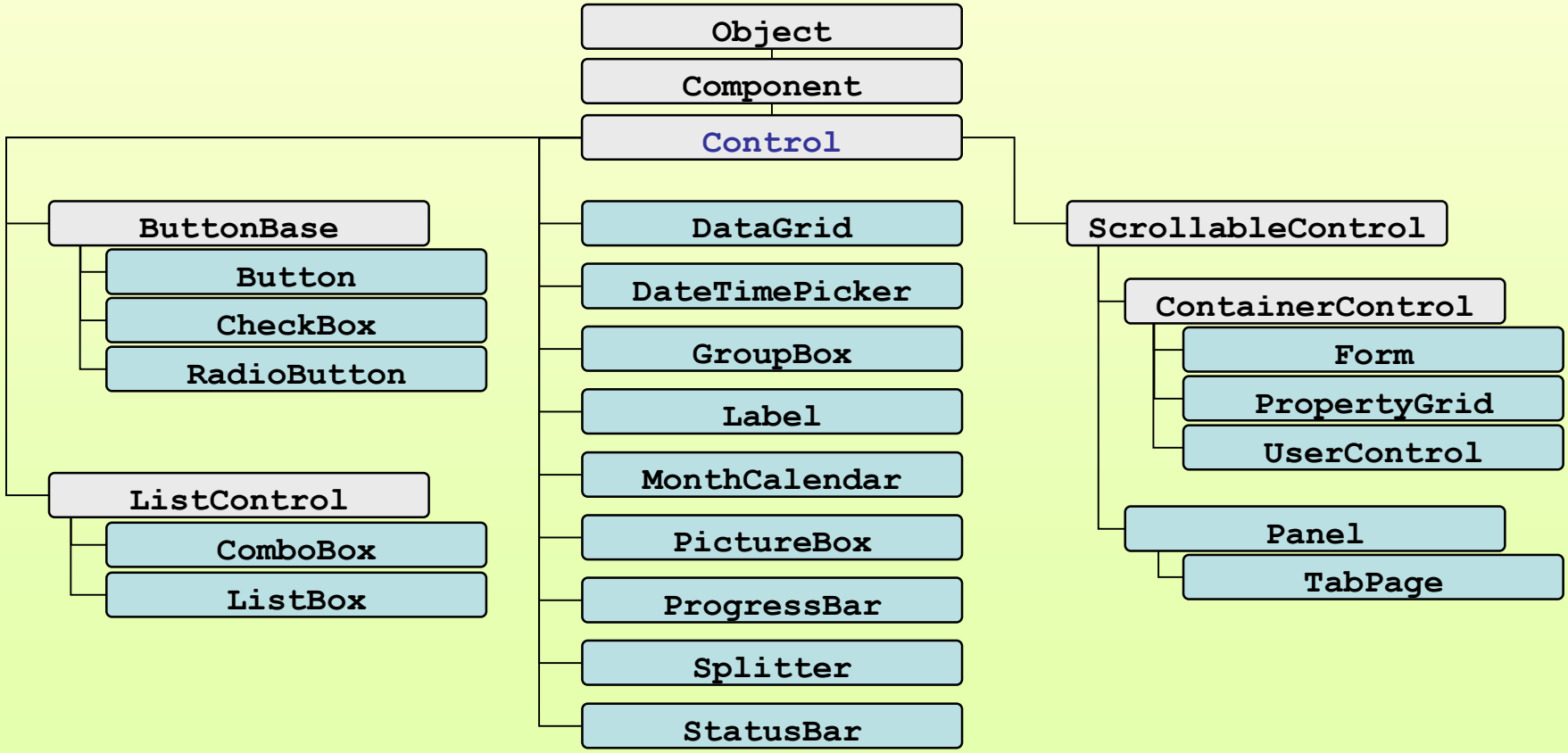
Модальний діалог

```
private void button2_Click(object sender, EventArgs e)
{
    Phone frmPhone = new Phone();
    frmPhone.ShowDialog();

    if (frmPhone.DialogResult == DialogResult.OK)
    {
        label1.Text = frmPhone.PhoneNumber;
    }
    else if (frmPhone.DialogResult == DialogResult.Cancel)
    {
        label1.Text = "Canceled";
    }
    frmPhone.Close();
}
```

Елементи керування (controls)

- **Елементи керування** – це візуальні компоненти, які є похідними від класу Control



Події елементів керування

Події елементів керування:

- **Click** на елементі керування
- події мишки та клавіатури (**MouseUp, …)**
- зміна значень властивостей (**AutoSizeChanged, FontChanged**,…)
- перемальовки (**Invalidated, Paint**)
- etc. (**Resize, LostFocus, ...)**

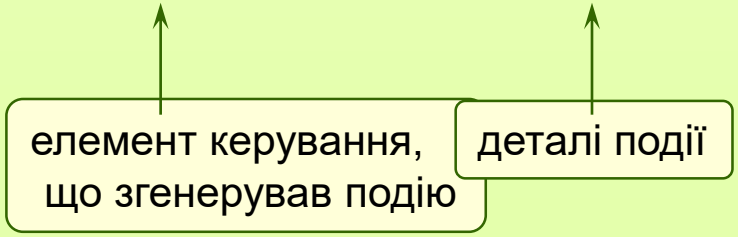
події →

```
public class Control : Component ...
{
  public event EventHandler          Click;
  public event EventHandler          Enter;
  public event EventHandler          TextChanged;
  public event KeyPressEventHandler  KeyPress;
  public event MouseEventHandler     MouseDown;
  public event PaintEventHandler     Paint;
  ...
}
```

Обробка подій (1)

- Є декілька типів делегатів для обробки подій:
  - **EventHandler** – основний, використовується для багатьох подій
  - специфічні типи для деяких подій, наприклад мишки чи клавіатури (**MouseEventHandler, KeyPressEventHandler** і ін.)
- Аргументи делегатів:
  - **object** - елемент керування, що згенерував подію
  - **EventArgs** або похідний від нього - містить деталі події

```
delegate void EventHandler         (object sender, EventArgs         e);
delegate void MouseEventHandler    (object sender, MouseEventArgs    e);
delegate void KeyPressEventHandler (object sender, KeyPressEventArgs e);
```



EventArgs

- Обробітники подій передають деталі подій через об'єкт типу **EventArgs** або похідного типу

Основний базовий клас: без даних

```
public class EventArgs
{
}
```

Інформація специфічна для подій мишки

```
class MouseEventArgs : EventArgs
{
  public MouseButtons Button { get; }
  public int          Clicks { get; }
  public int          X      { get; }
  public int          Y      { get; }
  ...
}
```

Інформація специфічна для подій клавіатури

```
class KeyPressEventArgs : EventArgs
{
  public char KeyChar { get; }
  ...
}
```

Підписка на події

- Клієнт може підписатися на події, виконавши такі кроки:

  1) визначити метод із сигнатурою, вказаною делегатом

  2) створити об'єкт делегату і додати до підписки

Визначити метод

підписка

```
public class MyForm : Form
{
  private Button button1;

  void button1_Click(object sender, EventArgs args)
  {
    ...
  }

  public MyForm()
  {
    ...
    button1.Click += new EventHandler(this.button1_Click);
    ...
  }
}
```

Обробка подій (2)

- В типах, похідних від **`Control`** – через заміщення відповідних методів базового класу: **`OnXXX()`**

protected virtual void **OnMouseUp** (***MouseEventArgs*** *e*)
protected virtual void **OnKeyDown** (***KeyEventArgs*** *e*)
protected virtual void **OnGotFocus** (***EventArgs*** *e*)
protected virtual void **OnPaint** (***PaintEventArgs*** *e*)

- Ці методи викликаються автоматично, коли відбувається дана подія (**MouseUp, KeyDown**,…)

Перевизначення методу

Підписка не потрібна

```
public class MyForm : Form
{
   private Button button1;

   protected override void OnClick( EventArgs args)
   {
      ...
   }

   public MyForm()
   {
      ...

      ...
   }
}
```

Повідомлення і події перемальовки

```
protected override void OnPaint ( PaintEventArgs e );

public event PaintEventHandler Paint ;
private void FormName_Paint(object sender, PaintEventArgs e);
```

```
class PaintEventArgs : EventArgs
{
  public Rectangle ClipRectangle { get; }
    public Graphics Graphics { get; }
  ...
}
```

ClipRectangle	надає прямокутник для малювання
Graphics	надає об’єкт Graphics для малювання.

Paint & OnPaint

```
protected override void OnPaint( PaintEventArgs e)
{
        base.OnPaint(e);
        Graphics dc = e.Graphics;
        Pen bluePen = new Pen(Color.Blue, 5);
        dc.DrawRectangle(bluePen, 0, 0, 100, 100);
}

private void MainForm_Paint(object sender, PaintEventArgs e)
{
        Graphics dc = e.Graphics;
        Pen redPen = new Pen(Color.Red, 5);
        dc.DrawRectangle(redPen, 200, 0, 100, 100);
}
```

`System.Drawing` Namespace

`Bitmap`	Encapsulates a GDI+ bitmap, which consists of the pixel data for a graphics image and its attributes. A Bitmap is an object used to work with images defined by pixel data.
`Brush`	Defines objects used to fill the interiors of graphical shapes such as rectangles, ellipses, pies, polygons, and paths.
`Brushes`	Brushes for all the standard colors. This class cannot be inherited.
`Font`	Defines a particular format for text, including font face, size, and style attributes. This class cannot be inherited.
`FontConverter`	Converts Font objects from one data type to another.
`FontFamily`	Defines a group of type faces having a similar basic design and certain variations in styles. This class cannot be inherited.
`Graphics`	Encapsulates a GDI+ drawing surface. This class cannot be inherited.
`Icon`	Represents a Windows icon, which is a small bitmap image used to represent an object. Icons can be thought of as transparent bitmaps, although their size is determined by the system.
`Image`	An abstract base class that provides functionality for the **Bitmap** and Metafile descended classes.
`ImageAnimator`	Animates an image that has time-based frames.
`Pen`	Defines an object used to draw lines and curves. This class cannot be inherited.
`Pens`	Pens for all the standard colors. This class cannot be inherited.
`Region`	Describes the interior of a graphics shape composed of rectangles and paths. This class cannot be inherited.
`SolidBrush`	Defines a brush of a single color. Brushes are used to fill graphics shapes, such as rectangles, ellipses, pies, polygons, and paths. This class cannot be inherited.
`StringFormat`	Encapsulates text layout information (such as alignment, orientation and tab stops) display manipulations (such as ellipsis insertion and national digit substitution) and OpenType features. This class cannot be inherited.
`SystemBrushes`	Each property of the SystemBrushes class is a SolidBrush that is the color of a Windows display element.
`SystemColors`	Each property of the SystemColors class is a **Color** structure that is the color of a Windows display element.
`SystemFonts`	Specifies the fonts used to display text in Windows display elements.

Graphics: Public Methods

DrawLine	Overloaded. Draws a line connecting the two points specified by the coordinate pairs.
DrawLines	Overloaded. Draws a series of line segments that connect an array of **Point** structures.
DrawPath	Draws a GraphicsPath.
DrawPie	Overloaded. Draws a pie shape defined by an ellipse specified by a coordinate pair, a width, a height, and two radial lines.
DrawPolygon	Overloaded. Draws a polygon defined by an array of **Point** structures.
DrawRectangle	Overloaded. Draws a rectangle specified by a coordinate pair, a width, and a height.
DrawRectangles	Overloaded. Draws a series of rectangles specified by Rectangle structures.
DrawString	Overloaded. Draws the specified text string at the specified location with the specified Brush and Font objects.
EndContainer	Closes the current graphics container and restores the state of this **Graphics** to the state saved by a call to the BeginContainer method.
EnumerateMetafile	Overloaded. Sends the records in the specified **Metafile**, one at a time, to a callback method for display at a specified point.
Equals	Overloaded. Determines whether two Object instances are equal. (Inherited from Object.)
ExcludeClip	Overloaded. Updates the clip region of this **Graphics** to exclude the area specified by a **Rectangle** structure.
FillClosedCurve	Overloaded. Fills the interior a closed cardinal spline curve defined by an array of **Point** structures.