* **Access modifiers -** 
  + Public – can be called by methods outside of that class
  + Private – can only be used by this class
* **Return data type** – void, can’t return data
* **Method name** – TurnLightOn
* **Parentheses** – (), methods always use

private void TurnLightOn()

{

lightOnPictureBox.Visible = true;

lightOffPictureBox.Visible = false;

lightStateLabel.Text = “ON”;

}

* **Paskel case** – every word starts with capital letter, all methods use paskel case
* **Best practices using methods**:

1. Method should perform one task
2. Use void if no data must be returned
3. Return only one thing if data must be returned
4. Declare/define the method in the appropriate class. What object does it provide the functionality for?

* **Argument** – passing what’s in the parentheses after the method, “Hello”
* **Class** - MessageBox
* **Method** - Show
  + MessageBox.Show(“Hello”) – Hello, actual parameter, value being passed
  + “Public void Show(string s1)” – string s1, formal parameter
  + When using “pass by value” actual parameter must have value
* **Parameter** – receiving whatever is being passed to it, string card
  + private void ShowCard(string card)
* **Void** **–** no data being returned
* Microsoft website dedicated to everything in the .NET Framework (Google, “.NET Framework” and functionality of method)
* Parse method to convert from string to int
  + S1 = “97408”;
  + int zip = int.Parse(s1);
  + Will return 97408
* Call the method sum and pass it x and y
  + int a = 15;
  + int x, y;
  + int z;
  + X =5;
  + Y = 20;
  + z = a \* sum(x,y);
* If “void” can just call method – sum(x,y); or TurnLightOn(); If something else like “int” has to be a part of an expression z + a \* sum(x,y);
* outputListBox.Items.Add(sum(x,y).ToString());
* **Reference parameter** – special type of parameter that does not receive a copy of the argument’s value. Can change value of the argument the variable passed to it.
  + Private void SetToZero(ref int number)
  + {
  + number = 0;
  + }
  + Above, reinitialized variable
* **Output parameter** – like a reference parameter but an argument does not have to be a value before it is passed into an output parameter and a method that has an output parameter must assign a value to the output parameter before it finishes executing.
  + Private void SetToZero (out int number)
  + {
  + number = 0;
  + }
* **Debugger –**

Pass by value – makes copy of argument into parameter, have to give variable value

Pass by reference – passing memory location, have to give variable value, does not make copy, avoid using this

Method2(ref x, ref y);

Private double Method2(double c, double d)

{

double temp;

blaa blaa blaa;

C=26;

return temp;

}

**Setter** – method that sets values of variables

**Output parameter** – method3(out x, out y); has to change value of x and y therefore don’t have to have values before method is called, works like ref parameter

**Returning Booleans** – private bool

if (methodName(argument list)) or if (methodName(x, y) == true)

{

}

else

{

}

* If methodName is true, else false