```csharp
private void ButCeiling_Click(object sender, EventArgs e)
{
  double x, y, z,t;
  x = double.Parse(TxtX.Text);
  y = double.Parse(TxtY.Text);
  z = Math.Ceiling(x);
  t = Math.Ceiling(y);
  MessageBox.Show("Ceiling (x)=" + z.ToString() +
          "\nCeiling (y)=" + t.ToString());
}

private void ButFloor_Click(object sender, EventArgs e)
{
  double x, y, z, t;
  x = double.Parse(TxtX.Text);
  y = double.Parse(TxtY.Text);
  z = Math.Floor(x);
  t = Math.Floor(y);
  MessageBox.Show("Floor (x)=" + z.ToString() +
          "\nFloor (y)=" + t.ToString());
}
```

```csharp
private void ButMax_Click(object sender, EventArgs e)
{
  double x, y, z;
  x = double.Parse(TxtX.Text);
  y = double.Parse(TxtY.Text);
  z = Math.Max(x,y);
  MessageBox.Show("Max(x,y)=" + z.ToString());
}

private void ButExp_Click(object sender, EventArgs e)
{
  double x, y, z;
  x = double.Parse(TxtX.Text);
  y = double.Parse(TxtY.Text);
  z = Math.Exp(x);
  MessageBox.Show("Exp(x)=" + z.ToString());
}
```

3.2 Static Methods

- There are several motivations for modularizing a program with methods.

    1- Make the program development more manageable

    2- Software reusability—using existing methods (and classes) as building blocks to create new programs. With proper method naming and definition, we can create programs from standardized methods, rather than building customized code

- Defining Methods

    Methods must to be defined inside the braces of a class definition. The format of a method definition is:

    public static return-type Method's name (parameter-list)
    {
        Method's body
    }