Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Platforma .NET

Laboratorium

Tworzenie własnych kontrolek dla Windows Forms Application (custom control).

Ćwiczenie ma za zadanie zademonstrowanie tworzenie bilbioteki własnych obiektów możliwych do umieszczenia na formularzu aplikacji okienkowej platformy .NET. W trakcie ćwiczenia wykonany zostanie obiekt spełniający rolę zegara analogowego. Dodatkowo ćwiczenie obejmuję również zagadnienia związane z tworzeniem wątków oraz korzystaniem z funkcji graficznych.

Przebieg ćwiczenia:

1. Utwórz nowy projekt
a. Z menu **File** wybierz **New/Project...**

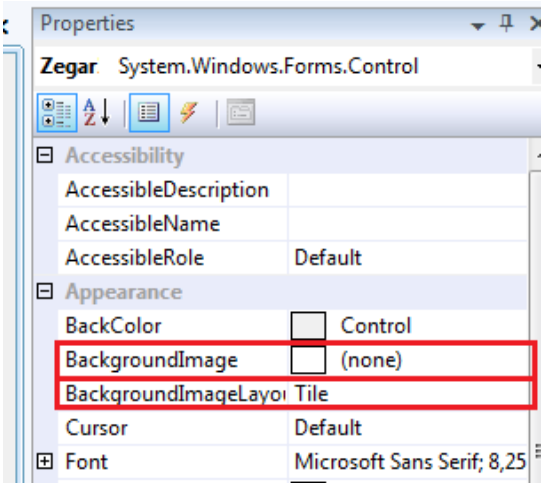
b. W oknie dialogowym **New Project** określ następujące właściwości**:**
- typu projektu: **Visual C#/Windows**
- szablon: **Class Library**
- lokalizacja: **D:\nr_grupy_nazwisko\visual studio projects**
- nazwa projektu: **Lab7_Zegar.**
- nazwa rozwiązania: **DotNETLab.**

2. Z projektu należy usunąć domyślnie dodaną pustą klasę **Class1.cs.**

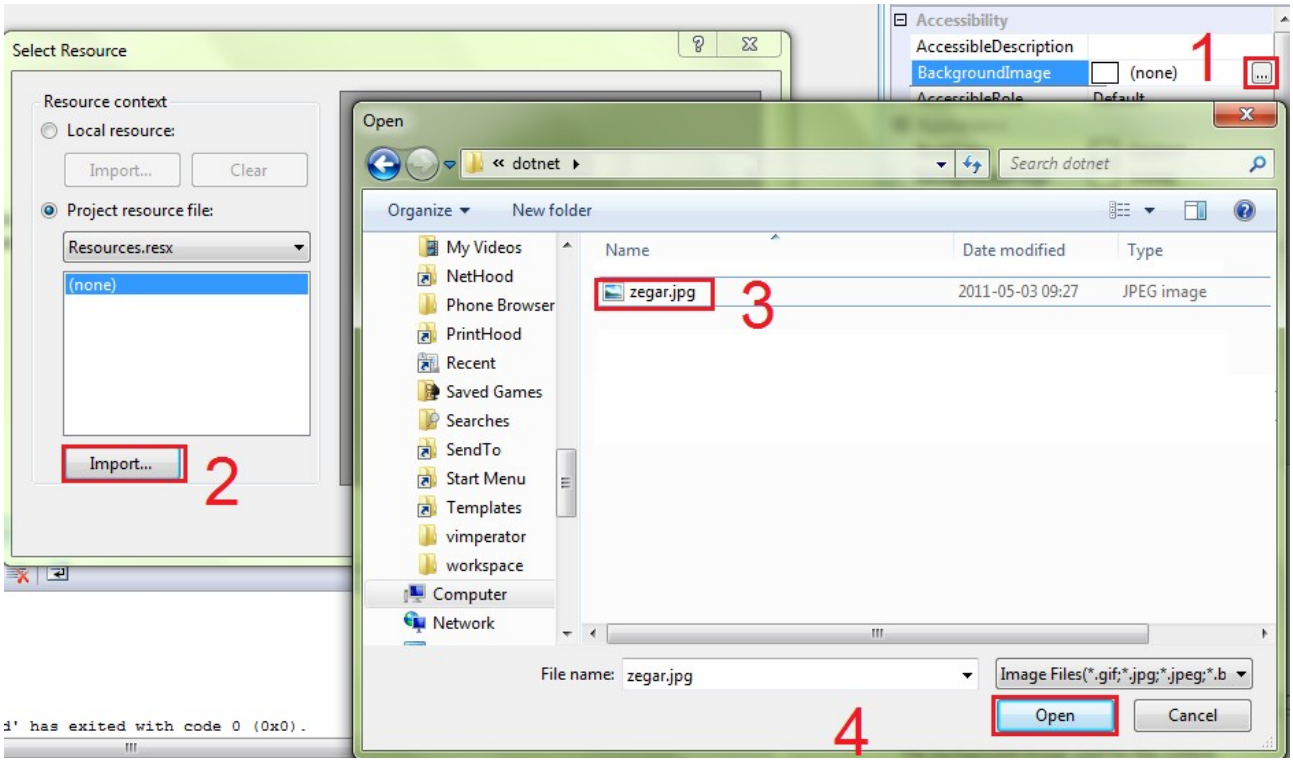
3. Do projektu należy dodać nowy obiekt poprzez kliknięcie prawym klawiszem myszki na nowym projekcie i wybranie **ADD->New Item.**

4. Z listy należy wybrać **Custom Control,** nowy obiekt należy nazwać **Zegar.**

5. Wtrybie **[Design]** obiektu **Zegar,** w oknie wałaściwości należy dodać obraz tła dla obiektu zegara, a dla pola **BackgroundImageLayout** wybrać opcję **Zoom**:



– Obraz należy załadować z dysku, z pliku typu graficznego :



6. Przekopiować i przeanalizować kod dla kontrolki pełniącej ciało zegara:
plik **Zegar.cs** :

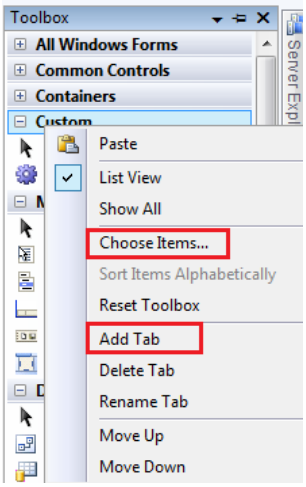
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading; // biblioteka dodana dla obsługi wątków

namespace Lab7_Zegar
{
    public partial class Zegar : Control
    {
        int h=10, m=9, s=36;
        private Color h_color = Color.LightGreen; // kolor domyślny
        private Color m_color = Color.LightGreen; // kolor domyślny
        private Color s_color = Color.LightGreen; // kolor domyślny

        //Dodawanie właściwości
        public Color HourTick
        {
            get { return h_color; }
            set { h_color = value; Invalidate(); }
        }
        public Color MinuteTick
        {
            get { return m_color; }
            set { m_color = value; Invalidate(); }
        }
        public Color SecondTick
```

```
        {
            get { return s_color; }
            set { s_color = value; Invalidate(); }
        }
        // Ukrywanie właściwości odziedziczonych po Control
        [Browsable(false)]
        public new Font Font { get ; set; }

        // Obsługa tykania
        private volatile bool _run = true;
        private void Ticking()
        {
            while (_run)
            {
                h = DateTime.Now.Hour;
                m = DateTime.Now.Minute;
                s = DateTime.Now.Second;
                System.Console.WriteLine("{0}:{1}:{2}", h, m, s);
                Thread.Sleep(1000);
                Invalidate();
            }
        }
        private Thread TickThread;
        public Zegar()
        {
            InitializeComponent();
            TickThread = new Thread(new ThreadStart(Ticking));
            TickThread.Start();
        }

        protected override void OnPaint(PaintEventArgs pe)
        {
            base.OnPaint(pe);
            int tickLength = (Width < Height ? Width : Height) / 2;
            Pen redPen = new System.Drawing.Pen(System.Drawing.Color.Red);
            Pen hPen = new Pen(Brushes.DeepSkyBlue);
            hPen.Color = h_color;
            hPen.Width = 8.0F;
            Pen mPen = new System.Drawing.Pen(m_color);
            mPen.Width = 4.0F;
            Pen sPen = new System.Drawing.Pen(s_color);
            sPen.Width = 2.0F;
            h = h % 12;
            double angle = (h+m/60.0) / 12.0 * 360 - 90;
            angle = angle / 180 *Math.PI;
            int x = Width / 2 + (int)(tickLength * Math.Cos(angle));
            int y = Height / 2 + (int)(tickLength * Math.Sin(angle));
            pe.Graphics.DrawLine(hPen, new Point(Width/2, Height/2), new Point(x, y));

            angle = m / 60.0 * 360-90;
            angle = angle / 180 *  Math.PI;
            x = Width / 2+ (int)(tickLength * Math.Cos(angle));
            y = Height / 2 + (int)(tickLength * Math.Sin(angle));
            pe.Graphics.DrawLine(mPen, new Point(Width/2, Height/2), new Point(x, y));
            angle = s / 60.0 * 360 - 90;
            angle = angle / 180 * Math.PI;
            x = Width / 2+(int)(tickLength * Math.Cos(angle));
            y = Height / 2+(int)(tickLength * Math.Sin(angle));
            pe.Graphics.DrawLine(sPen, new Point(Width/2, Height/2), new Point(x, y));
        }
    }
}
```

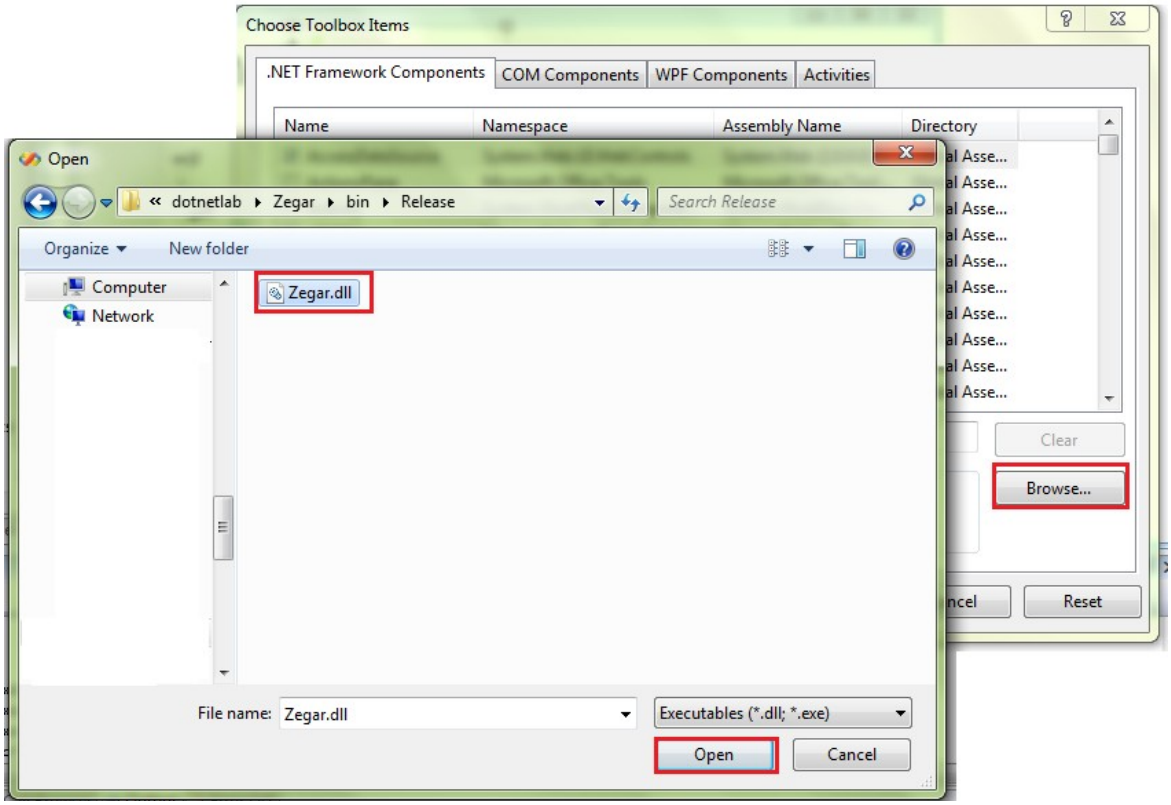
7. W **Zegar.Designer.cs,** metodę **Dispose(bool)** należy uzupełnić o zatrzymanie działania wątku**:**

```
protected override void Dispose(bool disposing)
 {
     if (disposing) _run = false;

     if (disposing && (components != null))
     {
         components.Dispose();
     }
     base.Dispose(disposing);
 }
```

8. Należy zbudować kontrolkę przy pomocy polecenia **build,** a następnie dodać ją do **Toolbox**'a za pomocą komendy **Choose items...** . Za pomocą opcji **add Tab** można dodać nową zakładkę np. **Custom** w której gromadzone będą własne kontrolki.



9. W okienku dialogowym należy odnaleźć plik **dll** aby dodać go do paska narzędzi.



10. Utworzyć nowy projekty typu **Windows Form Application** i dodać do niego kontrolkę Zegara.

Zadania do samodzielnego wykonania:

1. Wykonać kontrolkę pełniącą funkcję pola rysunkowego.
2. Wykonać kontrolki pełniące funkcję prostego komunikatora (kontrolka serwera i klienta).
3. Wykonać kontrolkę pełniącą funkcję wyświetlania obrazu z kamery USB (WebCam).
4. Wykonać kontrolkę pełniącą funkcję odtwarzacza multimedialnego (odtwarzane formaty: mp3, avi)