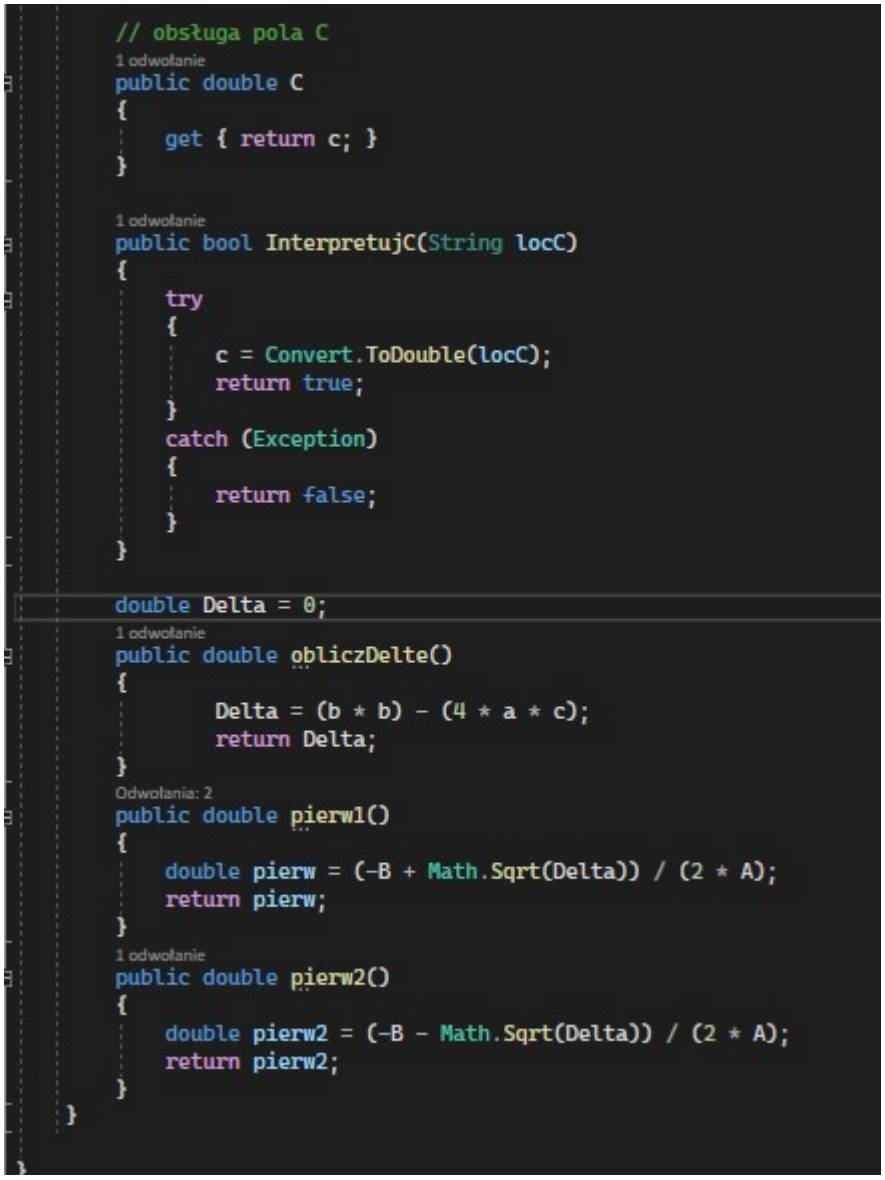
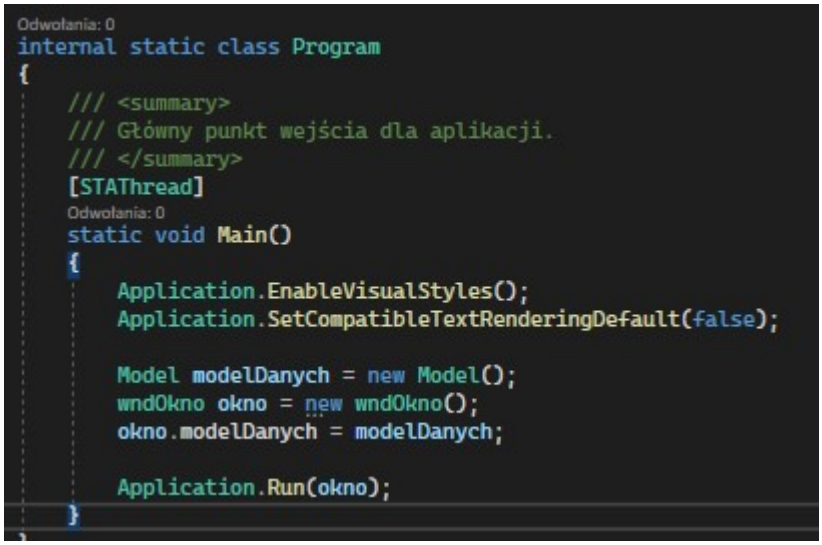
```
1 odwołanie
private void btCalculate_Click(object sender, EventArgs e)
{
    bool blad = false;
    // sprawdzanie czy zamiast liczby jest inny znak i przypisanie wartości
    if (!modelDanych.InterpretujA(textBoxA.Text))
    {
        MessageBox.Show("Niewłaściwa wartość współczynnika A");
        blad = true;

    }
    if (!modelDanych.InterpretujB(textBoxB.Text))
    {
        MessageBox.Show("Niewłaściwa wartość współczynnika B");
        blad = true;

    }
    if (!modelDanych.InterpretujC(textBoxC.Text))
    {
        MessageBox.Show("Niewłaściwa wartość współczynnika C");
        blad = true;

    }
    if (blad == true){
        return;
    }

    if (modelDanych.A == 0)
    {
        textBoxResult.Text = "A równe 0, równianie liniowe" + "\r\nPierwiastek : " + -modelDanych.C / modelDanych.B;
    } else
    {
        double Delta = modelDanych.obliczDelte();

        if (Delta < 0)
        {
            textBoxResult.Text = "Delta ujemna, brak wyników";
        }
        else if (Delta == 0)
        {
            double pierw = modelDanych.pierw1();//(-b + Math.Sqrt(Delta)) / (2 * A);
            textBoxResult.Text = "Pierwiastek 1: " + pierw.ToString();

        }
        else
        {
            double pierw1 = modelDanych.pierw1();//(-B + Math.Sqrt(Delta)) / (2 * A);
            double pierw2 = modelDanych.pierw2();//(-B - Math.Sqrt(Delta)) / (2 * A);
            textBoxResult.Text = "Pierwiastek 1: " + pierw1.ToString() + "\r\nPierwiastek 2: " + pierw2.ToString();

        }
```



```
        // obsługa pola C
        1 odwołanie
        public double C
        {
            get { return c; }
        }

        1 odwołanie
        public bool InterpretujC(String locC)
        {
            try
            {
                c = Convert.ToDouble(locC);
                return true;
            }
            catch (Exception)
            {
                return false;
            }
        }

        double Delta = 0;
        1 odwołanie
        public double obliczDelte()
        {
                Delta = (b * b) - (4 * a * c);
                return Delta;
        }
        Odwołania: 2
        public double pierw1()
        {
            double pierw = (-B + Math.Sqrt(Delta)) / (2 * A);
            return pierw;
        }
        1 odwołanie
        public double pierw2()
        {
            double pierw2 = (-B - Math.Sqrt(Delta)) / (2 * A);
            return pierw2;
        }
    }

}
```

Użyłem utworzonej klasy model w której interpretowałem i obliczałem wszystkie wejścia i wyjścia, do liczenia delty wykorzystałem kod z poprzedniego lab. Wykorzystałem funkcję "MessageBox" aby wyświetlać powiadomienia, a funkcję "*.Text" Aby zmienić tekst w polu tekstowym w trakcie działania programu. Wykorzytałem try/catch aby szukać błędów podczas interpretacji zmiennych.



Model modelDanych | tworzymy nowy obiekt modelu "modelDanych"

wndOkno okno | tworzy nowe okno programu

Application.Run(okno) | uruchamia aplikacje w oknie utworzonym przez wndOkno