**Variable Scope**

The **scope** of a variable determines it's level of accessibility. A variable's scope is normally limited to the **block** in which it is defined. A block is defined by the starting and ending braces {......}. Entire forms are defined within something called a class and these classes have their own start and end braces. Therefore any variables defined inside these braces are accessible anywhere else within the same braces. The following code illustrates the basic boilerplate code created by Visual Studio for a form:

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

}

}

Take a close look at the braces. There are three blocks created by three sets of start and end braces.

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

A

{

C

B

InitializeComponent();

}

}

}

A variable made in block A can be accessed in blocks B and C. However a variable made in block B (usually referred to as **form/class level** or **instance** variable)cannot be accessed in block A but can be accessed in block C(variables made in blocks found in a form are called local variables). Finally a variable made in block C can only be accessed in block C. Technically variables are never made in the **namespace** block (block A).

What is wrong in the following code:

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

int x;

public Form1()

{

int x;

x=10;

InitializeComponent();

}

}

}

The variable x is declared twice, once as an instance and once as a local. When it says x=10 the program doesn’t know which x youre talking about.

You cannot have variables whose scope overlap each other. When the code **x=10;** is executed the program cannot determine which x you are referring to. As a rule, never create a form class/form level variable that has the same name as a local variable. If you do you can specify an instance or form level variable by using the **this** keyword. The following is an example:

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

int x;

public Form1()

{

int x;

this.x=10;

InitializeComponent();

}

}

}

Look at the next piece of code and determine the problem:

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;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

int x;

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

x=10;

}

}

}

The problem here is that you have created a local variable **x** in one block and are attempting to access it in another block. This is not allowed.

The simple rule to follow is to always make local variables unless a variable needs to be accessible in ore than one place.

**Questions:**

1.   In the following program skeleton, fill in the blank line where it would be correct to have the statement: target = 25

In other words, click on each button where the variable target is in scope.

Top of Form

public partial class Form1 : Form

{

int target;

void cMethod()

{

\_\_\_ target = 25 \_\_\_\_\_\_

. . . .

}

void aMethod()

{

target = 25

. . . .

}

void bMethod()

{

\_\_\_\_\_\_\_\_\_

target = 25. . .

}

}

public partial class Form2 : Form

{

int sum;

void aMethod() ()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

. . . .

}

void anotherMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

. . . .

}

void someMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

. . . .

}

}

public partial class Form3 : Form

{

void aMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

2. In the following program skeleton fill in the blank line where it would be OK to have the statement target = 25

Top of Form

public partial class Form1 : Form

{

int target;

void cMethod() ()

{

\_\_\_ target = 25 \_\_\_\_\_\_\_\_\_\_\_\_\_;

. . . .

}

void aMethod()

{

target = 25

. . . .

}

void bMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

. . . . target = 25 }

. . . .

}

public partial class Form2 : Form

{

int sum;

void aMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

. . . .

}

void someMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

. . . .

}

}

public partial class Form3 : Form

{

void aMethod() {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

3. Fill in the line where it would be OK to have the statement Messagebox.show( sum + ””);

Top of Form

public partial class Form1 : Form

{

int sum;

void aMethod()

{

\_\_\_\_\_\_ Messagebox.show( sum + ””)\_\_\_\_\_\_\_\_

. . . .

}

void bMethod()

{

\_\_\_\_\_\_\_ Messagebox.show( sum + ””)\_\_\_\_\_\_\_

. . . .

}

}

public partial class Form2 : Form

{

void aMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

4. 4. Fill in the Fissss Fill in the line where it would be OK to have the statement value = 5;

Top of Form

public partial class Form1 : Form

{

int sum;

void aMethod()

{

int value;

\_\_\_\_\_\_ value = 5\_\_\_\_\_\_\_\_

. . . .

}

void bMethod()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

. . . .

}

void cMethod( )

{

. . . .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int value;

. . . .

}

void dMethod( )

{

double value;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

value = 5

. . . .

}

}

5. 5. Decide if each statements sets the **instance variable** sum or the **local variable** sum.

Top of Form

public partial class Form1 : Form

{

int sum; // the instance variable

void aMethod()

{

sum = 2 ; // \_\_\_\_\_instance\_\_\_\_\_\_\_\_\_

. . . .

}

void bMethod()

{

int sum; // \_\_\_\_\_\_local\_\_\_\_\_\_\_\_

sum = 3 ; // \_\_\_\_\_\_\_\_\_\_local\_\_\_\_

. . . .

}

void cMethod( )

{

. . . .

sum = 23 ; // \_\_\_\_\_\_instance\_\_\_\_\_\_\_\_

int sum;// \_\_\_\_\_\_\_\_local\_\_\_\_\_\_

. . . .

}

}

6. 6. In the following program skeleton, decide if each statements sets the **instance variable** sum or the **local variable** sum.

Top of Form

public partial class Form1 : Form

{

int sum; // \_\_\_\_\_instance\_\_\_\_\_\_\_\_\_

void aMethod()

{

int sum; // \_\_\_\_\_local\_\_\_\_\_\_\_\_\_

this.sum = 5; // \_\_\_\_\_\_\_\_instance\_\_\_\_\_\_

. . . .

}

void bMethod()

{

int sum; // \_\_\_\_\_\_local\_\_\_\_\_\_\_\_

sum = 6 ; // \_\_\_\_\_\_\_\_\_\_local\_\_\_\_

. . . .

}

}

Bottom of Form

Bottom of Form

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Bottom of Form