

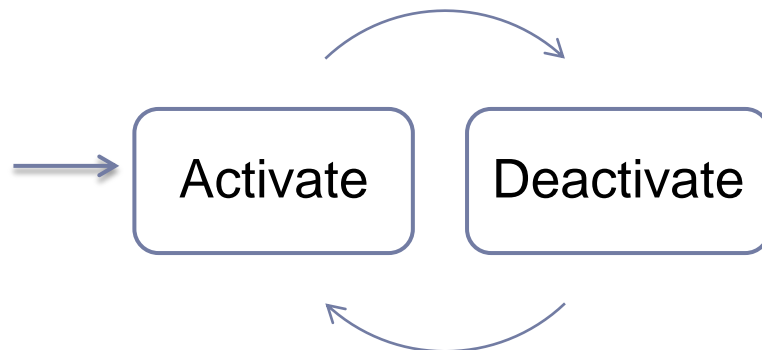
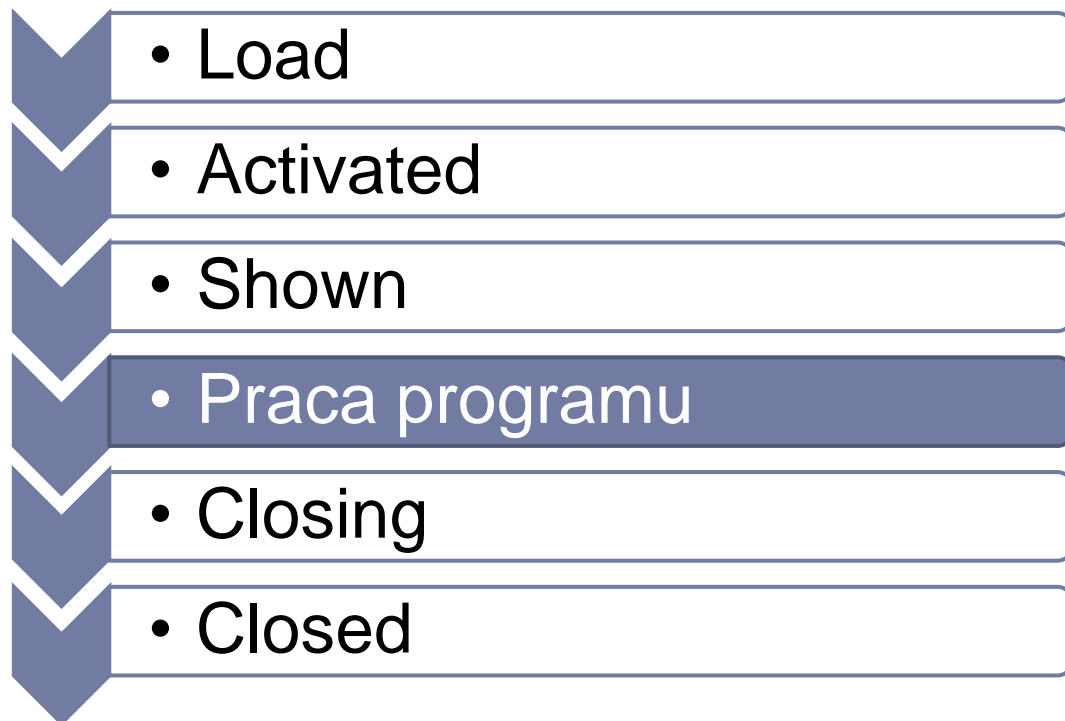
## Wykład 3

- **Cykl życia aplikacji** (zdarzenia związane z cyklem życia)
- **Zdarzenia myszy i klawiatury**
- **Klasa Timer**



## Cykl życia aplikacji

## Cykl życia programu



# Cykl życia programu

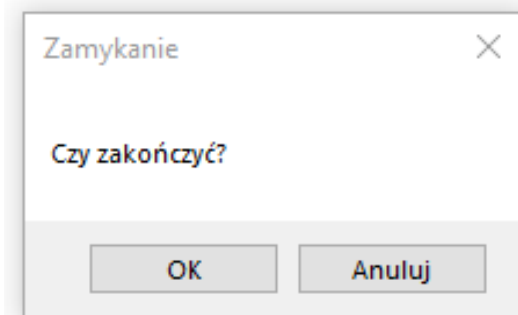
Zdarzenia dotyczą całego okna – nie poszczególnych komponentów.

<b>Behavior</b>	
ChangeUICues	
ControlAdded	
ControlRemoved	
FormClosed	Form1_FormClosed
FormClosing	Form1_FormClosing
HelpButtonClicked	
HelpRequested	
ImeModeChanged	
InputLanguageChanged	
InputLanguageChanging	
Load	Form1_Load
QueryAccessibilityHelp	
Shown	Form1_Shown
StyleChanged	
SystemColorsChanged	
<b>Data</b>	
(DataBindings)	
<b>Drag Drop</b>	
DragDrop	
DragEnter	
DragLeave	
DragOver	
GiveFeedback	
QueryContinueDrag	
<b>Focus</b>	
Activated	Form1_Activated
Deactivate	Form1_Deactivate

## Zdarzenia

### Przykład:

Potwierdzenie zamknięcia programu



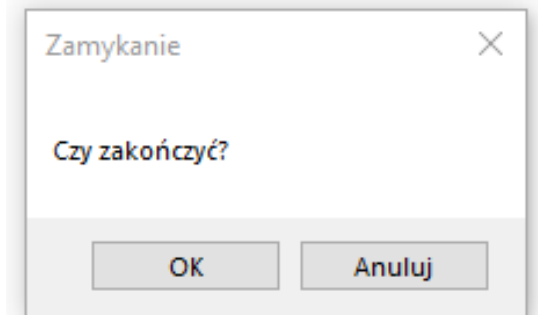
```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (MessageBox.Show("Czy zakończyć?", "Zamykanie",
        MessageBoxButtons.OKCancel) == DialogResult.Cancel)
    {
        e.Cancel = true;
    }
}
```

## Zdarzenia

---

### Przykład:

Potwierdzenie zamknięcia programu



```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (MessageBox.Show("Czy zakończyć?", "Zamykanie",
        MessageBoxButtons.OKCancel) == DialogResult.Cancel)
    {
        e.Cancel = true;
    }
}
```



## Wygląd / rozmiar / położenie aplikacji

# Zdarzenia

## [-] Action

Click	
DoubleClick	
MouseCaptureChanged	
MouseDown	
MouseDoubleClick	
ResizeBegin	
ResizeEnd	
Scroll	

Początek zmiany rozmiaru okna/ zakończenie zmiany rozmiaru

Zdarzenie wywoływane podczas przesuwania okna

## [-] Layout

Layout	
MdiChildActivate	
Move	
PaddingChanged	
Resize	

Zdarzenie wywoływane podczas zmiany rozmiaru (w sposób ciągły)

Zdarzenie wywoływane podczas przerysowania okna lub jego fragmentu

## [-] Appearance

Paint	
-------	--



# Zdarzenia

## Przykład:

Wyświetlenie aktualnej pozycji lub rozmiaru okna na pasku statusu

```
private void Form1_Move(object sender, EventArgs e)
{
    statusStrip1.Items[0].Text =
        Form1.ActiveForm.Location.X.ToString() + " " +
        Form1.ActiveForm.Location.Y.ToString();
}
```

Pozycja

Rozmiar

```
private void Form1_Resize(object sender, EventArgs e)
{
    statusStrip1.Items[0].Text =
        Form1.ActiveForm.Width.ToString() + " " +
        Form1.ActiveForm.Height.ToString();
}
```



## Zdarzenia myszy

## Zdarzenia myszy

Action	
Click	Form1_Click
DoubleClick	Form1_DoubleClick
MouseCaptureChanged	
MouseClicked	Form1_MouseClick
MouseDoubleClick	Form1_MouseDoubleClick
ResizeBegin	

Zdarzenia **Click** i **DoubleClick** można (na niektórych obiektach) wywołać za pomocą entera.

Zdarzenia **MouseClicked** i **MouseDoubleClick** są związane tylko z myszą





## Zdarzenia myszy

### ☐ Mouse

<code>MouseDown</code>	- Naciśnięcie przycisku myszy
<code>MouseEnter</code>	- Wjechanie kursora na obiekt
<code>MouseHover</code>	- Wjechanie i zatrzymanie kursora na obiekcie
<code>MouseLeave</code>	- Opuszczanie obiektu przez kursor
<code>MouseMove</code>	- Poruszanie się kursora nad obiektem (zdarzenie ciągłe)
<code>MouseUp</code>	- Puszczanie przycisku myszy

Informacje o zdarzeniu – w szczególności:

- który przycisk został naciśnięty,
- jaka jest pozycja myszy

można wyodrębnić z obiektu „e”

```
if (e.Button == MouseButton.Left)
    richTextBox1.AppendText(e.X.ToString());
```



## Zdarzenia Klawiatury



## Zdarzenia Klawiatury

---

### [-] **Key**

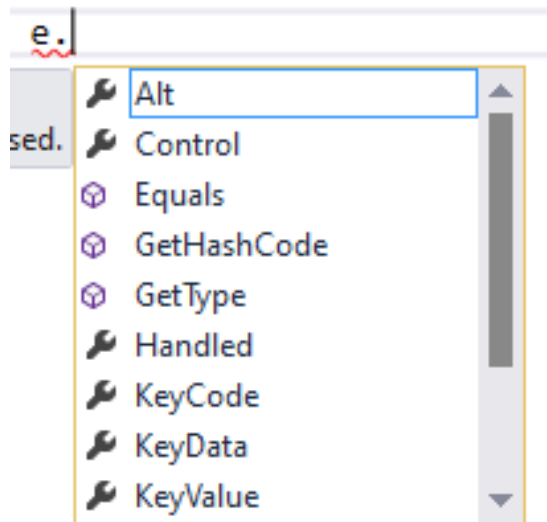
<code>KeyDown</code>	- Naciśnięcie klawisza
<code>KeyPress</code>	- Pełny cykl (naciśnięcie + zwolnienie)
<code>KeyUp</code>	- Zwolnienie klawisza
<code>PreviewKeyDown</code>	- Obsługa klawiszy modyfikatorów (Alt, Ctrl, Shift)

# Zdarzenia Klawiatury

Odczytanie wartości klawisza z metody **KeyPress**

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    char klawisz = e.KeyChar;
}
```

W metodzie **KeyDown** mamy nieco więcej możliwości



Możemy między innymi odczytać klawisze modyfikatorów.

## Zdarzenia Klawiatury

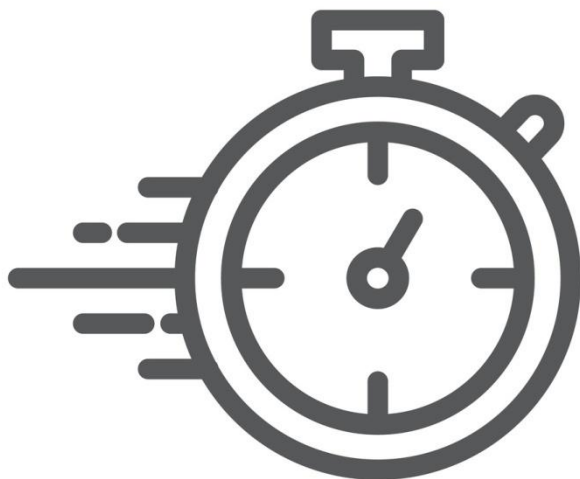
- ✓ Każdy element (kontrolka) ma swoją własną kolejkę zdarzeń klawiatury.
- ✓ Zdarzenie przechwytuje ta kontrolka, która posiada „fokus”
- ✓ Aby zdarzenie zostało dodatkowo przechwycone przez okno główne (Form1) – należy zmienić jego właściwość **KeyPreview** na **true**

☐ Misc	
AcceptButton	(none)
CancelButton	(none)
KeyPreview	<b>True</b>



## Timer

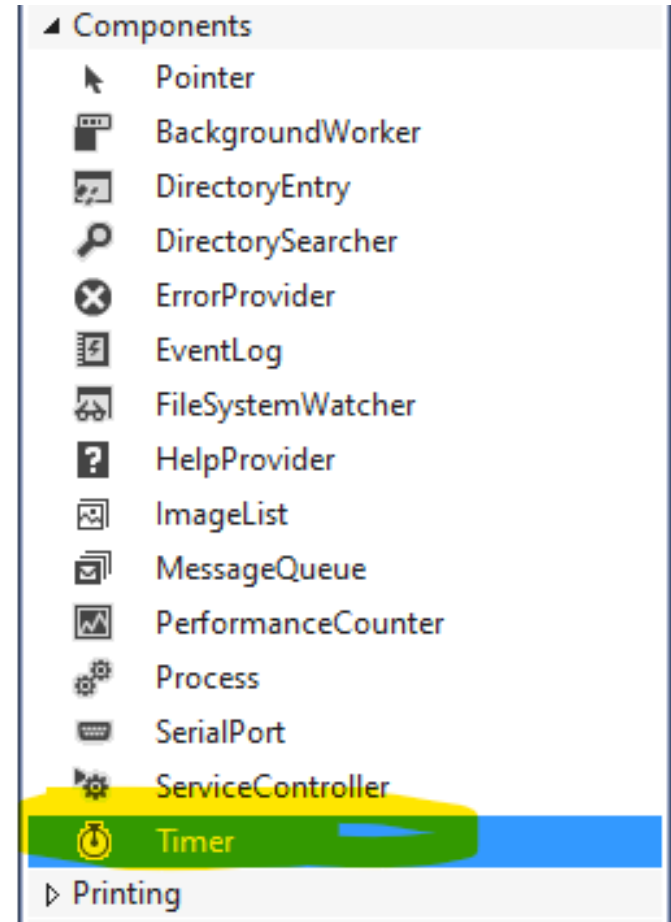
Timer jest wątkiem pracującym w tle który co pewien czas wywołuje jakąś akcję



# Timer

Rozpocząć należy od dodania komponentu **Timer** do projektu.

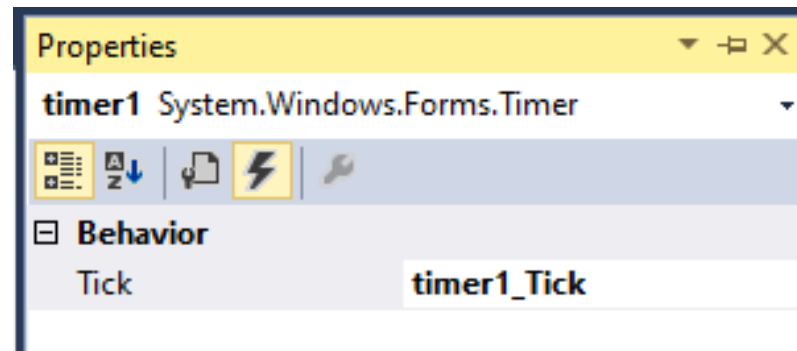
Utworzony zostanie  
instancja tej klasy –  
w naszym  
przykładzie - obiekt  
timer1



# Timer

Timer posiada metodę **Tick** która wywoływana jest co określony interwał czasu.

Oprogramowanie timera to właściwie oprogramowanie tej metody



```
private void timer1_Tick(object sender, EventArgs e)
{
    //to co mamy cyklicznie wykonać
}
```

# Timer

Ustawienia timera:

- **Enabled** – timer włączony / zatrzymany
- **Interval** – odstęp pomiędzy wywołaniami metody Tick – w milisekundach)

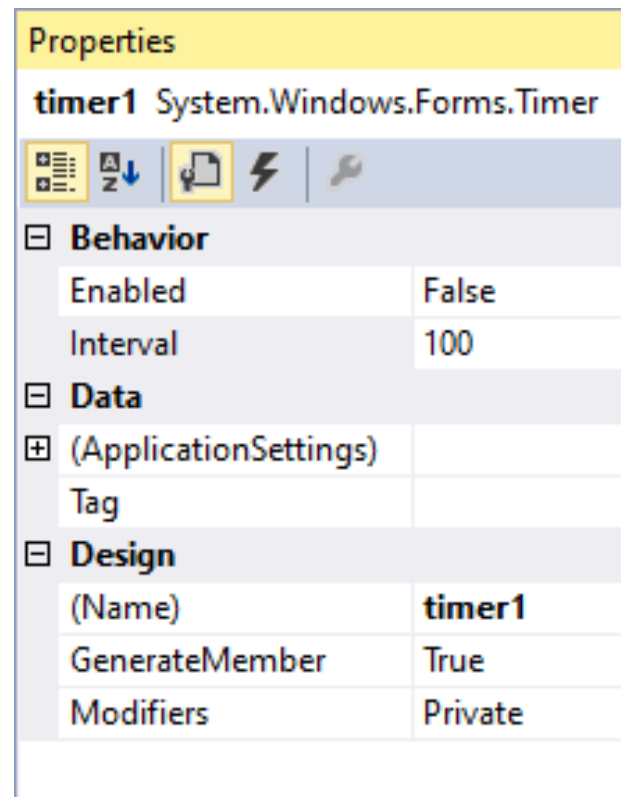
Z poziomu kodu:

- Uruchomienie timera

```
timer1.Enabled = true;
```

- Zmiana interwału

```
timer1.Interval = 100;
```



Properties

**timer1** System.Windows.Forms.Timer

Behavior

Enabled	False
Interval	100

Data

(ApplicationSettings)	
Tag	

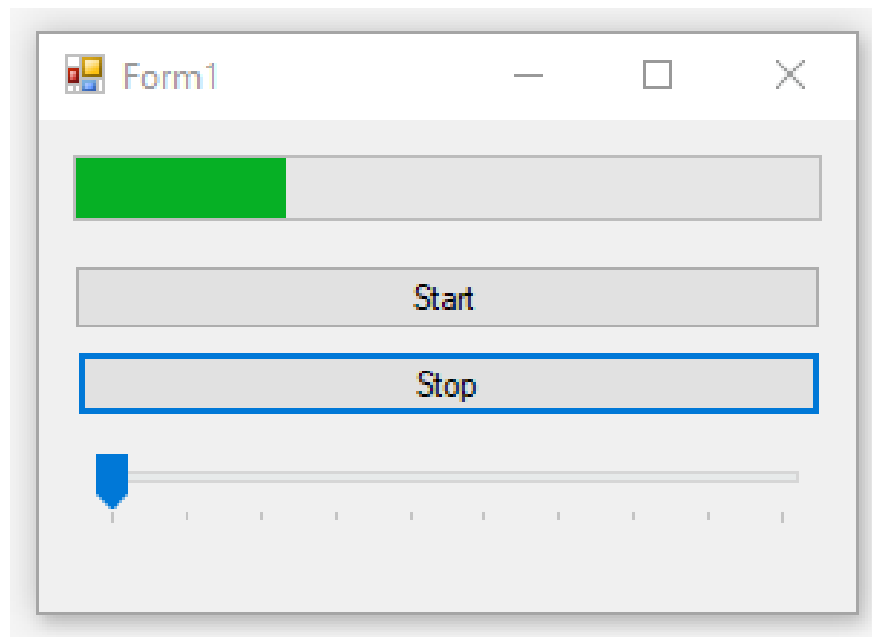
Design

(Name)	<b>timer1</b>
GenerateMember	True
Modifiers	Private

# Timer

## PRZYKŁAD:

Sterowanie paskiem postępu za pomocą timera, z regulacją prędkości.



# Timer

```
private void button1_Click(object sender, EventArgs e)
{
    timer1.Enabled = true;
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    timer1.Enabled = false;
}
```

Uruchomienie i zatrzymanie Timera



# Timer

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (progressBar1.Value >= progressBar1.Maximum)
    {
        progressBar1.Value = 0;
    }
    progressBar1.Value = progressBar1.Value + 1;
}
```

Metoda **Tick** – zwiększa wartość kontrolki progressBar  
(pilnuje też, żeby nie nastąpiło jej przepełnienie)



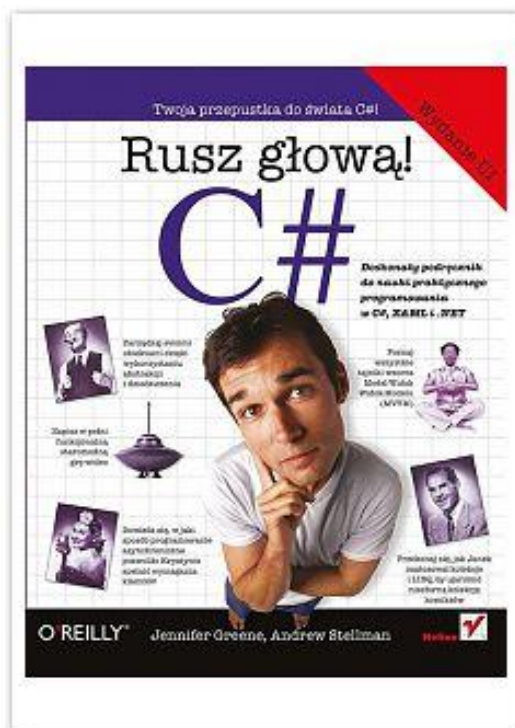
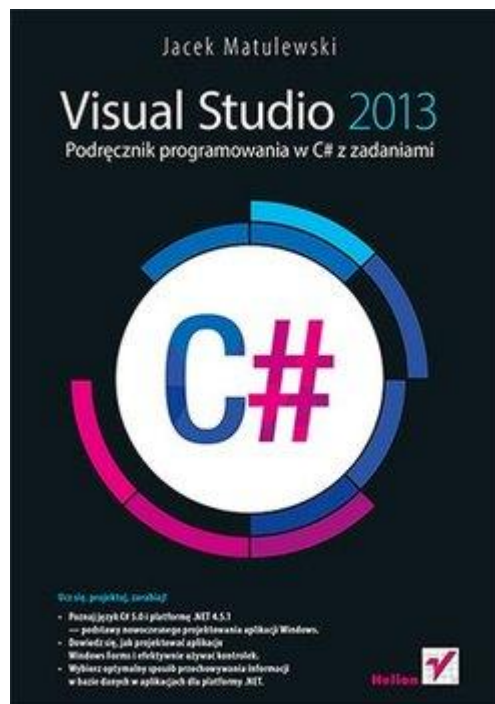
## Timer

---

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
    timer1.Interval = 1000 - trackBar1.Value *100;
}
```

Sterowanie „prędkością” timera – czyli  
ostępami pomiędzy wykonaniami metody  
Tick





Użyte w tej prezentacji tabelki pochodzą z książki: Visual Studio 2013. Podręcznik programowania w C# z zadaniami Autor: Matulewski Jacek, Helion