អារម្ភកថា

សព្វថ្ងៃនេះ ការប្រើប្រាស់កុំព្យូទ័រ គឺជាមធ្យោបាយយ៉ាងសំខាន់មួយ ក្នុងការដោះស្រាយបញ្ហា និងជួយសម្រួលដល់ការងារ ប្រចាំថ្ងៃរបស់មនុស្សដែលរស់នៅលើពិភពលោក។ ភាគច្រើនការងារ ស្ទើរតែទាំងអស់របស់ពួកគេគឺពឹងផ្អែកទៅលើការ ប្រើប្រាស់កុំព្យូទ័រទាំងស្រុង ដែលជាហេតុទាមទារឲអ្នកប្រើប្រាស់ត្រូវមានសមត្ថភាព និងចំនេះដឹងគ្រប់គ្រាន់ក្នុងការ ប្រើប្រាស់ទៅលើវា។

ដើម្បីចូលរួមលើកស្ទួយវិស័យព័ត៌មានវិទ្យា នៅក្នុងព្រះរាជាណាចក្រកម្ពុជា មជ្ឈមណ្ឌលព័ត៌មានវិទ្យា អិនធ័រ ត្រូវបានបង្កើតឡើងផងដែរ ក្នុងគោលបំនង ដើម្បីជួយបណ្តុះបណ្តាលចំនេះដឹងផ្នែកព័ត៌មានវិទ្យាដល់ សិស្ស និស្សិត និងអ្នកសិក្សា ឲមានចំនេះដឹងពិតប្រាកដក្នុងផ្នែកនេះ ជាពិសេសមានសមត្ថភាពច្បាស់លាស់ក្នុងការប្រើប្រាស់កុំព្យូទ័រ។ ដូចនេះសូម សិស្ស និស្សិត និងអ្នកសិក្សាទាំងអស់ ខិតខំប្រឹងប្រែង សិក្សា ស្រាវជ្រាវ និងចាប់យកនូវចំនេះដឹង ដែលមជ្ឈមណ្ឌល បានបណ្តុះបណ្តាលជូន ដើម្បីឲពេលវេលាដែលអ្នកបានចំនាយ ក្លាយទៅជាប្រយោជន៍មួយយ៉ាងធំធេងសំរាប់ជីវិតរបស់អ្នកនាពេលអនាគត។

យើងខ្ញុំដែលជាអ្នករៀបរៀងនៃ មជ្ឈមណ្ឌលព័ត៌មានវិទ្យា អិនធ័រ សូមអរគុណចំពោះការគាំទ្ររបស់ សិស្ស និស្សិត និងអ្នកសិក្សាទាំងអស់ និងសូមអភ័យ ទោសរាល់កំហុសឆ្គងដែលកើតមានក្នុងករណីណាមួយ ហើយយើងខ្ញុំ នឹងខិតខំ រិះរកនូវអ្វីដែលថ្មីក្នុងផ្នែកព័ត៌មានវិទ្យានេះ ដើម្បីផ្តល់ជូនដល់ អ្នកសិក្សាបន្ថែមទៀត សូមអរគុណ និងសូមជំរាបសួរ ។

ក្រុមអ្នករៀបរៀងនៃ
មជ្ឈមណ្ឌលព័ត៌មានវិទ្យា អិនធ័រ

ក្រុមអ្នករៀបរៀង

អ្នករៀបរៀង:

ហូ ម៉ូន្នី

អ្នករចនាគំរប:

ហូ ម៉ូន្នី

អ្នកនាយអត្ថបទ និងរៀបចំរូបភាព:

ហូ ម៉ូន្នី

អ្នកកែសំរួលអក្ខរាវរុទ្ធ:

ហូ ម៉ូន្នី
ចំន្ធារិទ្ធ
ចិន សុវណ្ណ
កែវ សោភ័ណ្ឌ

រៀបរៀងលើកទី 1:

អាសយដ្ឋាន:
ផ្ទះលេខ 179 ផ្លូវ 173 សង្កាត់ ទំនប់ទឹក ខណ្ឌចំការមន រាជធានីភ្នំពេញ
Tel: 010-012-016 603 314
Website: www.enteritc.com | facebook.com/enteritc

មាតិកា

មេរៀនទី 1: ការណែនាំពី

C# Language

1.<u>និយមន័យ</u>:

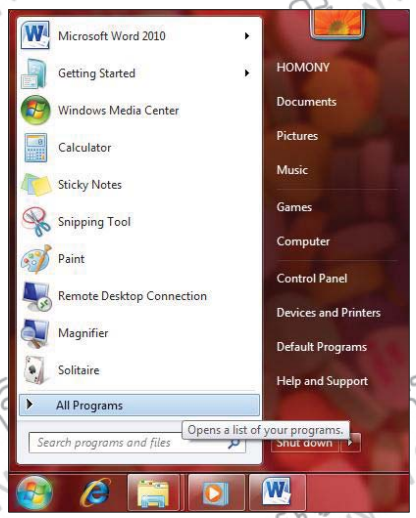
Microsoft Visual C# គឺជា component-oriented language មួយដែលមានសារៈសំខាន់បំផុតនៅក្នុង language ផ្សេងទៀតរបស់ក្រុមហ៊ុន Microsoft ។ C# ដើរតួយ៉ាងសំខាន់នៅក្នុង architecture នៃ Microsoft .NET Framework ហើយប្រសិនបើយើងមានចំនេះដឹងនៅក្នុងភាសា C, C++ ឬ Java រួចរាល់ហើយនោះ គឺយើងសិក្សា C# បានយ៉ាងងាយស្រួល។

2.<u>របៀបចាប់ផ្តើមដំណើការ Program ជាមួយ Visual Studio 2008 Environment</u>:

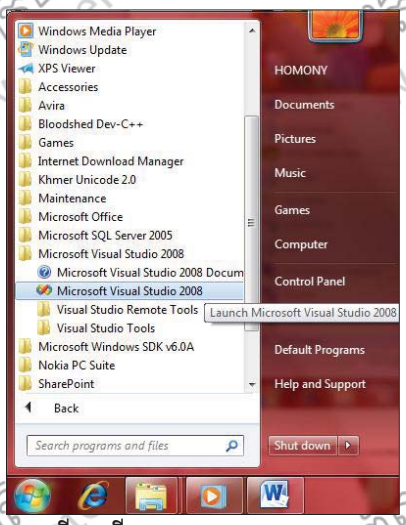
Visual Studio 2008 គឺជា Tool ឬ Environment មួយដែលមានសមត្ថភាពយ៉ាងពេញលេញក្នុងការបង្កើត Projects C# ដែលមានទំហំតូច ឬធំ។ Visual Studio 2008 អាចឲ្យយើងប្រើប្រាស់ C# ដើម្បីបង្កើតជា Console Application ឬ Graphical User Interface។ Console Application គឺជា Application ទាំងឡាយណាដែល run នៅក្នុង Command Prompt ចំនែក Graphical User Interface គឺ run ចេញជាទំរង់ Form សំរាប់ឲ users ងាយស្រួលក្នុងការចុចបញ្ជា ដើម្បីប្រើប្រាស់។

ដើម្បីបើកកម្មវិធីនោះសូមអនុវត្តតាមជំហានដូចខាងក្រោម:

1. ចុច Start Button >
2. All Program >

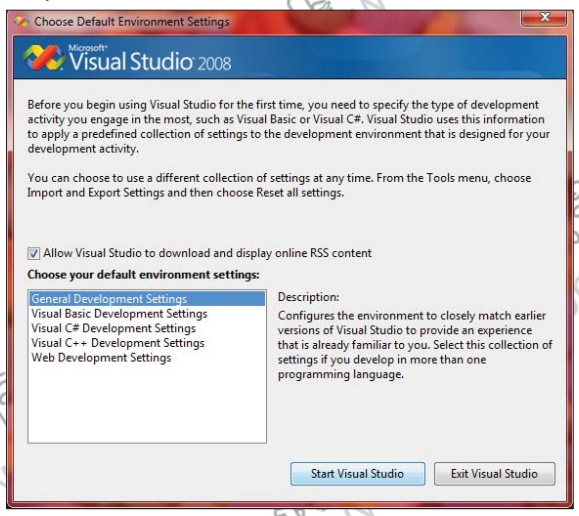


3. Microsoft Visual Studio 2008 >
4. Microsoft Visual Studio 2008 >



5. ជ្រើសរើសយក General Development Settings >

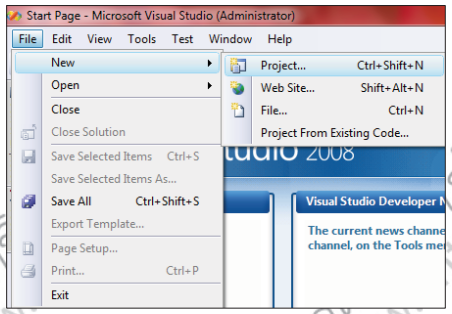
6. ចុច Start Visual Studio Button >



7. ចុច File Menu >

8. New >

9. Project >



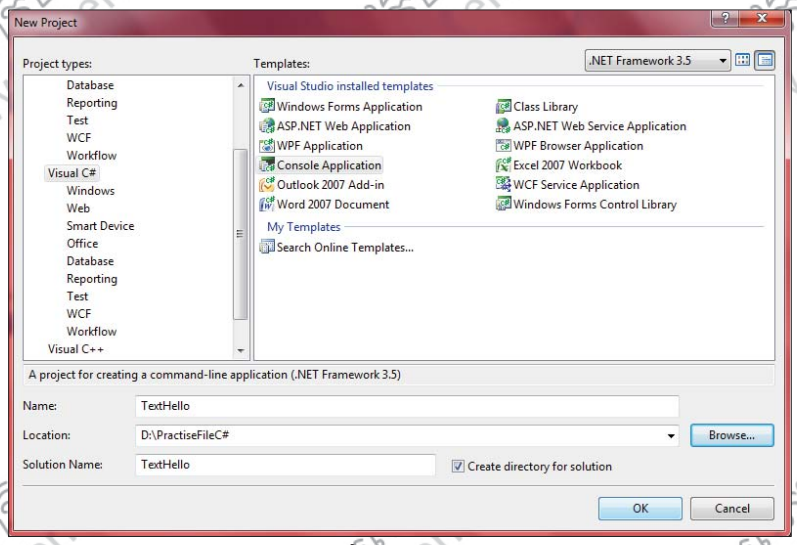
10. ជ្រើសរើសយក Visual C# >

11. ជ្រើសរើសយក Console Application >

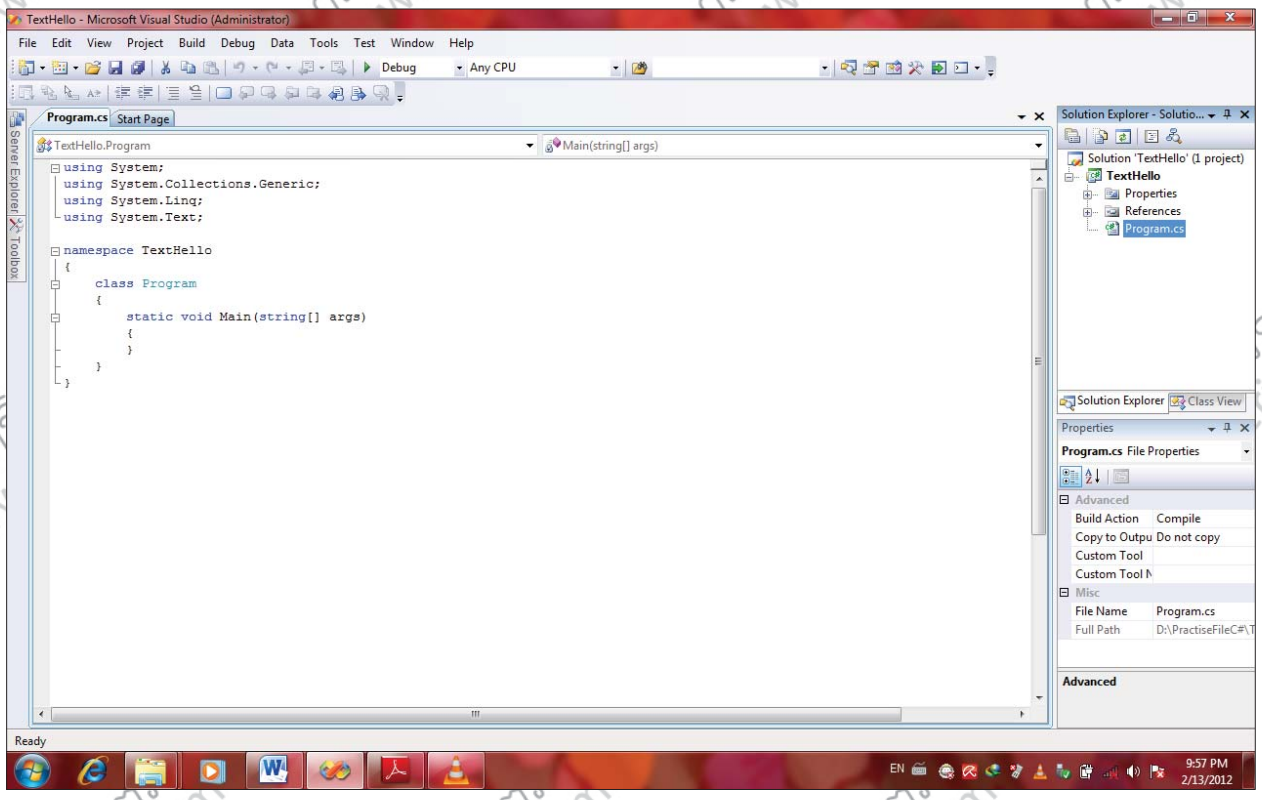
12. ក្នុងប្រអប់ Name សូមដាក់ឈ្មោះ (Ex: TextHello) >

13. ក្នុងប្រអប់ Location សូមជ្រើសរើសយកទីតាំងរក្សាទុក >

14. ចុច OK Button >



15. ខាងក្រោមនេះជាលទ្ធផលដែលទទួលបាន



មុននឹងធ្វើការសរសេរកូដ យើងត្រូវស្គាល់ពី Solution Explorer នៅក្នុង Visual Studio ជាមុនសិន ដែលក្នុងនោះមាន:

> Solution 'TextHello' :គឺជា top-level solution file ហើយវាមានតែមួយប៉ុណ្ណោះក្នុងមួយ Application ដែលឈ្មោះពិតរបស់វាគឺមានបន្ថែម *.sln នៅខាងក្រោយ (TextHello.sln) ។

> TextHello :គឺជា C# project file ដែលនៅក្នុង Solution Folder ឈ្មោះពិតរបស់វាគឺ TextHello.csproj។

> Properties : គឺជា Folder នៅក្នុង TextHello project ដែលនៅក្នុងវាមានដូចជា File មួយឈ្មោះ AssemblyInfo.cs (វាគឺជា File ពិសេសដែលអនុញ្ញាតិឲយើង add ឈ្មោះអ្នកបង្កើត (author), កាលបរិច្ឆេទនៃការ បង្កើត Program ។

> References : គឺជា Folder ដែលផ្ទុកនូវ references សំរាប់ compile Code ដែលយើងសរសេរទៅជា Assembly (ភាសាម៉ាស៊ីន)។

> Program.cs : គឺជា C# Source file ដែលតែងតែត្រូវបាន display នៅក្នុង Code and Text Editor window ។ វាជាកន្លែងដែលយើងត្រូវសរសេរកូដ ដើម្បីបង្កើត Console Application ហើយ

ក្នុងនោះ Visual Studio 2008 បានផ្តល់នូវ Code មួយចំនួនបន្ថែមដោយស្វ័យប្រវត្តិ ដើម្បី ជួយឲ Programmer ងាយស្រួលក្នុងការសរសេរកូដ។

3.ការចាប់ផ្តើមសរសេរកូដ:

ឧទាហរណ៍ខាងក្រោមនេះបង្ហាញពីការសរសេរកូដ ដោយបង្ហាញពី Welcome to Enter Center មកលើ Screen:

```
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to Enter Center!");
        }
    }
```

ការបកស្រាយកូដ:

```
class Program
    {

    }
```

គឺជា Class មួយឈ្មោះ Program ដែលវាស្ថិតនៅក្នុង Solution Explorer ឈ្មោះ Program.cs file

```
        static void Main(string[] args)
        {

        }
```

វាគឺជា Main Function ឬ Function មួយដែលសំខាន់ជាងគេនៅក្នុង Code (គ្រប់ Code ទាំងអស់ត្រូវសរសេរនៅក្នុង Main Function ជានិច្ច) ។ C# គឺជាភាសា Case-Sensitive មានន័យថា Main ខុសពី main និងខុសពី MAIN ។

```
        Console.WriteLine("Welcome to Enter Center!");
```

គឺសំរាប់ធ្វើការបង្ហាញពាក្យដែលនៅក្នុង Double Quote ("") មកលើ Screen ។ Ex: Welcome to Enter Center ចំពោះ Console គឺជា Class សំរាប់ឲយើងប្រើប្រាស់នូវ Standard Input Output មួយចំនួនដូចជា WriteLine សំរាប់បង្ហាញព័ត៌មានចេញមកលើ Screen ឬ ReadLine សំរាប់ទទួលយកទិន្នន័យពី Keyboard ។ Semicolon (;) ត្រូវបានប្រើប្រាស់នៅខាងចុងនៃ Statement ដើម្បីបញ្ចប់នូវ Statement នីមួយៗជានិច្ច។

4.ការប្រើប្រាស់ Comment:

Comment ត្រូវបានប្រើប្រាស់នៅក្នុង source code ដើម្បីសរសេរជា statement ខ្លីៗ ការសម្រាប់ធ្វើជាការសំគាល់ឬពាក្យពន្យល់ផ្សេងៗ ហើយវាមិនត្រូវបាន read ដោយ compilerនោះទេ។ ជាទូទៅ comment ត្រូវបានប្រើប្រាស់ដោយ programmerដើម្បីសរសេរពន្យល់ ឬបញ្ជាក់ពីថ្ងៃដែលចាប់ផ្តើមសរសេរ code ក្នុង source code ។ comment មានពីរប្រភេទដូចជា Line comment និង Block comment ។

Line comment ប្រើសំរាប់ដាក់ comment នៅក្នុង Source code ជាជួរមួយៗដោយប្រើប្រាស់សញ្ញា double slash (//)

Block comment ប្រើសំរាបដាក់ comment នៅក្នុង Source code ជាច្រើនជួរដោយប្រើ /* comment */ ។

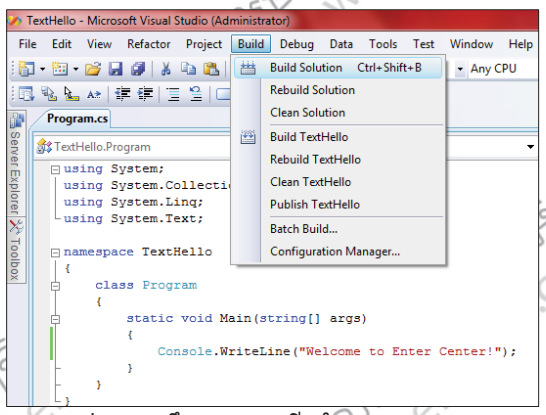
Ex:

```
/* Write on 14 Feb 2012
By Ho Mony */
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to Enter Center!");//print
        }
    }
```

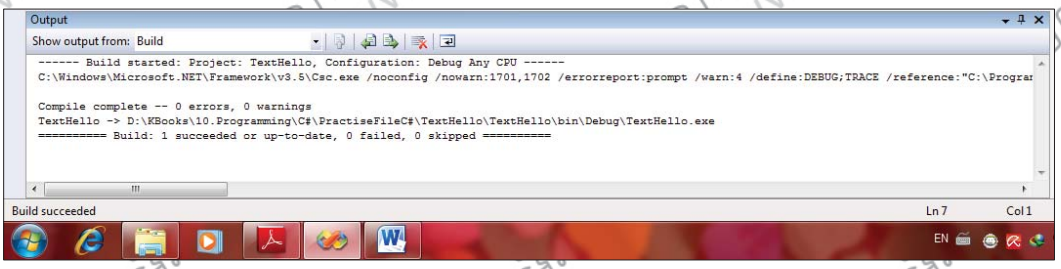
5. របៀប Build Console Application:

1. Build Menu >

2. Build Solution (Ctrl+Shift+B) >

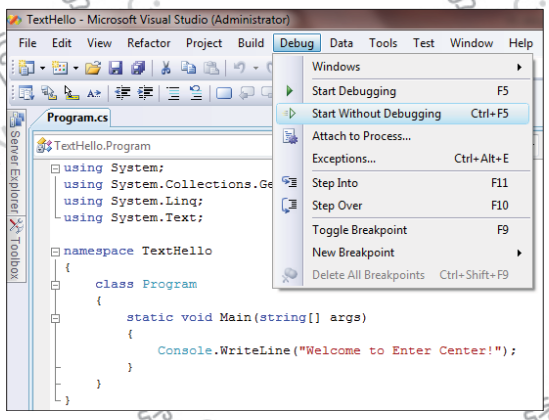


3. បន្ទាប់មកវានឹងបង្ហាញពីផ្ទាំង Output windows ដែលជាដំនើការ Compile ទៅលើ Code ដែលបានសរសេរ >

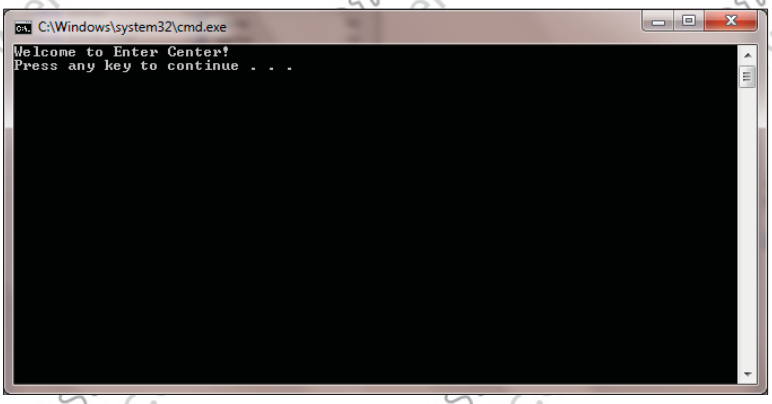


4. ចុច Debug Menu >

5. Start Without Debugging (Ctrl+F5) >



6. ខាងក្រោមនេះជាលទ្ធផលដែលទទួលបាន



6. <u>សិក្សាពី namespace និង Assembly:</u>

Using Statement គឺត្រូវបានប្រើប្រាស់នៅខាងមុខ namespace ដើម្បីនាំយក items (Method ឬ Properties) របស់ Class មកប្រើប្រាស់ដោយសេរីនៅក្នុង Source Code។

Ex:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

Class ដែលបានសរសេរគឺត្រូវបាន compiled ទៅជា Assemblies ដែលវាជា File មួយមាន extension *.dll ឬទៅជា *.exe file ។

7. <u>ការបង្កើត Graphical Application:</u>

ចំពោះការបង្កើត Graphical Application, Visual Studio 2008 បានផ្តល់នូវ Views ពីរសំរាប់ធ្វើប្រាស់ដូចជា:

> Design View : សំរាប់រៀបចំ Layout ឬទំរង់របស់ Form ដែលត្រូវបង្កើត ។

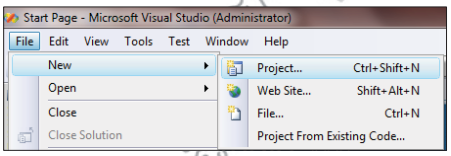
> Code View : សំរាប់កែប្រែ ឬសរសេរកូដបន្ថែមទៅឲ Application ។

ក្នុងនោះ Visual Studio 2008 បានផ្តល់នូវ templates ចំនួនពីរសំរាប់ build graphical application ។

> Windows Forms Application គឺជា technology ដំបូងគេរបស់ .NET Framework version 1.0

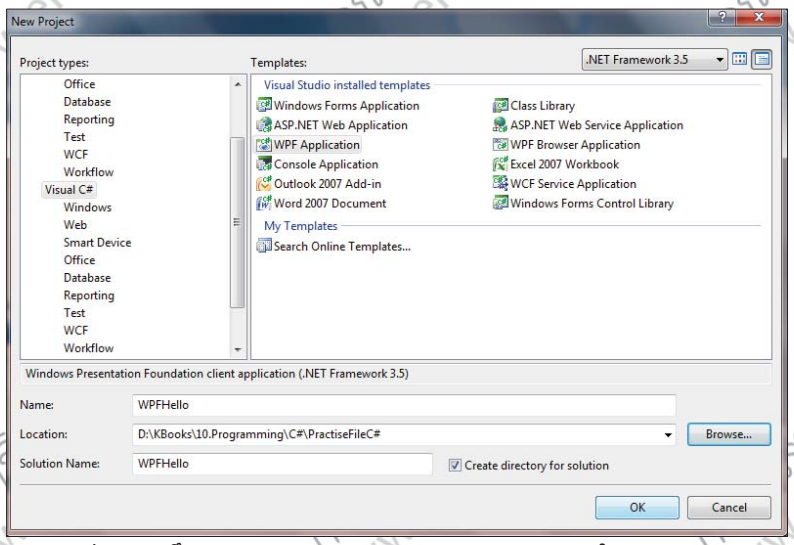
> Windows Presentation Foundation គឺជា enhanced technology ថ្មីដែលត្រូវបានបង្ហាញនៅក្នុង .NET Framework version 3.0 ដោយវាបានបន្ថែម features សំខាន់ៗមួយចំនួនទៀតលើស Windows Forms

1. ចុច File Menu >
2. New >
3. Project (Ctrl+Shift+N) >

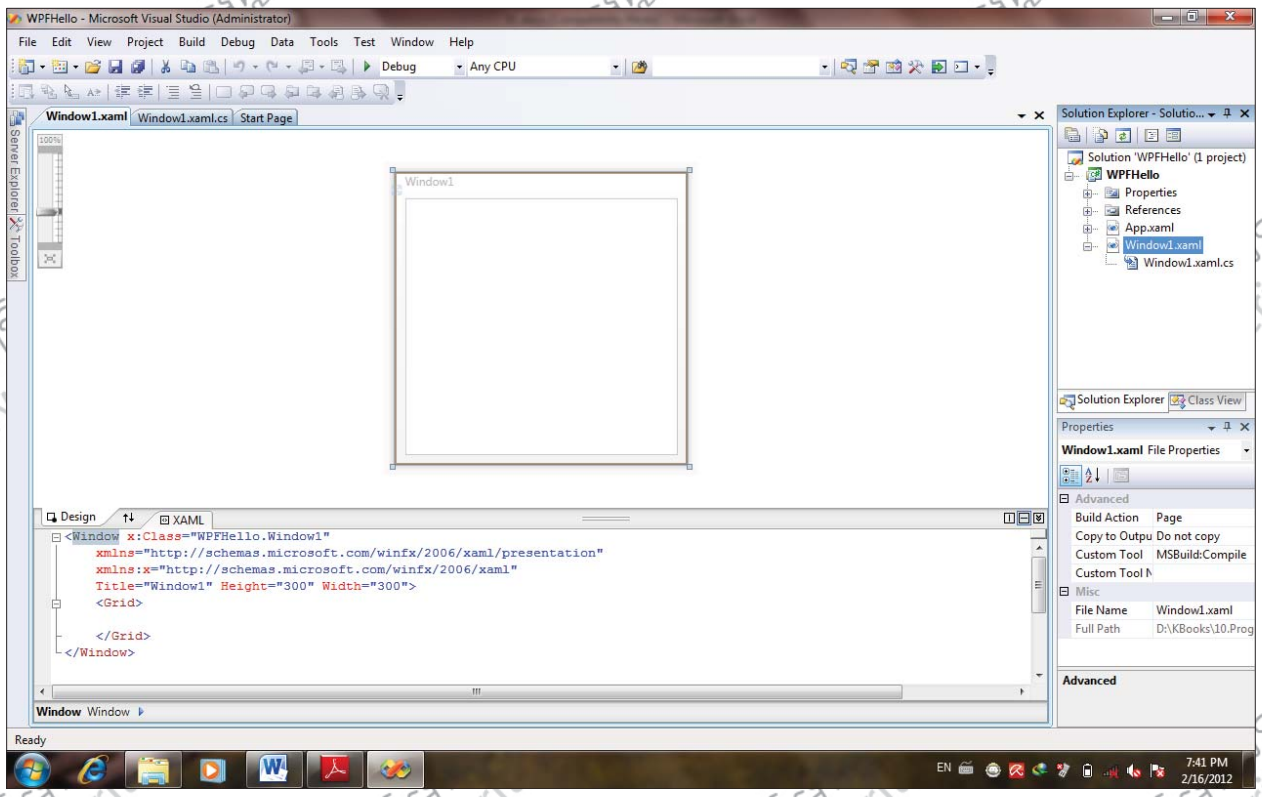


4. ត្រង់ Project types ចុចលើ visual C# >
5. ត្រង់ Templates ជ្រើសរើសយក WPF Application >
6. ក្នុងប្រអប់ Name ដាក់ឈ្មោះ WPFHello >
7. ក្នុងប្រអប់ Location សូមជ្រើសរើសទីតាំងរក្សាទុក >

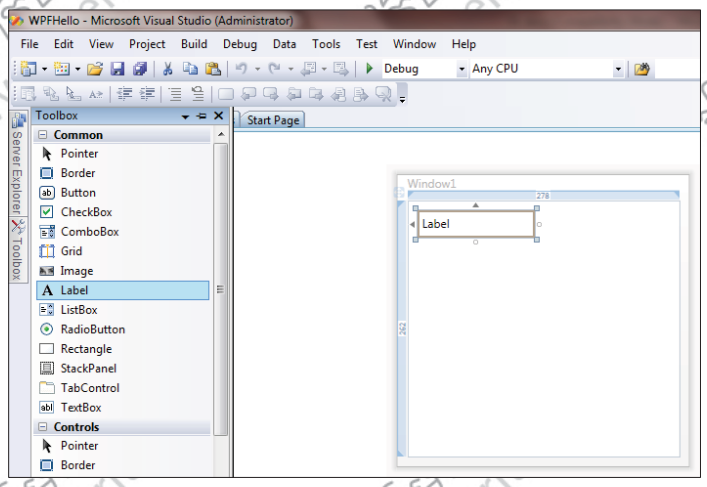
8. ចុច OK Button >



9. បន្ទាប់មកវានឹងបង្ហាញ Design View Window រួមទាំង XAML Windows (eXtensible Application Markup Languge) ដូចខាងក្រោម >

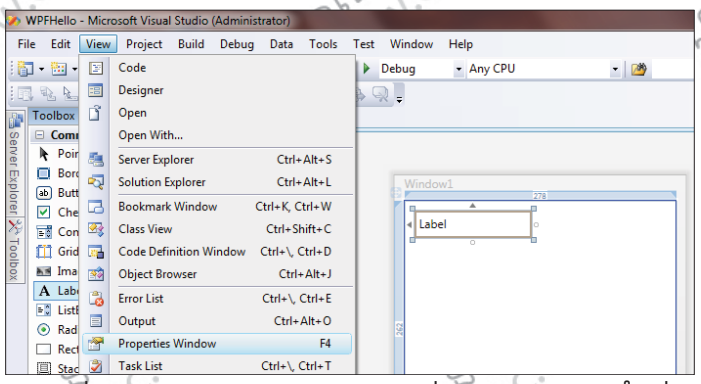


10. ក្នុង Toobox សូមចុច Double Click លើ Label ដើម្បីគូរវានៅក្នុង Window Form >

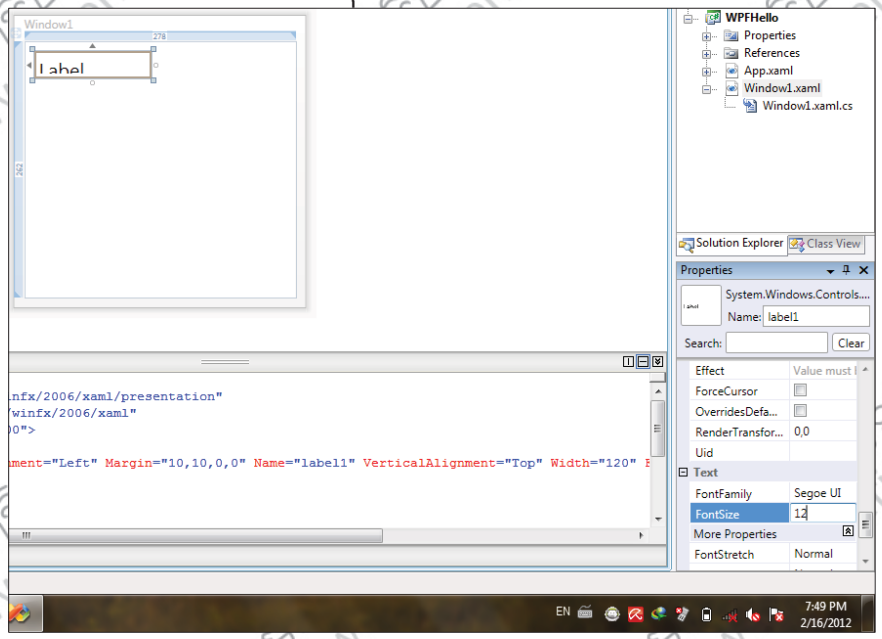


11. ចុច View Menu >

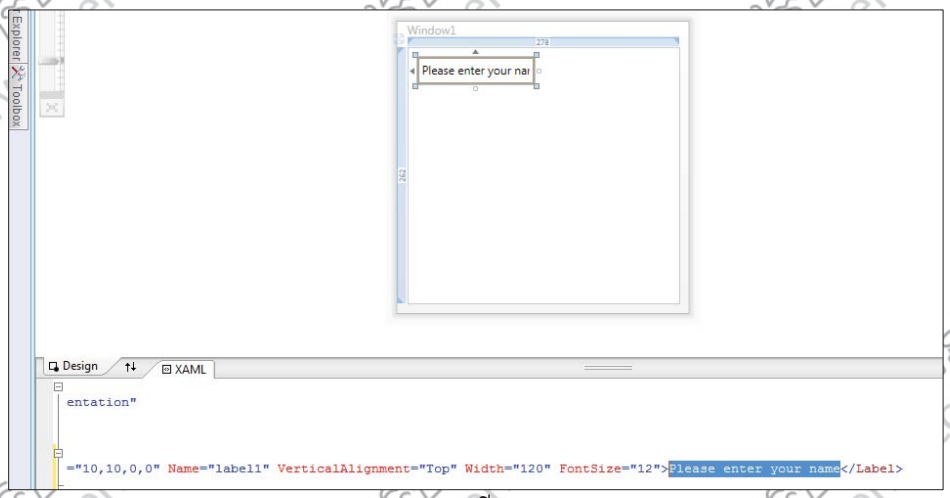
12. Properties >



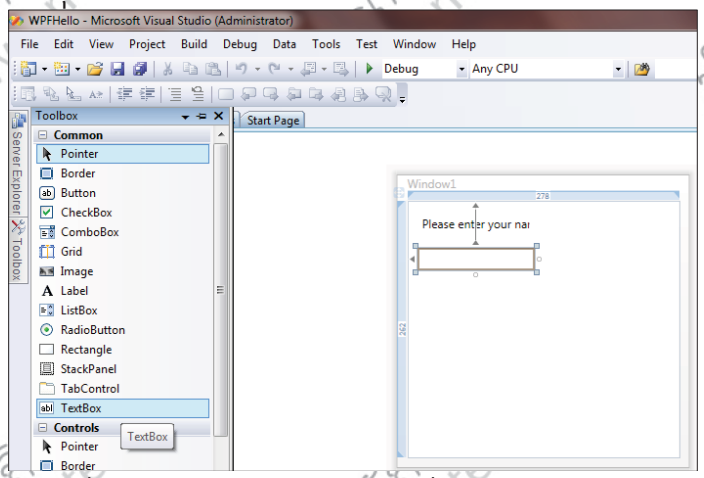
13. ត្រង់ Properties Window ក្នុងប្រអប់ FontSize សូមកំនត់លេខ 12 >



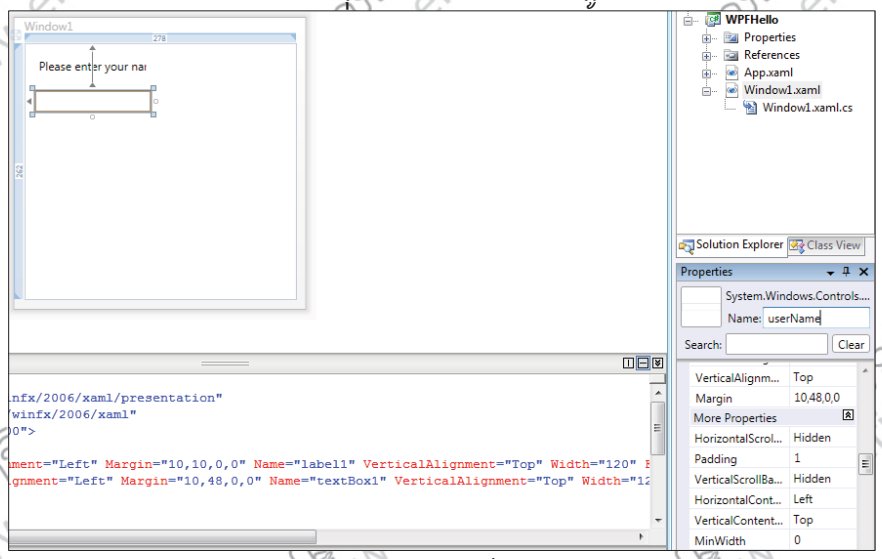
14. ក្នុង XAML window សូមសរសេរពាក្យ Please enter your name នៅត្រង់ចន្លោះ <Label> </Label>



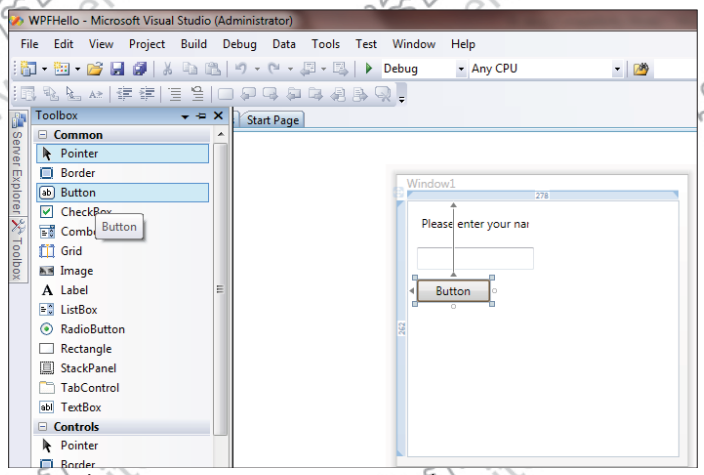
15. ក្នុង Toolbox សូមចុច Double Click លើ TextBox >



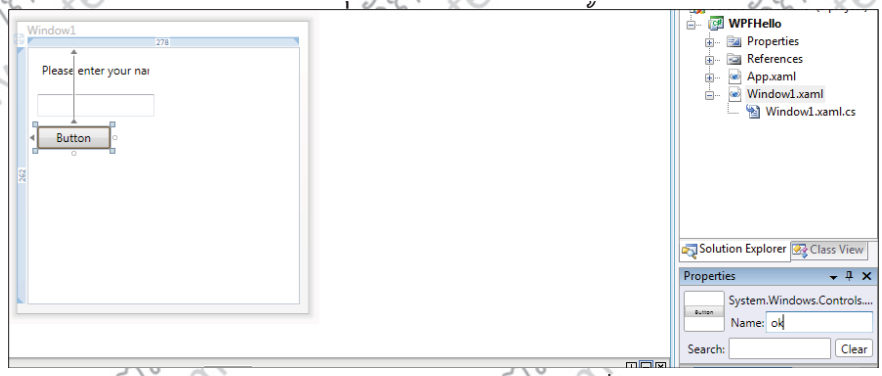
16. ត្រង់ Properties window ក្នុងប្រអប់ Name សូមប្តូរទៅជា userName >



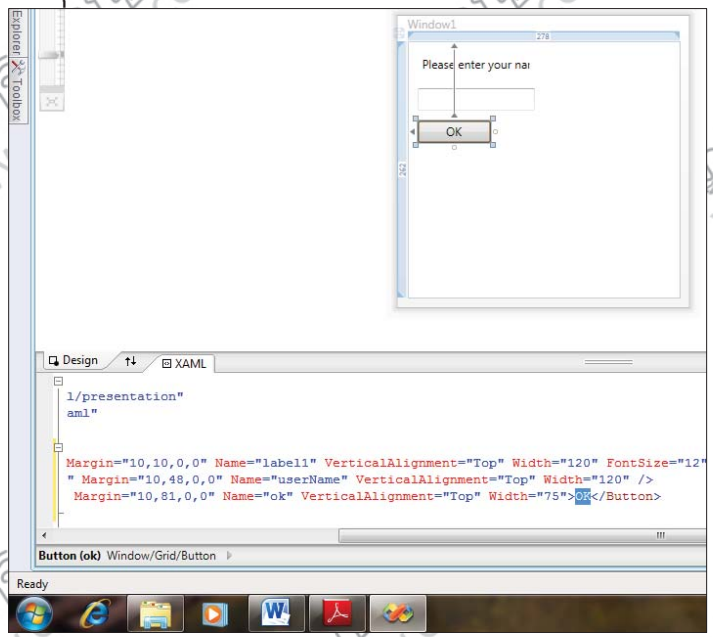
17. ក្នុង Toolbox សូមចុច Double Click ត្រង់ Button >



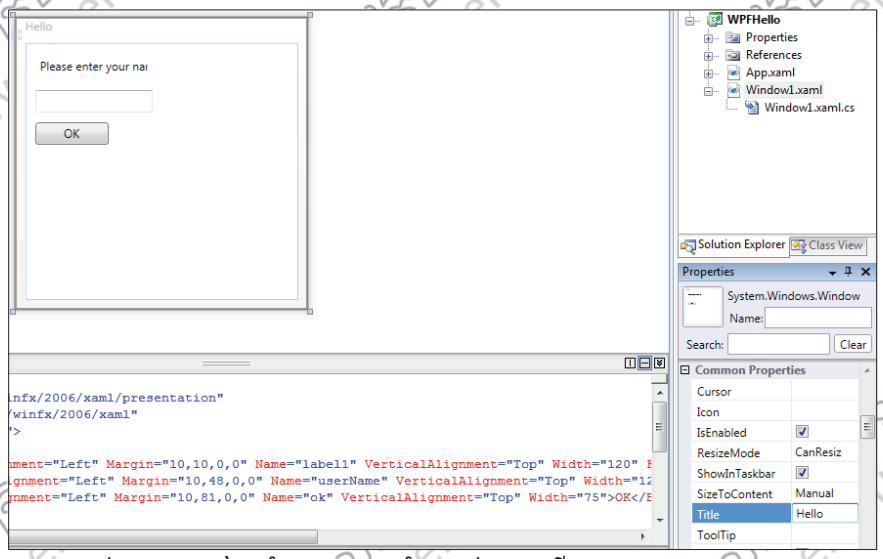
18. ត្រង់ Properties window ក្នុងប្រអប់ Name សូមប្តូរទៅជា ok >



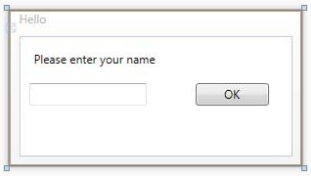
19. ក្នុង XAML window សូមសរសេរពាក្យ OK នៅត្រង់ចន្លោះ <Button> </Button>



20. Select លើ Form ហើយក្នុង Properties Window ត្រង់ប្រអប់ Title សូមប្តូរទៅជា Hello >

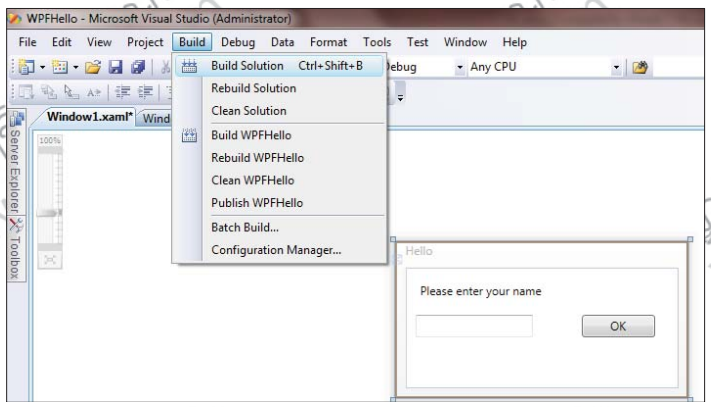


21. បន្ទាប់មកសូមរៀបចំ Objects ទាំងអស់នៅលើ Form ឲបានដូចរូបខាងក្រោម >



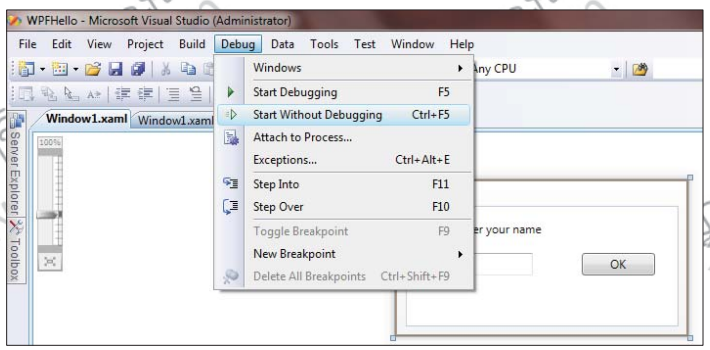
22. ចុច Build Menu >

23. Build Solution (Ctrl+Shift+B) >

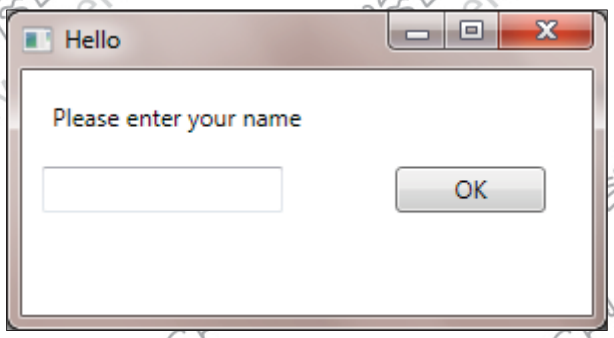


24. ចុច Debug Menu >

25. Start Without Debugging (Ctrl+F5) >

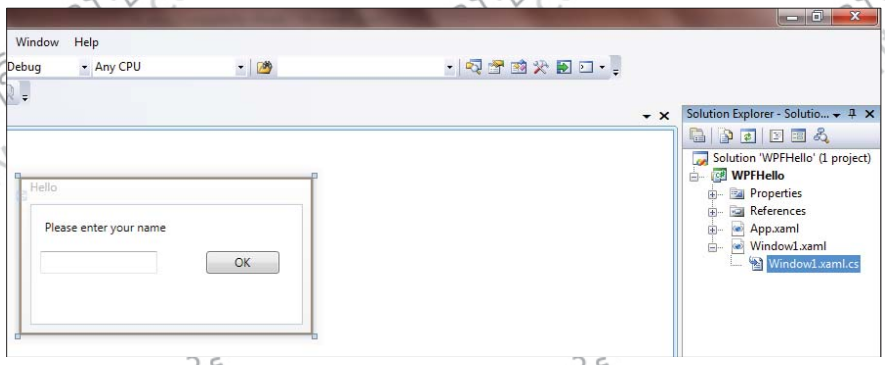


26. ខាងក្រោមនេះជាលទ្ធផលដែលទទួលបាន



8. <u>ការសរសេរ Code បន្ថែមនៅក្នុង Graphical Application</u>:

1. ចុច Double លើ Button OK >



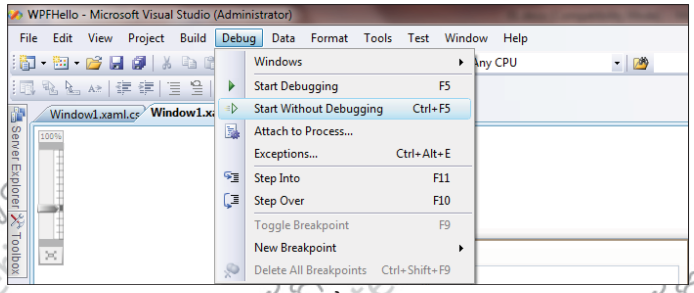
.

2. មន្តាម៉ែកសូមសរសេរកូដនៅក្នុងចន្លោះ { } របស់ private void ok_Click ដូចខាងក្រោម

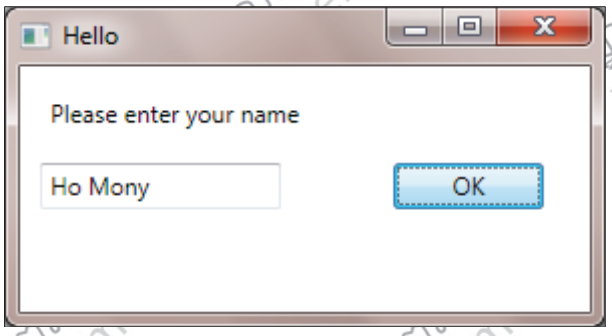
```
private void ok_Click(object sender, RoutedEventArgs e)
        {
            MessageBox.Show("Hello " + userName.Text);
        }
```

3. ចុច Debug Menu >

4. Start Without Debugging >



5. ខាងក្រោមនេះជាលទ្ធផលដែលទទួលបាន



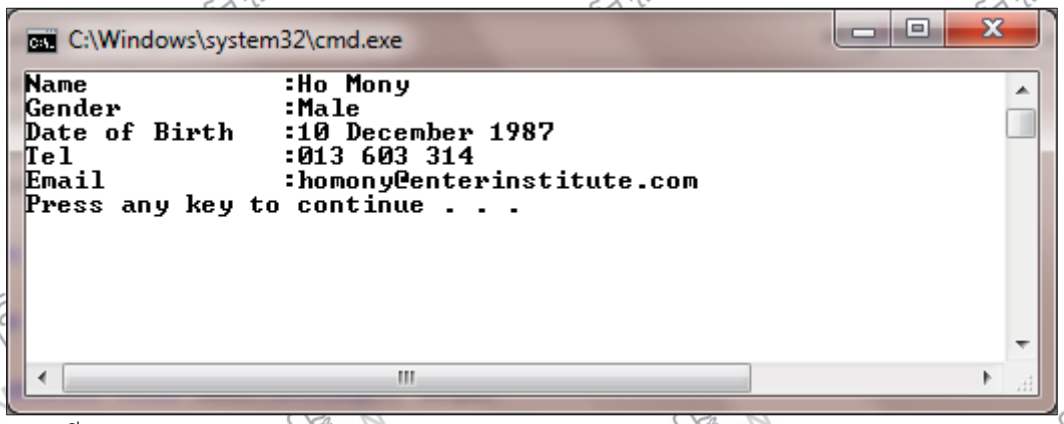
9. <u>លំហាត់</u>:

ចូរសរសេរ code មួយដើម្បី display ព័ត៌មានមួយចំនួនដូចខាងក្រោម:

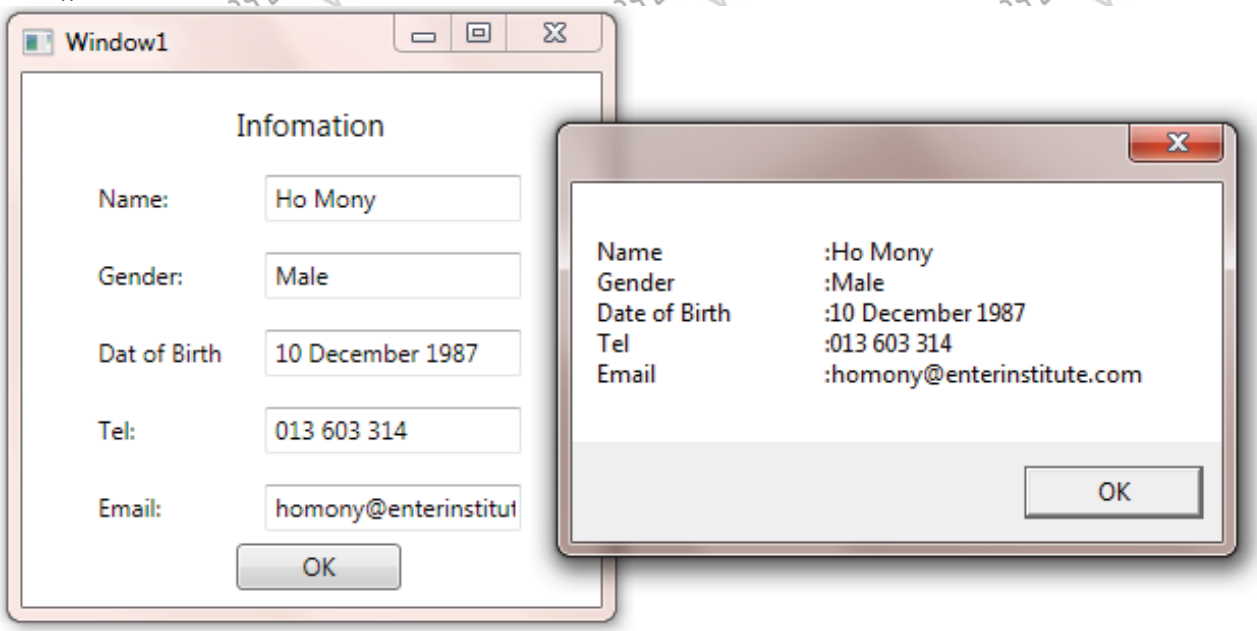
> ឈ្មោះរបស់អ្នក
> ភេទរបស់អ្នក
> ឆ្នាំកំនើត
> លេខទូរសព្ទ
> Email

```
Name              :Ho Mony
Sex               :Male
Date of Birth     :10 December 1987
Tel               : 013 603 314
Email       : homony@enterinstitute.com
```

1. បង្កើតតាម Console Application:



2. បង្កើតតាម WPF Application:



មេរៀនទី 2: សិក្សាពី Variables, Operators, និង Expressions

1.Statements:

Statement គឺជាការបញ្ជាដើម្បីដំនើការការងារណាមួយនៅក្នុង Source Code ហើយ Statement នីមួយៗ ត្រូវបញ្ចប់ដោយ Semicolon (;) ។

Ex:

```
Console.WriteLine("Welcome to Enter Center!");
```

ចំពោះ C# គឺជាប្រភេទ free format language ដែលមាននិយមថាក្នុងការបន្ថែម space ទទេ , Tab, ឬ Enter នៅ ក្នុង Source Code គឺមិនធ្វើឲមានភាព Error កើតឡើងនោះទេ។

2.ការប្រើប្រាស់ Variable:

Variables គឺជាកន្លែងរក្សាទុកទិន្នន័យក្នុង memory។ គ្រប់ Variables ទាំងអស់ត្រូវតែមានឈ្មោះនិងប្រភេទទិន្នន័យដែលវាត្រូវផ្ទុក ហើយក្នុងនោះត្រូវ declare (ប្រកាស) វាជាមុនទើបអាចប្រើប្រាស់បាននៅពេលក្រោយ។

3.សិក្សាពី Identifiers:

Identifiers គឺជាការដាក់ឈ្មោះឲខុសៗគ្នាទៅឲ elements នៅក្នុង programs ដែលមានដូចជា Variables, namespaces, classes, ឬ methods ដែលការដាក់ឈ្មោះគឺត្រូវបាន និងទៅតាមក្បួនខ្នាតត្រឹមត្រូវដែលបានទទួលស្គាល់ដោយ C# ។

ក្នុងការកំនត់ឈ្មោះ Identifiers ត្រូវកំនត់តាមលក្ខខណ្ឌដូចខាងក្រោម៖

1. តួអក្សរដំបូងចាប់ផ្តើមដោយ អក្សរ ឬ underscore ប៉ុន្តែមិនមែនជាលេខ

Ex:

Identifiers	លទ្ធផល	ហេតុផល
Enter	ត្រូវ	ចាប់ផ្តើមដោយអក្សរ
_score	ត្រូវ	ចាប់ផ្តើមដោយ underscore
3plan	ខុស	ចាប់ផ្តើមដោយលេខ
plan3	ត្រូវ	ចាប់ផ្តើមដោយអក្សរមុនលេខ

2. មិនអនុញ្ញាតិឲប្រើប្រាស់ Space ឬសញ្ញាពិសេស (#,$,*,+,….)

Ex:

Identifiers	លទ្ធផល	ហេតុផល
Enter Center	ខុស	មិនអាចប្រើ Space បានទេ
result%	ខុស	មាននិមិត្តសញ្ញា%
footballTeam$	ខុស	មាននិមិត្តសញ្ញា%

3. មិនអនុញ្ញាតិឲប្រើប្រាស់ជាមួយនឹង Reserved Identifiers ដែលមានចំនួន 77 identifiers (Keyword)

Ex:

C++ Keywords	C++ Keywords	C++ Keywords
abstract as	fixed	sealed
base	float	short
bool	for	sizeof
break	in	stackalloc
byte	int	static
case	interface	string
break	internal	struct
case	is	switch
catch	lock	this
char	long	throw
checked	namespace	true
class	new	try
const	null	typeof
continue	object	uint
decimal	operator	ulong
delegate	out	unchecked
do	override	unsafe
double	params	unshort
else	private	using
enum	protected	virtual
even	public	void
explicit	readonly	volatile
extern	ref	while
false	return	
finally	sbyte	

ចំពោះ keywords ដែលបានប្រើប្រាស់នៅក្នុង Code and Text Editor window គឺតែងតែបង្ហាញពណ៌ខៀវជានិច្ច។

4.<u>ការប្រកាស Variables (Variables Declaration)</u>:

ពេលដែលយើងប្រកាស Variable គឺយើងត្រូវធ្វើការកំនត់ពី data type (ប្រភេទទិន្នន័យ) ដែលវាត្រូវទទួលយកផងដែរ។ Data type មានដូចជា: ចំនួនគត់ (integers), លេខរៀស (floating-point numbers), ឬអក្សរ (string) ជាដើម។ ហើយការប្រកាស Variable គឺជាការប្រាប់ទៅដល់ Compiler ឲរៀបចំទីតាំង memory សម្រាប់ រក្សាទុកនូវទំហំ និងប្រភេទទិន្នន័ដែល Variable នោះទទួលយក។

Ex: ខាងក្រោមនេះគឺជាការប្រកាស Variable មួយឈ្មោះ age មានប្រភេទទិន្នន័យ (Data Type) ជាចំនួនគត់ integer

```
int age;
```

បន្ទាប់ពីយើងបានធ្វើការប្រកាស Variable រួចរាល់ហើយនោះ គឺយើងអាចធ្វើការ assign តំលៃទៅឲ Variable បានដែរ។ សញ្ញា(=) គឺជា assignment operator ដែលវាត្រូវបានប្រើប្រាស់ដើម្បី បោះតំលៃដែលនៅខាងស្តាំទៅឲ Variable ដែលនៅខាងឆ្វេង។

Ex: ខាងក្រោមនេះគឺជាការតំលៃ 42 ទៅឲ Variable មួយឈ្មោះ age:

```
int age;
age = 42;
```

បន្ទាប់មកពីបាន assign តំលៃទៅឲ Variable ហើយនោះគឺយើងអាចធ្វើការ display វាមកលើ Screen បាន ហើយក្នុងនោះ សូមចងចាំថានៅលើ Screen គឺបង្ហាញតំលៃរបស់ Variable មិនមែនបង្ហាញឈ្មោះ Variable នោះទេ។

Ex:

```
int age;
age = 42;
Console.WriteLine(age);//42
```

យើងមិនអាចធ្វើការ ប្រើប្រាស់ Variable ភ្លាមៗ ដោយមិនបាន assign តំលៃទៅឲវានោះទេ ព្រោះវានឹងមាន Error កើតឡើងនៅក្នុង Program។

Ex:

```
int age;
Console.WriteLine(age);//compile-time error
```

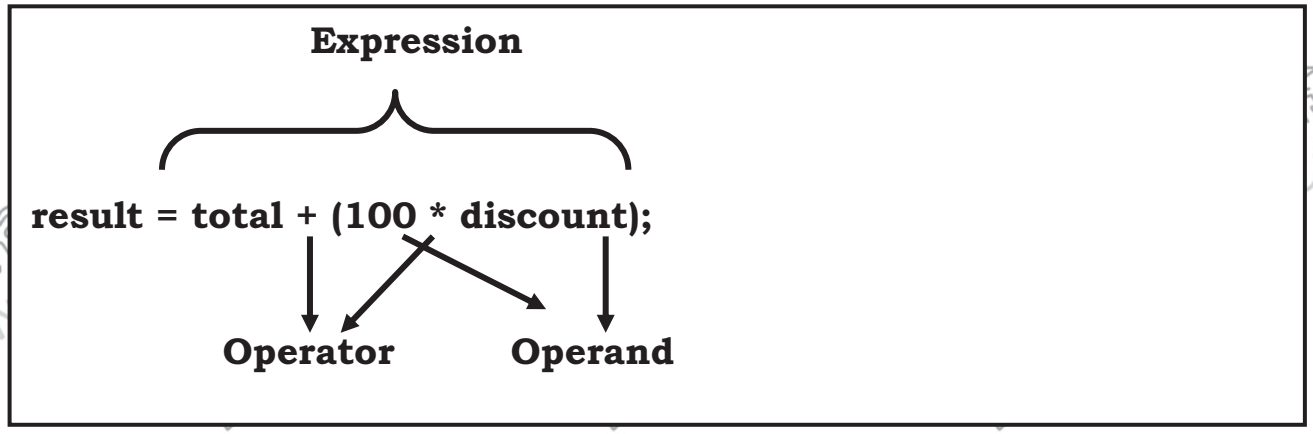
5.សិក្សាពី Primitive Data Type:

Data Type គឺត្រូវបានប្រើប្រាស់ជាមួយនឹង Variable ដើម្បីធ្វើការបញ្ជាក់ពីប្រភេទទិន្នន័យពិតប្រាកដដែល Variable ត្រូវទទួលយក។

Data Type	Description	Size (bits)	Range	Sample usage
int	លេខចំនួនគត់	32 bits = 4 bytes	- 2^{31} ដល់ 2^{31}- 1	int count; count = 42;
long	លេខចំនួនគត់	64 bits = 8 bytes	- 2^{63} ដល់ 2^{63}- 1	long wait; wait = 42L;
float	លេខមានក្បៀស	32 bits = 4 bytes	±1.5 x 10^{45} ដល់ ±3.4 x 10^{38}	float away; away = 0.42F;
double	លេខមានក្បៀស	64 bits = 8 bytes	±5.0 x 10^{-324} ដល់ ±1.7 x 10^{308}	Double trouble; trouble = 0.42;
decimal	លេខមានក្បៀស	128 bits = 16 bytes	28 significant figures	decimal coin; coin = 0.42M;
string	តួអក្សរច្រើនតួ	16 bits ក្នុង 1 តួ	មិនកំនត់	string vest; vest = "fortytwo;
char	តួអក្សរមួយតួ	16 bits = 2 bytes	0 ដល់ 2^{16}- 1	char grill; grill = 'x';
bool	Boolean	8 bits = 1 byte	True or False	bool teeth; teeth = false;

6.សិក្សាពី Arithmetic Operator:

Arithmetic Operator គឺជាសញ្ញាគណនាជាមួយនឹង ផ្នែគណិតវិទ្យា ដើម្បីរកតំលៃលទ្ធផលនៃការគណនាណាមួយ។ វាមានដូចជា + - * / ។ ចំពោះ តំលៃ ឬ Variable ដែលត្រូវបានប្រើប្រាស់ជាមួយនឹង Operator ដើម្បធ្វើការគណនា ត្រូវបានហៅថា Operand ។



យើងអាចប្រើប្រាស់ Arithmetic Operator ទាំងអស់ជាមួយនឹងតំលៃរបស់ char, int, long, float, double, ឬ decimal។ ក្នុងនោះសញ្ញា + ក៏អាចប្រើប្រាស់បានជាមួយនឹង string បានផងដែរ។
ខាងក្រោមនេះជាឧទាហរណ៍ពីភាពខុសគ្នានៃប្រើប្រាស់សញ្ញា + :

Ex:

```
Console.WriteLine(43+1);//44
Console.WriteLine("43" + "1");//431
```

ចំពោះ Operands ដែលត្រូវបានប្រើប្រាស់ជាមួយនឹង Arithmetic Operator អាចផ្តល់លទ្ធផលខុសគ្នាទៅតាមប្រភេទតំលៃដែលត្រូវបានសរសេរ។

Ex:

```
Console.WriteLine(5/2);//2
Console.WriteLine(5.0/2.0);//2.5
Console.WriteLine(5/2.0);//2.5
```

តាមឧទាហរណ៍ខាងលើបញ្ជាក់ថា:
> នៅត្រង់ Statement ទី 1 ផ្តល់លទ្ធផលតំលៃ 2 ព្រោះ Program គិតថា 5 និង 2 គឺជាលេខចំនួនគត់ integer
> នៅត្រង់ Statement ទី 2 ផ្តល់លទ្ធផលតំលៃ 2.5 ព្រោះ Program គិតថា 5.0 និង 2.0 គឺជាលេខក្បៀស double
> នៅត្រង់ Statement ទី 3 ផ្តល់លទ្ធផលតំលៃ 2.5 ព្រោះ Program គិតថា 5 ជា integer និង 2.0 គឺជាលេខក្បៀស double ដូច្នេះនៅពេលគណនា int គឺជា Data Type មានទំហំតូចជាង double គឺត្រូវ Convert ទៅជា double ជាមុនទើបធ្វើការគណនាតាមក្រោយ។

ចំពោះ Arithmetic Operator មួយទៀតដែលត្រូវបានប្រើប្រាស់នៅក្នុង Program ដែរ គឺ Modulus Operator (%)។ ការគណនារបស់វាគឺយកសំនល់នៃការចែកណាមួយមកធ្វើជាលទ្ធផលរបស់វា។
ក្នុងភាសា C ឬ C++ មិនអនុញ្ញាតិឲ Modulus ប្រើប្រាស់បានជាមួយនឹង Floating-Point Number នោះទគឺអាចប្រើប្រាស់បានជាមួយនឹង Integer តែប៉ុណ្ណោះ។ ប៉ុន្តែចំពោះ C# វិញគឺអាចប្រើប្រាស់ជាមួយនឹង Integer ក៏បានឬ Floating-Point Number ក៏បានផងដែរ។

Ex:

```
Console.WriteLine(5.0/2.0);//2.5
Console.WriteLine(5.0%2.0);//1
```

7.<u>សិក្ខាពី Controlling Precedence</u>:

នៅក្នុង C# ចំពោះ Operator សញ្ញាមួយចំនួនដូចជា (*, /, និង %) គឺធ្វើការគណនាមុន សញ្ញា (+ និង -)។
ដូច្នេះ 2 + 3 * 4 លទ្ធផលដែលទទួលបានគឺ

Ex:

```
int i = 2 + 3 * 4;
int i = 2 + 12;
int i = 14;
```

ដើម្បីកុំឲទទួលបានលទ្ធផលខុសយើងអាចប្រើប្រាស់សញ្ញា parentheses ( ) ហ៊ុំព័ទ្ធ Expression ទាំងឡាយណាដែលត្រូវការគណនាមុន ដូច្នេះមានន័យថា សញ្ញា ( ) គឺធ្វើការគណនាមុនគេបង្អស់។

Ex:

```
int i = (2 + 3) * 4;
int i = 5 * 4;
int i = 20;
```

ចំពោះ Operator ដែលមាន Precedence ដូចគ្នាគឺវាដំនើការគណនាពីឆ្វេងទៅស្តាំតាមធម្មតា។

Ex:

```
int first = 6/2*4;//12

int second = 6+2-4;//4
```

ចំពោះ Associativity គឺជាការលំដាប់នៃការគណនាដែលក្នុងនោះ Operator ដែលមាន Precedence ដូចគ្នានោះ Associativity របស់វាគឺជា left-associative (6/2*4) មានន័យថាលំដាប់នៃការគណនាគិតចាប់ពីឆ្វេងទៅស្តាំ។

8. Assignment Operator:

Assignment Operator (=) គឺត្រូវបានប្រើប្រាស់ដើម្បីបោះតំលៃ ឬ Variable ដែលនៅខាងស្តាំទៅ Variable ដែលនៅខាងឆ្វេងវា។

Ex:

```
int myInt;
myInt = 10;
```

ក្នុងនោះយើងអាចប្រើប្រាស់ Assignment Operator ដើម្បីធ្វើការបោះតំលៃពីស្តាំទៅឆ្វេងជាមន្តបន្ទាប់ដូចឧទាហរណ៍ខាងក្រោម:

```
int myInt1;
int myInt2;
myInt2 = myInt1 = 10;
```

9. Incrementing and Decrementing Variables:

ប្រសិនបើយើងត្រូវការបន្ថែម តំលៃ 1 ទៅឲ Variable នោះយើងអាចប្រើសញ្ញា + Operator:

Ex:

```
count = count + 1;
```

C# បានផ្តល់នូវ Operator មួយសំរាប់បន្ថែមតំលៃ 1 ទៅឲ Variable ខ្លួនវា ដោយយើងត្រូវប្រើប្រាស់ សញ្ញា ++ នៅខាងក្រោយ Variable នោះ ។

Ex:

```
count++;
```

ក្នុងនោះយើងក៏អាចប្រើប្រាស់សញ្ញា -- ដើម្បីបន្ថយតំលៃ 1 ចេញពី Variable បានផងដែរ។

Ex:

```
count--;
```

ចំពោះ -- Operator និង ++ Operator ត្រូវបានហៅថា Unary Operator ។

10. Prefix and Postfix:

Increment ++ និង decrement – Operator គឺនឹងផ្តល់តំលៃខុសគ្នានៅពេលដែលយើងដាក់វានៅខាងមុងឬខាងក្រោយ Variable ។ក្នុងការដាក់សញ្ញានៅខាងមុខ Variable ត្រូវបានហៅថា prefix form ចំនែកការដាក់សញ្ញានៅខាងក្រោយ Variable ត្រូវបានហៅថា postfix form ។

Ex:

```
count++; //postfix increment
++count; //prefix increment
count--; //postfix decrement
--count; //prefix increment
```

ខាងក្រោមនេះជាលទ្ធផលខុសគ្នានៃការប្រើប្រាស់ ++x ជាមួយនឹង x++

Ex:

```
int x;
x = 42;
Console.WriteLine(x++); //x is now 43, 42 written out
Console.WriteLine(++x); //x is now 43, 43 written out
```

តាមរយៈឧទាហរណ៍ខាងលើបង្ហាញថា:

x++ គឺវាធ្វើការមុននឹងបន្ថែមតំលៃមានន័យថាវាផ្តល់ទៅឲ Console.WriteLine តំលៃចាស់ 42 រួចទើបបន្ថែមតំលៃទៅឲខ្លួនវាស្មើ 43។

++x គឺវាបន្ថែមតំលៃមុននឹងវាធ្វើការមានន័យថាបន្ថែមតំលៃ 1 ទៅឲខ្លួនវាស្មើ 43 រួចទើបផ្តល់ទៅឲ Console.WriteLine តំលៃថ្មី 43 ដែរ។

11. Declaring Implicitly Typed Local Variables:

យើងអាចធ្វើការ initialize Variable នៅលើត្រង់ Statement តែមួយក៏បាន ដូចឧទាហរណ៍ខាងក្រោម:

Ex:

```
int myInt = 99;
```

ឧទាហរណ៍ខាងលើមានន័យថាគឺជាការបង្កើត Variable មួយឈ្មោះ myInt មាន Data Type ជា int ហើយក្នុងនោះ យើងបានផ្តល់តំលៃ 99 ភ្លាមៗទៅវា។ សូមចងចាំថាយើងត្រូវផ្តល់តំលៃ Variable ទៅតាមប្រភេទទិន្នន័យដែលវាត្រូវទទួលយកផងដែរ។

លក្ខណៈពិសេសរបស់ C# គឺយើងអាចវាជ្រើសរើសយក Data Type ណាមួយដែលសាកសមទៅនឹង Variable ដែលយើងបង្កើតបានផងដែរ ដោយវាធ្វើការត្រួតពិនិត្យទៅលើតំលៃដែលបោះទៅ Variable នោះ។

Ex:

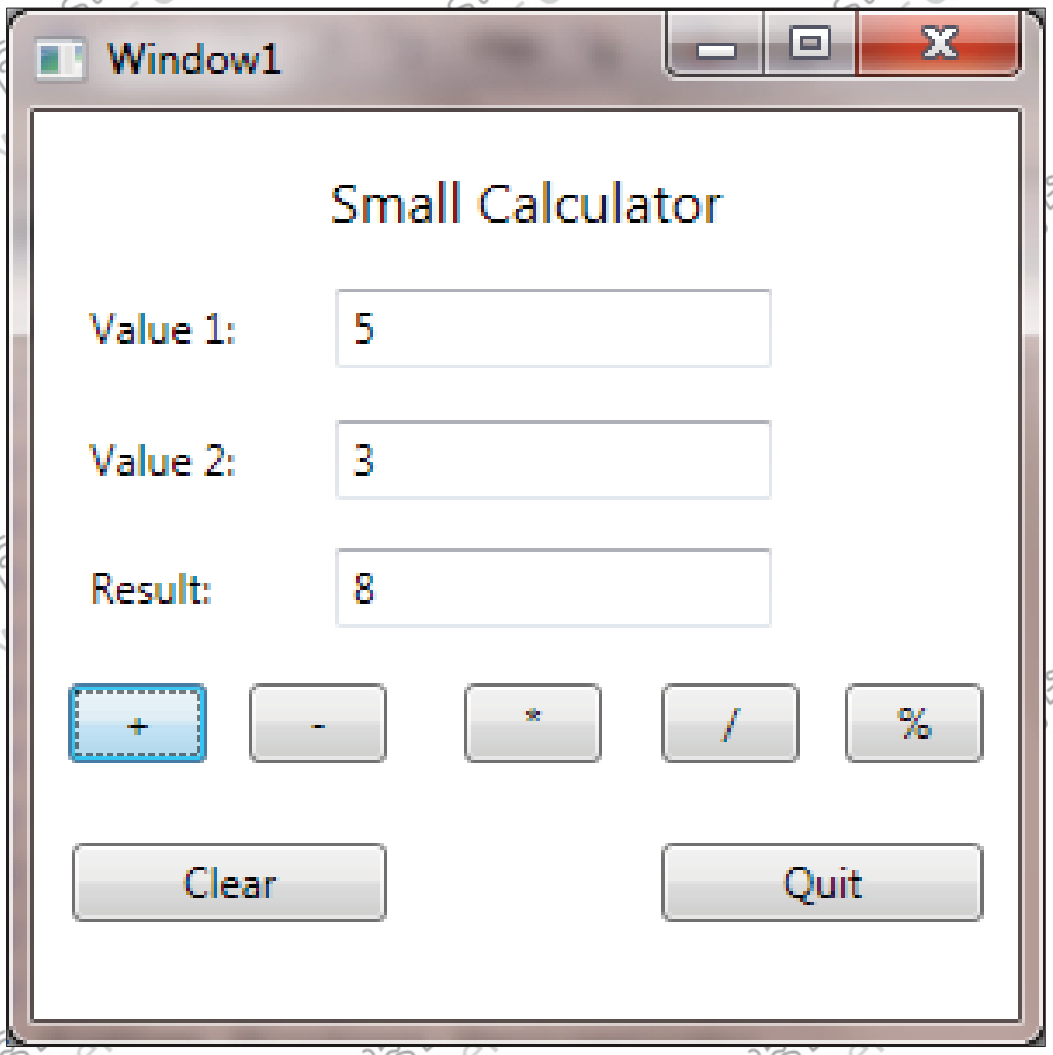
```
var myVariable = 99;
var myOtherVariable="Hello";
```

តាមឧទាហរណ៍ខាងលើ Variable myVariable និង myOtherVariable គឺត្រូវបានហៅថា implicitly typed variables។ ចំពោះ var Keyword គឺត្រូវបានប្រើប្រាស់ដើម្បីប្រាប់ទៅ compiler ឲជ្រើសរើសយក Data Type ដ៏ត្រឹមត្រូវមួយសំរាប់ Variable ទៅតាមតំលៃដែលផ្តុក។ ដូច្នេះយើងបាន myVariable គឺជា int ចំនែក myOtherVariable គឺជា String ។ ហើយនៅពេលក្រោយទៀត យើងមិនអាចធ្វើការផ្តល់តំលៃផ្សេងៗទៀតដូចជា float, double, ឬ string ទៅឲ myVariable បានទៀតឡើយ។

ចំពោះ Variable ទាំងឡាយណាដែលប្រើប្រាស់ជាមួយ var Keyword ដាច់ខាតត្រូវតែ assign តំលៃឲវាភ្លាមៗ បើមិនដូច្នេះទេ នឹងមាន Error កើតឡើង។

12. <u>លំហាត់</u>:

ចូរសរសេរ code មួយដើម្បី display ព័ត៌មានមួយចំនួនដូចខាងក្រោម:



មេរៀនទី 3: ការបង្កើត Methods និង Applying Scope

1. <u>Declaring Methods:</u>

Methods គឺជាបណ្តុំនៃ Statement ដែលជាមានតួនាទីធ្វើការងារជាក់លាក់ណាមួយ ហើយត្រូវបានហៅឲធ្វើការ (execute) នៅពេលដែលវាត្រូវបានហៅនៅក្នុងចំនុចណាមួយនៃ Program ។ Methods ត្រូវបានបង្កើតដើម្បីកាត់បន្ថយនូវការសរសេរកូដ ដដែលៗពីលើចុះក្រោម ។

Methods នីមួយៗតែងតែមាន Name និង Body ដែល Name គឺជាឈ្មោះរបស់ Method ត្រូវកំនត់ដូចនឹងក្បួននៃឈ្មោះ Variable ដែរ ចំនែក Body គឺជា Statements ដែលធ្វើការងារណាមួយនៅពេលដែលវាត្រូវបានហៅយកទៅប្រើប្រាស់។

ខាងក្រោមនេះជា Syntax របស់ Microsoft C# Method:

```
returnType methodName ( parameterList )
{
    // method body statements go here
}
```

> returnType គឺជា data type ឬជាប្រភេទទិន្នន័យដែល function ត្រូវ return ជាលទ្ធផលត្រឡប់ទៅវិញ ដែលអាចមានដូចជា string ឬ int ហើយប្រសិនបើយើងសរសេរ Method ដែលមិនត្រូវការ return តំលៃនោះ ត្រូវប្រើប្រាស់ Keyword void នៅត្រង់តំបន់ returnType ។ ចំពោះ var Keyword គឺមិនអាចដាក់ជា returnType របស់ Method ឡើយ។

> MethodName គឺជា ឈ្មោះរបស់ Method (ទំរង់នៃការកំនត់ឈ្មោះមានលក្ខណៈដូចនឹង Variable)

> parameterList គឺជា Variable ដែលមាន Data Type ផ្សេងៗ ជាបន្តបន្ទាប់នៅក្នុង Method ដែលវា មានតួនាទីសំរាប់ទទួលយក arguments ដែលបាន Pass ដែលពេល Method ត្រូវបានហៅ (Call) ។ក្នុង Method អាចមានឬគ្មាន Parameter ប៉ុន្តែប្រសិនបើមាន Parameters ច្រើននោះត្រូវ ខណ្ឌចែកដោយប្រើប្រាស់សញ្ញា comma ។

> Method Body គឺជា statements នីមួយៗធ្វើការងារ នៅពេលដែល Method ត្រូវបានហៅ ហើយវាស្ថិតនៅចន្លោះ braces ({ }) ជានិច្ច។

```
int addValues(int leftHandSide, int rightHandSide)
{
      Return leftHandSide + rightHandSide;
}
```

2. <u>Calling Methods</u>:

Methods គឺត្រូវបង្កើតឡើងសំរាប់ ហៅ (Call) យកទៅប្រើប្រាស់នៅពេលក្រោយ ហើយប្រសិនបើ Method ត្រូវការដូចជា Parameter មួយចំនួននោះ យើងត្រូវធ្វើដាក់ឲត្រឹមត្រូវទៅតាមចំនួន និង Data Type របស់វាផងដែរ។ ក្នុងនោះប្រសិនបើ Method មាន return នោះគឺត្រូវបង្កើត Variable ឲមាន Data Type ដូចគ្នាដើម្បីទទួលយកលទ្ធផលដែលវាបាន return មក។

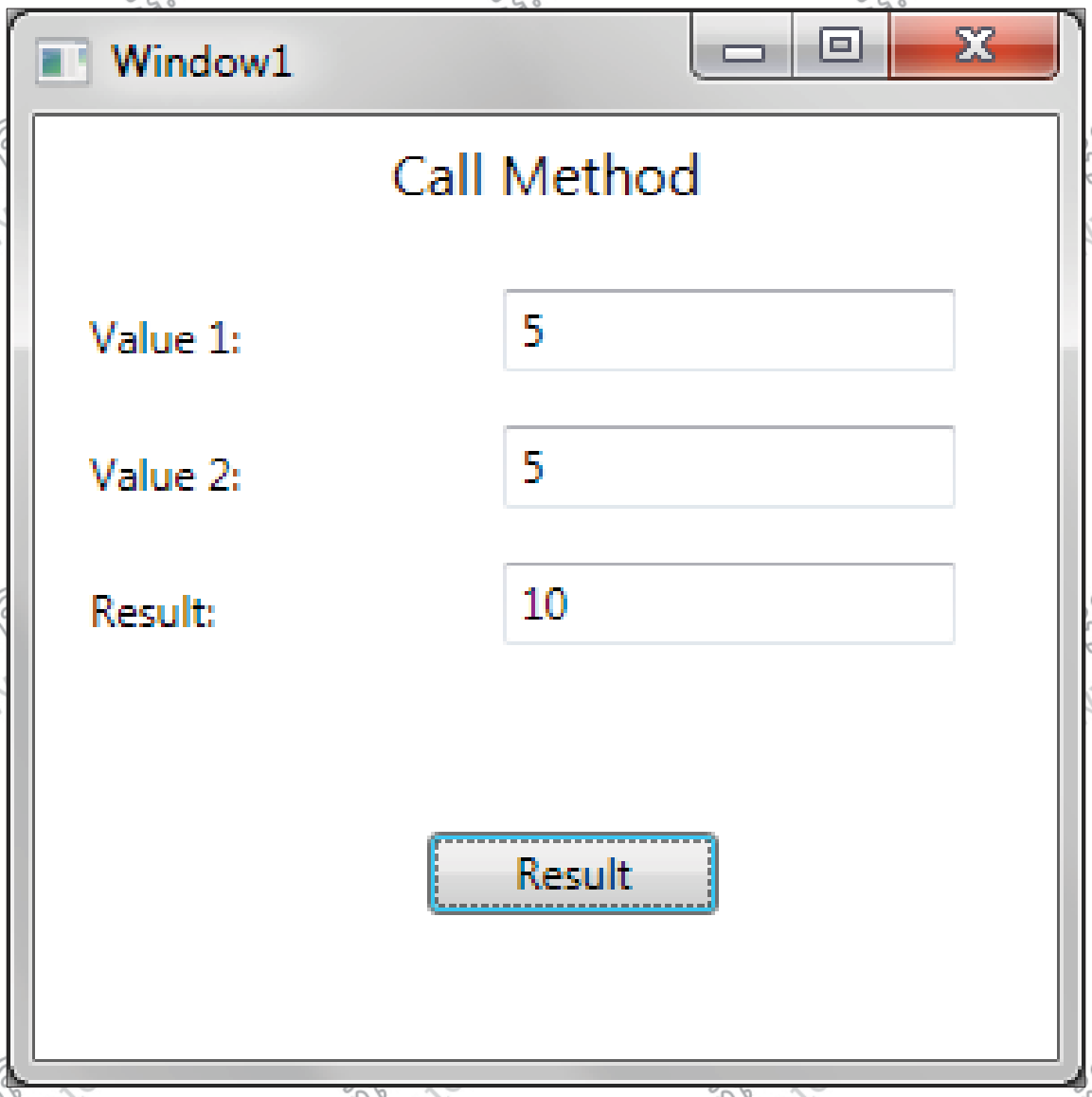
```
result = methodName( argumentList);
```

> methodName គឺជាឈ្មោះរបស់ Method ដែលត្រូវ Call យកមកធ្វើការ។

> result = គឺជា Variable ដែល store លទ្ធផលនៃការការងាររបស់ Method ហើយប្រសិនបើ Method ជា void នោះ Variable មិនចាំបាច់ប្រើប្រាស់ដើម្បី store លទ្ធផលរបស់ Method ឡើយ។

> ArgumentList គឺជា តំលៃ ឬ Variable សំរាប់ Pass ទៅឲ Method ដើម្បីយកទៅធ្វើការ ដោយ តំលៃ ឬ Variable ដែលបានដាក់ទៅឲត្រូវមាន Data Type ដូចគ្នានឹង Method ដែលបានកំនត់ ហើយចំនួន Arguments ដែលបានដាក់ទៅឲ ត្រូវដូចគ្នានឹងចំនួន ParameterList ដែលមានក្នុង Method ផងដែរ។

Ex:



```
// Declaring Method
private int addValue(int x, int y)
        {
            int sum;
            sum = x + y;
            return sum;
        }

// Calling Method
private void button1_Click(object sender, RoutedEventArgs e)
        {
            int a = int.Parse(textBox1.Text);
            int b = int.Parse(textBox2.Text);
            int result = addValue(a, b);
            textBox3.Text = result.ToString();
        }
```

3. <u>Applying Scope</u>:

Scope គឺជាការកំនត់ព័ទំហំទីតាំងនៃ Variable ដែលអាចប្រើប្រាស់បាននៅក្នុង Class ដែលក្នុងនោះត្រូវបានបែងចែកជា 2 ប្រភេទរួមមាន Local Scope និង Class Scope ។

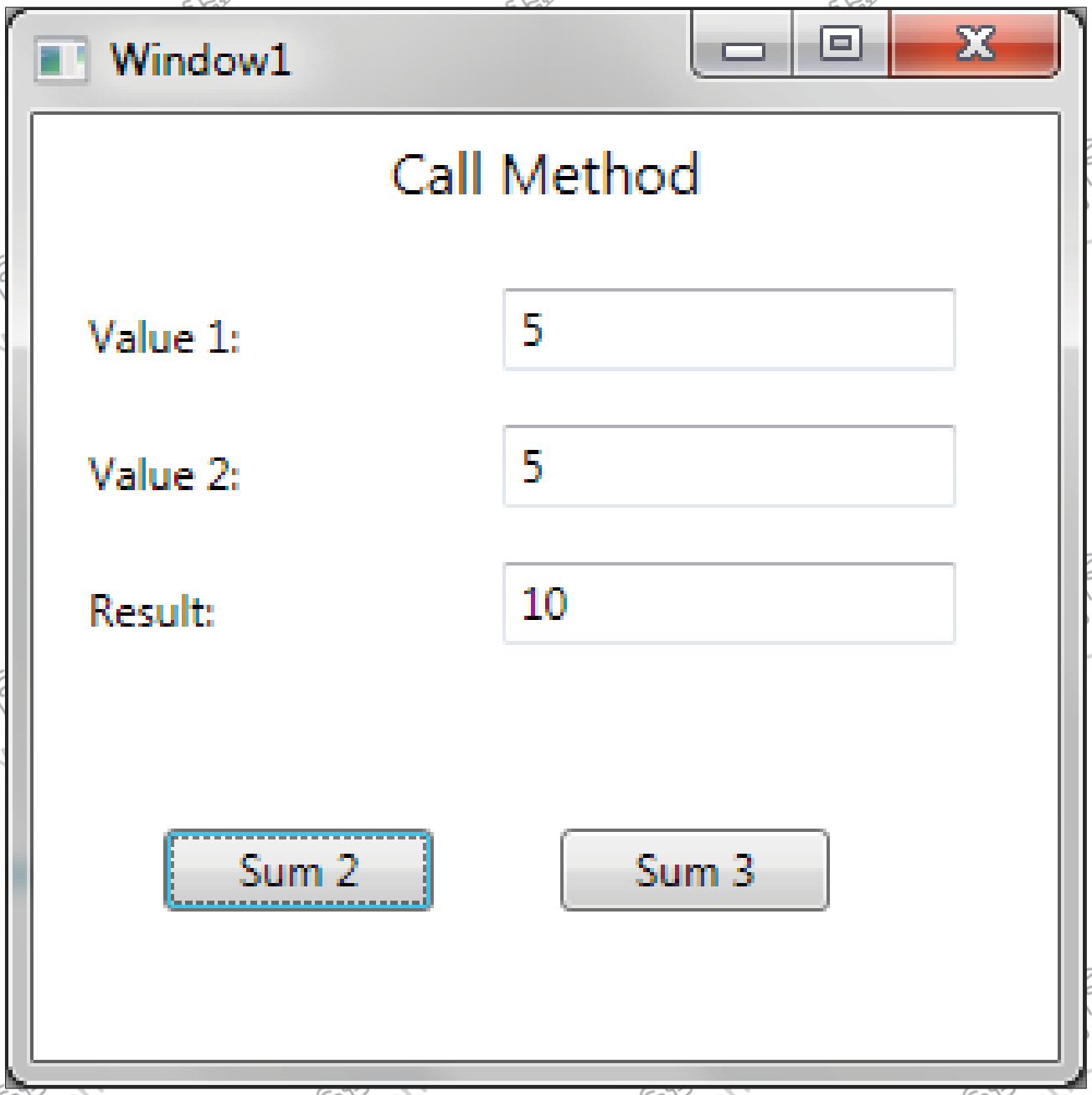
> Local Scope: គឺជា Variable ដែលត្រូវបានប្រកាសនៅក្នុង Body នៃ Method ឬស្ថិតនៅក្នុង Braces { } របស់ Method ណាមួយ ដែលអាចប្រើប្រាស់នៅក្នុងតំបន់របស់ Method នោះតែប៉ុណ្ណោះ ។

> Class Scope: គឺជា Variable ដែលត្រូវបានប្រកាសនៅក្នុង Body នៃ Class ឬស្ថិតនៅក្នុង Braces { } របស់ Class ដែលមានលទ្ធភាពអាចប្រើប្រាស់បានៅក្នុង Class និង Methods ដទៃទៀតដែលស្ថិតនៅក្នុង Class បានផងដែរ ។

```
int grandFather = 9;
private void testMethod()
{
            grandFather = 8;
            int Father = 5;
}
private void Window_Loaded(object sender, RoutedEventArgs e)
{
            //Correct
             MessageBox.Show(grandFather.ToString());

             //Incorrect
            MessageBox.Show(Father.ToString());
}
```

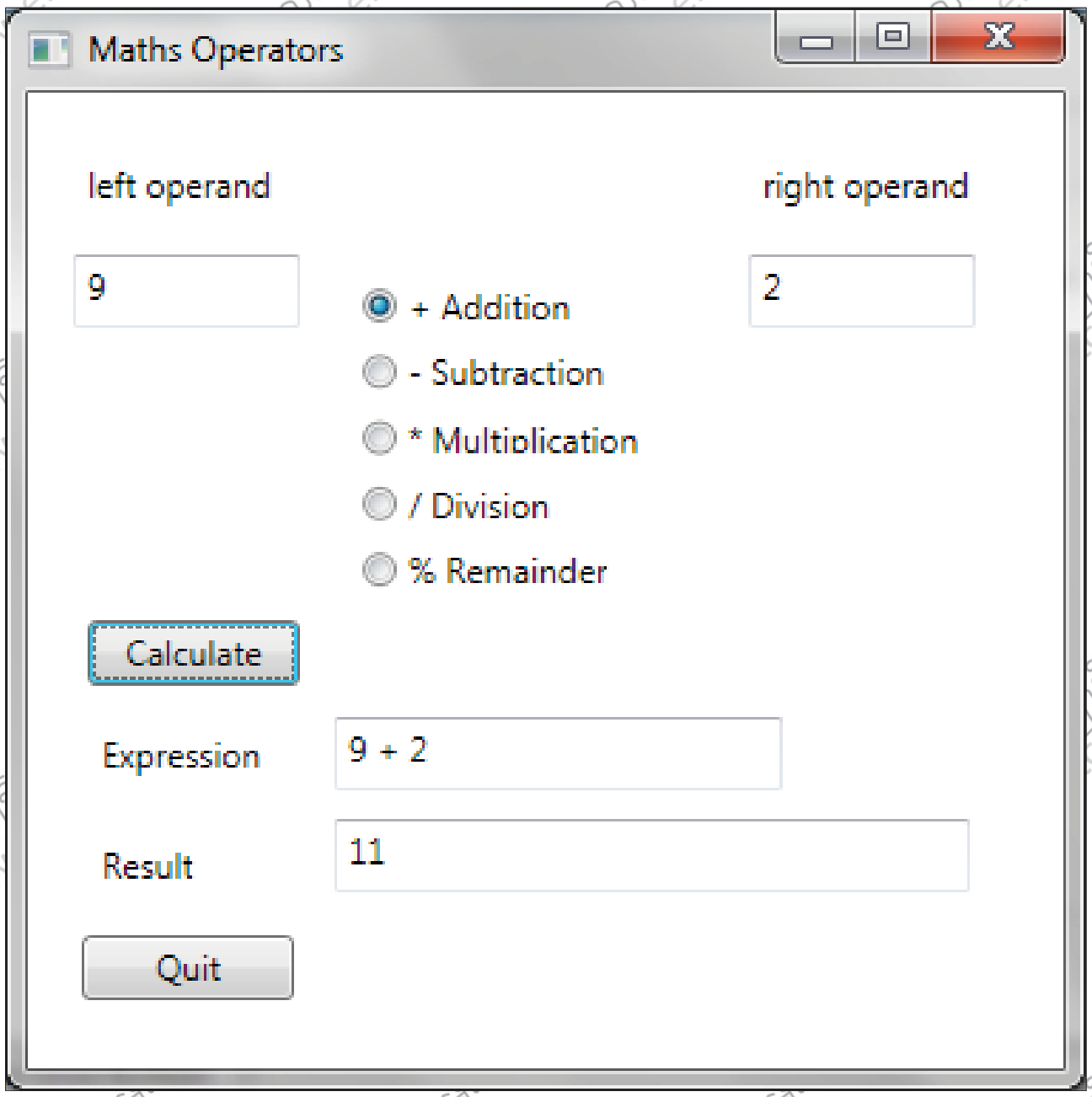
4. <u>Overloading Methods</u>:

Overload Methods គឺជាការបង្កើត Methods ចាប់ពី 2 ឡើងទៅ ដោយមានឈ្មោះដូចគ្នាប៉ុន្តែមាន Data Type ខុសៗគ្នា ឬមានចំនួន Parameters ខុសៗគ្នា។



```
private int addValue(int x, int y)
        {
            int sum;
            sum = x + y;
            return sum;
        }
        private int addValue(int x, int y,int z)
        {
            int sum;
            sum = x + y+z;
            return sum;
        }
private void button1_Click(object sender, RoutedEventArgs e)
        {
            int a = int.Parse(textBox1.Text);
            int b = int.Parse(textBox2.Text);
            int result = addValue(a, b);
            textBox3.Text = result.ToString();
        }
private void button2_Click(object sender, RoutedEventArgs e)
        {
            int a = int.Parse(textBox1.Text);
            int b = int.Parse(textBox2.Text);
            int c = int.Parse(textBox3.Text);
            int result = addValue(a, b,c);
            textBox3.Text = result.ToString();
        }
```

5. **លំហាត់:**


Maths Operators
left operand
right operand
9
2
+ Addition
- Subtraction
* Multiplication
/ Division
% Remainder
Calculate
Expression
9 + 2
Result
11
Quit

មេរៀនទី 4: ការប្រើប្រាស់ Decision Statement

1. Boolean Variable:

Boolean Variable គឺ Variable ដែល store តំលៃតែពីរប៉ុណ្ណោះគឺ true និង false ។

Ex:

```
bool areYouReady;
areYouReady = true;
Console.WriteLine(areYouReady); // writes True
```

ក្នុងនោះយើងអាចបន្ថែមការប្រើប្រាស់ជាមួយនឹង Boolean Operator ដើម្បីធ្វើការគណនារកមើលតំលៃ true ឬ false បន្ថែមទៀតបានផងដែរ។

```
bool areYouReady;
areYouReady = true;
Console.WriteLine(!areYouReady); // writes False
```

2. Equality Operators:

យើងអាចប្រើប្រាស់ Equality Operators ចំនួនពីរដែលមានដូចជា equality (==) និង inequality (!=) ដើម្បីត្រួតពិនិត្យរកមើល ថាតើ Variable ឬ Expression ទាំងពីរពិតជាមានតំលៃដូចគ្នាដែរឬទេ។

Operator	Meaning	Example	Outcome if age is 42
==	Equal to	age == 100	False
!=	Not equal to	age !=0	True

3. Relational Operators:

ចំពោះ Relational Operators វិញ គឺអាចធ្វើការត្រួតពិនិត្យរកមើល ថាតើ Variable ឬ Expression ទាំងពីរពិតជាមានតំលៃដូចគ្នាដែរឬទេ ដោយផ្អែកទៅលើ Operator ចំនួន 4 ដូចខាងក្រោម:

Operator	Meaning	Example	Outcome if age is 42
<	Less than	age < 21	False
<=	Less than or equal to	age <=18	False
>	Greater than	age > 16	True
>=	Greater than or equal to	age >=30	True

លក្ខណៈខុសគ្នារវាង សញ្ញា = ជាមួយនឹង សញ្ញា == :

> សញ្ញា = មានន័យថាគឺជាការ assign ខាងស្តាំតំលៃទៅឲ Variable នៅខាងឆ្វេង

Ex: x = 5

មានន័យថាបោះតំលៃ 5 ចូលទៅក្នុង Variable x

> សញ្ញា == មានន័យថាគឺជាការប្រៀបធៀបតំលៃរបស់ Variable ដែលនៅខាងឆ្វេងជាមួយនឹង Variable ខាងស្តាំ ដើម្បីស្វែងរកលទ្ធផល True ឬ False ។

Ex: x==5

មានន័យថាប្រៀធៀបរវាង x ជាមួយនឹង 5 ថាតើ x មានតំលៃជាលេខ 5 ពិតមែនដែរឬទេ?

4. <u>Condition Logical Operators</u>:

Condition Logical Operator ដែលមានដូចជា And Operator (&&) និង Or Operator (||) ដែលវាត្រូវបានប្រើប្រាស់ដើម្បីភ្ជាប់ជាមួយនឹង Comparison Operator ឬប្រើក្នុងលក្ខណៈដទៃទៀត ដើម្បីធ្វើការ ស្វែងរកលទ្ធផលពិត ឬមិនពិត។

<u>ការប្រៀបធៀបអំពី && (And)</u>:

តំលៃទី 1	ឈ្នាប់	តំលៃទី 2	លទ្ធផល
True (ពិត)	**&&**	**True (ពិត)**	**True (ពិត)**
True (ពិត)	&&	False (មិនពិត)	False (មិនពិត)
False (មិនពិត)	&&	True (ពិត)	False (មិនពិត)
False (មិនពិត)	&&	False (មិនពិត)	False (មិនពិត)

Ex:

```
Int percent = 55;
bool validPercentage;
validPercentage = (percent >= 0) && (percent <= 100);
Console.WriteLine(validPercentage); // writes True
```

<u>ការប្រៀបធៀបអំពី || (Or)</u>:

តំលៃទី 1	ឈ្នាប់	តំលៃទី 2	លទ្ធផល
True (ពិត)	**||**	**True (ពិត)**	**True (ពិត)**
True (ពិត)	||	False (មិនពិត)	True (ពិត)
False (មិនពិត)	||	True (ពិត)	True (ពិត)
False (មិនពិត)	||	False (មិនពិត)	False (មិនពិត)

Ex:

```
Int percent = 55;
bool validPercentage;
validPercentage = (percent < 0) && (percent > 100);
Console.WriteLine(validPercentage); // writes False
```

5. Operator Precedence and Associativity:

Table ខាងក្រោមនេះបង្ហាញពីលំដាប់នៃដំនើការការងាររបស់ Operator នីមួយៗដែលក្នុងនោះ លំដាប់នៃដំនើការការងាររបស់វាគឺមានអាទិភាព ចាប់ពីលើចុះក្រោម ហើយ Operators ទាំងឡាយណាដែលនៅក្នុងក្រុមជាមួយគ្នា គឺដំនើការពីឆ្វេងទៅស្តាំ។

Category	Operators	Description	Associativity
Primary	( ) ++ --	Precedence override Post-increment Post-decrement	Left
Unary	! + - ++ --	Logical NOT Addition Subtraction Pre-increment Pre-decrement	Left
Multiplicative	* / %	Multiply Divide Division remainder (modulus)	Left
Additive	+	Addition Subtraction	Left
Relational	< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	Left
Equality	== !=	Equal to Not equal to	Left
Conditional AND Conditional OR	&& ||	Logical AND Logical OR	Left Left
Assignment	=		Right

6. If Statement:

If Statement គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការឬ execute នូវ block នៃ code នៅពេលដែលលក្ខខណ្ឌរបស់វាពិត។

```
if ( booleanExpression )
        statement-1;
else
        statement-2;
```

> if គឺជា Keyword សំរាប់ប្រើប្រាស់ដើម្បីដាក់លក្ខខណ្ឌនៅក្នុង Source Code

> booleanExpression គឺជា តំលៃ ឬ expression ដែលប្រើភ្ជាប់ជាមួយនឹង Comparison Operator ដើម្បីស្វែងរកលទ្ធផល True

> statement-1 គឺជាបណ្តុំនៃ Code ដែលត្រូវធ្វើការនៅពេលដែល condition ទទួលបានតំលៃ True ប៉ុន្តែបណ្តុំនៃ Code នឹងត្រូវបានរំលងចោល ប្រសិនបើ booleanExpression នៅក្នុង if ផ្តល់លទ្ធផល False វិញ។
> else គឺជា Keyword សំរាប់ប្រើប្រាស់ដើម្បីដាក់លក្ខខណ្ឌនៅក្នុង Source Code បន្ទាប់ពី if ដើម្បីធ្វើការ នៅពេលដែល if ទទួលបាន booleanExpression ជាតំលៃ False ។
> statement-2 គឺជាបណ្តុំនៃ Code ដែលត្រូវធ្វើការនៅពេលដែល booleanExpression ទទួលបានតំលៃ False ប៉ុន្តែបណ្តុំនៃ Code នឹងត្រូវបានរំលងចោល ប្រសិនបើ booleanExpression នៅក្នុង if ផ្តល់លទ្ធផល True វិញ។
ចំពោះ else Keyword គឺ Optional មានន័យថា ពុំចាំបាច់ដាក់បន្ទាប់ពី if ក៏បាន។

Ex:

```
int seconds;
if (seconds == 59)
      seconds = 0;
else
seconds++;
```

ប្រសិនបើយើងប្រើ Boolean Variable ធ្វើជា booleanExpression វិញនោះ យើងអាចប្រើទំរង់កាត់ដូចខាងក្រោម:

Ex:

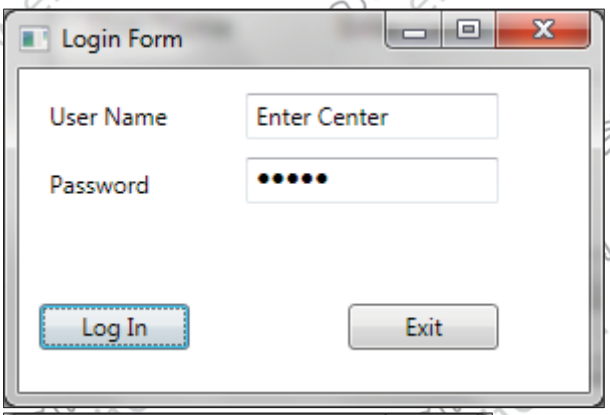
```
bool inWord;
...
if (inWord == true) // ok, but not commonly used
...
if (inWord) // better
```

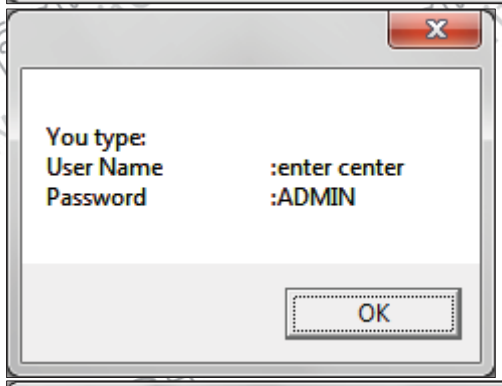
ក្នុងករណីដែល statement ក្នុង if ត្រូវបានសរសេរចាប់ពី 2 ជួរឡើងទៅគឺយើងអាចប្រើប្រស់ braces { } ដើម្បីកំនត់ Block នៃ Code ដែលត្រូវធ្វើការ។

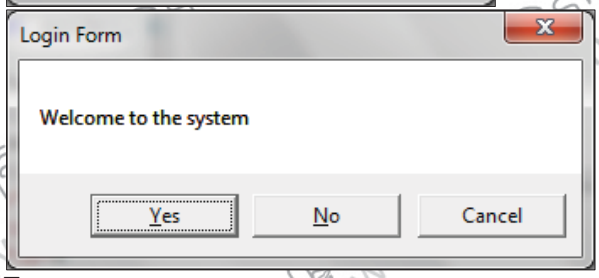
Ex:

```
int seconds = 0;
int minutes = 0;
...
if (seconds == 59)
{
      seconds = 0;
      minutes++;
}
else
      seconds++;
```

7. <u>ការបង្កើត Login Form</u>:







Ex:

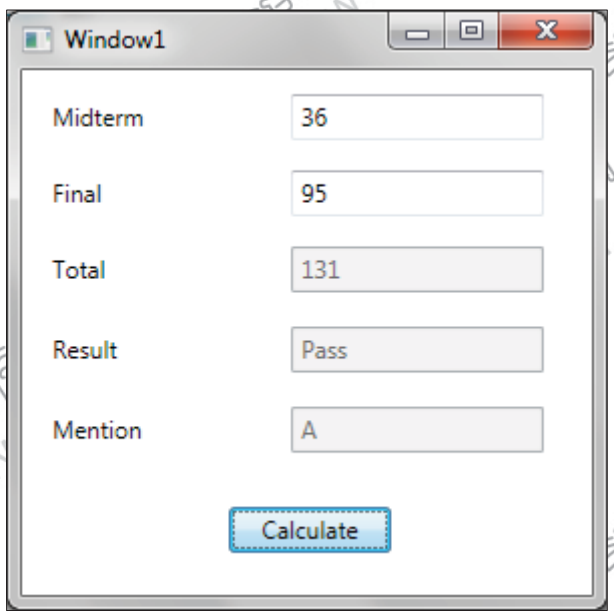
```
private void button1_Click(object sender, RoutedEventArgs e)
{
   string userName = textBox1.Text;
   userName = userName.ToLower();
   string password = passwordBox1.Password;
   password = password.ToUpper();
   MessageBox.Show("You type:\n"+
                           "User Name \t:"+
                           userName + "\n"+
                           "Password \t\t:"+
                           password);
   if (userName == "enter center" && password == "ADMIN")
          MessageBox.Show("Welcome to the system", "Login Form",
          MessageBoxButton.YesNoCancel);
   else
         MessageBox.Show("Incorrect userName or Password");
          this.Close();
}
private void button2_Click(object sender, RoutedEventArgs e)
{
   this.Close();
}
```

8. Cascading If Statement:

Ex:

```
if (day == 0)
      dayName = "Sunday";
else if (day == 1)
      dayName = "Monday";
else if (day == 2)
      dayName = "Tuesday";
else if (day == 3)
      dayName = "Wednesday";
else if (day == 4)
      dayName = "Thursday";
else if (day == 5)
      dayName = "Friday";
else if (day == 6)
      dayName = "Saturday";
else
      dayName = "unknown";
```

9. Cascade If Statement Practical:



```
private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            textBox3.IsEnabled = false;
            textBox4.IsEnabled = false;
            textBox5.IsEnabled = false;
        }
private void button1_Click(object sender, RoutedEventArgs e)
        {
            float midterm, final,total;
            midterm = float.Parse(textBox1.Text);
            final = float.Parse(textBox2.Text);

            total = midterm + final;
            textBox3.Text = total.ToString();
            if (total >= 50)
                textBox4.Text = "Pass";
            else
                textBox4.Text = "False";

            if (total >= 90)
                textBox5.Text = "A";
            else if (total >= 80)
                textBox5.Text = "B";
            else if (total >= 70)
                textBox5.Text = "C";
            else if (total >= 60)
                textBox5.Text = "D";
            else if (total >= 50)
                textBox5.Text = "E";
            else
                textBox5.Text = "F";
        }
```

10. Switch Statement:

switch Statement គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការឬ execute នូវ block នៃ code នៅពេលដែលលក្ខខណ្ឌរបស់វាពិតដែលវាមានលក្ខណៈដូចគ្នាទៅនឹង if ដែរ ។

Syntax:

```
switch (controllingExpression)
{
    case constantExpression1:
        statements1;
        break;
    case constantExpression2:
        statements2;
        break;
    ---------------
    default:
        statements n;
        break;
}
```

```
switch (day)
{
case 0:
      dayName = "Sunday";
      break;
case 1:
      dayName = "Monday";
      break;
case 3:
      dayName = "Tuesday";
      break;
case 4:
      dayName = "Wednesday";
      break;
case 5:
      dayName = "Thursday";
      break;
```

```
case 6:
      dayName = "Friday";
      break;
case 7:
      dayName = "Saturday";
      break;
default:
      dayName = "Unknown";
      break;
}
```

11. <u>Switch Statement Rule</u>:

switch Statement មានភាពងាយស្រួលក្នុងការប្រើប្រាស់ ប៉ុន្តែដើម្បីមានភាពច្បាស់លាស់ក្នុងការប្រើប្រាស់ យើងត្រូវគោរពតាមគោលការណ៍មួយចំនួនដូចខាងក្រោម:

> យើងត្រូវប្រើប្រាស់ Switch ជាមួយនឹង primitive data types ដូចជា int ឬ string តែម៉ុណ្ណោះ ដោយមិនអាចប្រើប្រាស់ជាមួយនឹង data type ផ្សេងទៀតដូចជា float ឬ double បានឡើយ។

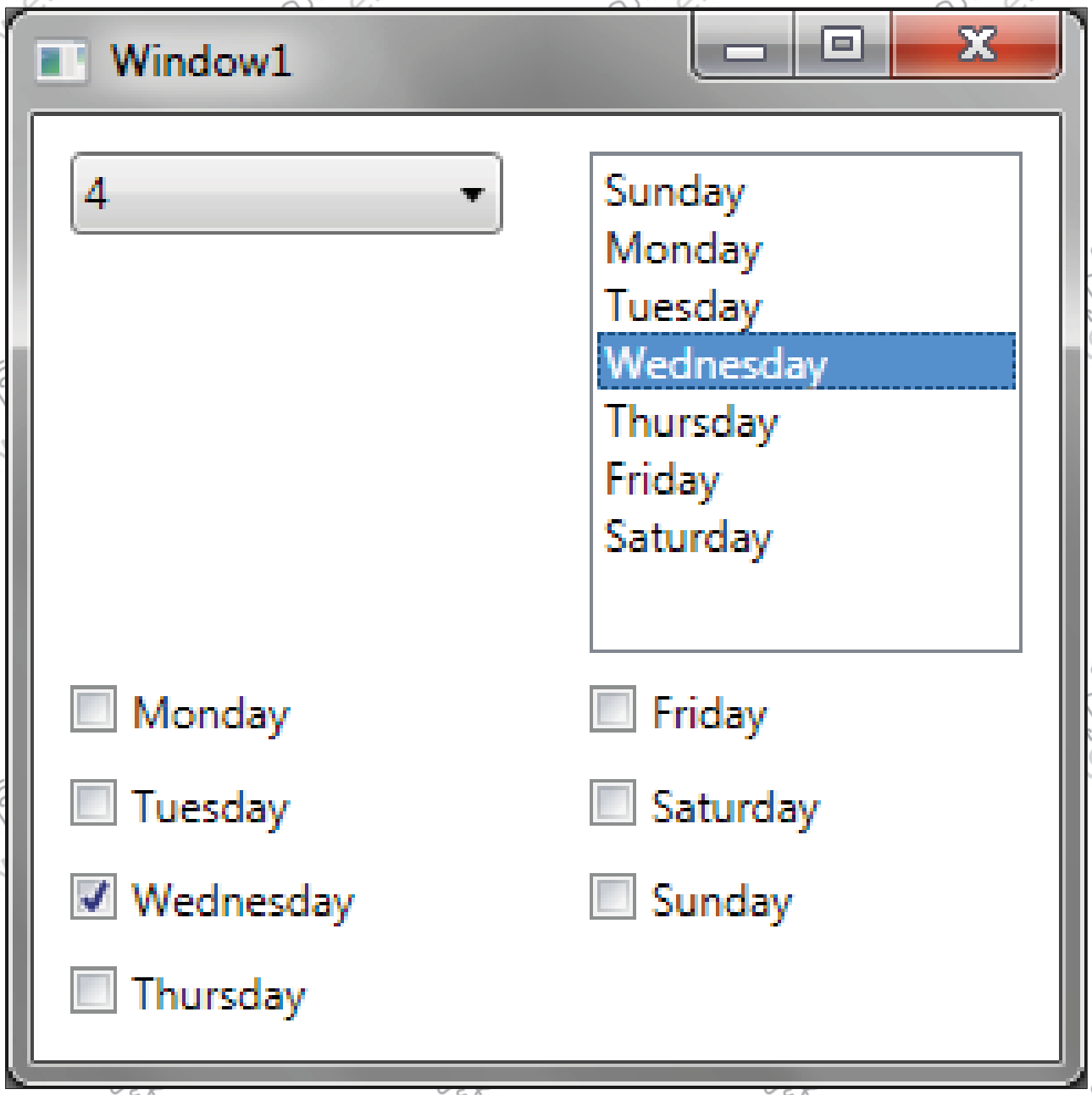
> Case Labels គឺត្រូវតែជា constant expression មានន័យថា expression ត្រូវមានតំលៃពិតប្រាកដសំរាប់ប្រើប្រាស់ជាមួយនឹង Case Labels ។

> យើងអាចប្រើប្រាស់ Case Labels ពីរឬច្រើនដើម្បីផ្តៀតផ្តាត់រកលក្ខខណ្ឌដើម្បីបង្ហាញលទ្ធផលដូចគ្នា

Ex:

```
switch (trumps)
{
case Hearts :
case Diamonds : // Fall-through allowed - no code between labels
      color = "Red"; // Code executed for Hearts and Diamonds
      break;
case Clubs :
      color = "Black";
case Spades : // Error - code between labels
      color = "Black";
      break;
}
```

12. ឈ្មោះ:


Window1
4
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday

មេរៀនទី 5: ការប្រើប្រាស់ Compound Assignment និង Iteration Statements

1. <u>Compound Assignment Operators</u>:

ក្នុងការ add តំលៃបន្ថែមទៅឲ Variable គឺយើងត្រូវធ្វើការយក Variable ដដែលរួចយកបន្ថែមជាមួយនឹង តំលៃ ផ្សេងទៀត ឬ Variable ណាមួយផ្សេងទៀត។

Ex:

```
int x=42;
x= x + 5;
Console.WriteLine(x); // 47
```

ក្នុងនោះយើងអាចធ្វើការប្រើប្រាស់ជាមួយនឹង Compound Assignment Operator ដើម្បីធ្វើការជួយសំរួលដល់ការសរសេរឲលឿនជាងមុនបានផងដែរ។

Ex:

```
int x=42;
x += 5;
Console.WriteLine(x); // 47
```

ខាងក្រោមនេះជា Compound Assignment Operator សំរាប់បង្កើតជាការគណនាផ្សេងៗ:

Full Form Operator	Compound Assignment Operator
x = x + 5	x += 5
x = x - 5	x -= 5
x = x * 5	x *= 5
x = x / 5	x /= 5
x = x % 5	x %= 5

យើងក៏អាចធ្វើការយក Compound Assignment មួយគឺ += សំរាប់ប្រើប្រាស់ដើម្បីធ្វើការបន្ថែម ទិន្នន័យប្រភេទ string ផងដែរ។

Ex:

```
string name = "John";
string greeting = "Hello ";
greeting += name;
Console.WriteLine(greeting);// Hello John
```

2. While Statement:

while គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការដំនើការនូវ Block នៃ Code ដដែលៗ នៅពេលដែលលក្ខខណ្ឌរបស់ True ។ មុននឹងវាដំនើការទៅលើ Block នៃ Code គឺវាឆែកមើលលក្ខខណ្ឌជាមុនសិន ប្រសិនបើ True ធ្វើ ប៉ុន្តែបើ False គឺមិនធ្វើ សូម្បីតែមួយដងក៏ដោយ។

Syntax:

```
while (booleanExpression)
{
    statement;
}
```

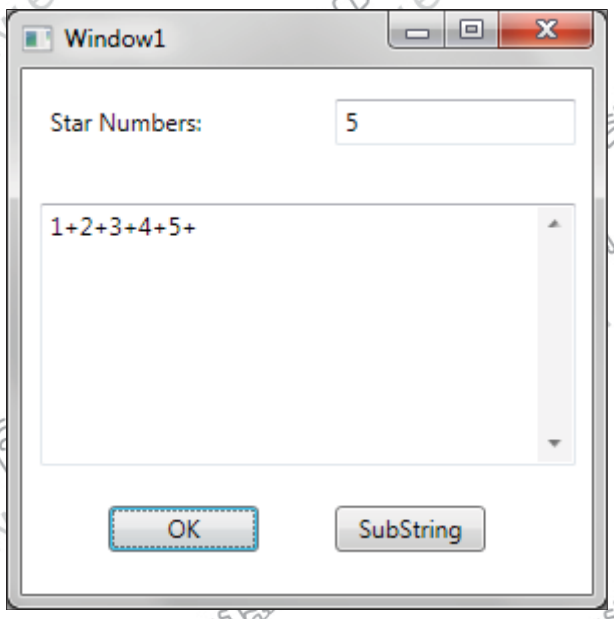
Ex:

```
int i = 0;
while (i < 10)
{
    Console.WriteLine(i);
    i++;
}
```

Output:

```
0
1
2
3
4
5
6
7
8
9
Press any key to continue . . .
```

3. While Statement Practical:




```
string store = "";
     private void button1_Click(object sender, RoutedEventArgs e)
     {
                int number = int.Parse(textBox1.Text);

                int i = 1;
                while (i <= number)
                {
                    store = store + i + "+";
                    i++;
                }
                textBox2.Text = store;


      }
     private void button2_Click(object sender, RoutedEventArgs e)
      {
                string cut = store.Substring(0,store.Length-1);
                textBox2.Text = cut;
      }
```

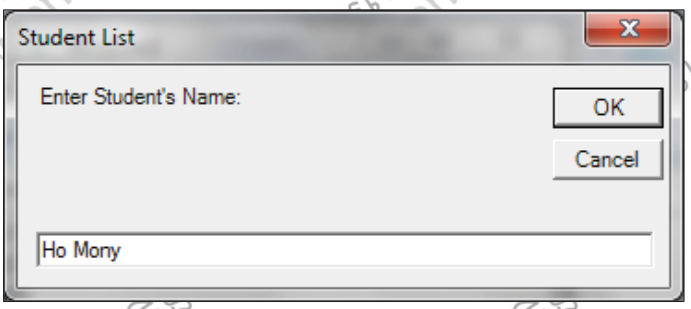
4. <u>Do Statement</u>:

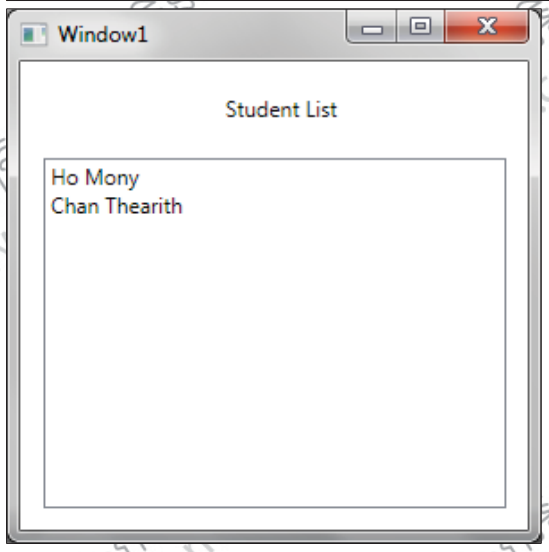
do statement គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការដំនើការនូវ Block នៃ Code ដដែលៗ នៅពេលដែលលក្ខខណ្ឌរបស់ True ។ មុននឹងឆែកមើលលក្ខខណ្ឌ វាដំនើការទៅលើ Block នៃ Code ម្តងជាមុនសិន ប្រសិនបើ True ធ្វើឡើងវិញប៉ុន្តែបើ False គឺនឹងចាកចេញពី Loop ។

Syntax:

```
do{
    statement;
}while(booleanExpression);
```

Ex:






```
using Microsoft.VisualBasic;

private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            string x;
            do
            {
             x = Interaction.InputBox("Enter Student's
                 Name:", "Student List", "", 100, 100);
            listBox1.Items.Add(x);

            }
            while (x != "");
        }
```

5. <u>for statement</u>:

for គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការដំនើការនូវ Block នៃ Code ដដែលៗ ទៅតាមចំនួនដែលបានកំនត់យ៉ាងត្រឹមត្រូវ នៅពេលដែលលក្ខខណ្ឌរបស់ True ។ ចំពោះ while និង do while គឺធ្វើគិតទៅលើ លក្ខខណ្ឌ ដោយមិនគិតពីចំនួនដងឡើយ ប៉ុន្តែ for វិញគឺធ្វើឧគិតទៅលើចំនួនដង ពិតប្រាកដ។ ជាទូទៅចំពោះការ loop គេនិយមប្រើប្រាស់ for ពីព្រោះយើងអាចដឹងពីចំនួនដែលវាត្រូវធ្វើការនៅក្នុង loop ។

Syntax: ❶ ❷ ❹
for(initialization; Boolean expression; update control variable)
{ ❸
statement;
}

1. Initialization: គឺជាតំលៃចាប់ផ្តើមដំនើការ loop ហើយវាធ្វើការតែម្តងប៉ុណ្ណោះ។
2. Boolean expression: គឺជាលក្ខខណ្ឌដែលត្រូវត្រួតពិនិត្យ ប្រសិនបើ True loop នឹងបន្តដំនើការ ប៉ុន្តែប្រសិនបើ False វិញ នោះ Loop នឹងបញ្ឈប់ដំនើការ
3. statement: គឺជា Block នៃ code ដែលត្រូវដំនើការក្នុង braces { } នៅពេលដែល លក្ខខណ្ឌ True
4. update control variable: គឺជាការតំឡើងឬបន្តុយ value របស់ variable នៅក្នុង initialization ឧកើនឡើង ឬថយចុះ ហើយបន្ទាប់មកវានឹងត្រលប់ទៅដំនើការនៅក្នុងតំមន់ Condition វិញ។

Ex:

```
for (int i = 0; i < 10; i++)
          {
               Console.Write(i + ",");
           }
           Console.WriteLine("Fire!");
```

Output:

```
0,1,2,3,4,5,6,7,8,9,Fire!
Press any key to continue . . .
```

យើងអាចធ្វើការកំនត់តំលៃរបស់ Initialization និង increase នៅត្រង់ទីតាំងផ្សេងៗទៀតបាន ប៉ុន្តែត្រូវដឹងពីចំនុច start របស់ loop និងពេលដែលវាត្រូវ False ចាកចេញពី Loop ។ គ្រប់ Loop ទាំងអស់ដូចជា while, do while, និង for គឺទាមទារឧមាននូវចំនុច False មួយដើម្បីឧវាបញ្ឈប់ និងចាកចេញពី Loop។

Ex:

```
int i = 0;
for (; i < 10; )
{
      Console.WriteLine(i);
      i++;
}
```

យើងអាចប្រើប្រាស់ Comma (,) ដើម្បីធ្វើការបំបែកនៅត្រង់ Initialization និង update control variable ដើម្បីកំនត់ច Variable 2 ឬច្រើនអាចធ្វើការ Loop នៅក្នុង for តែមួយ។

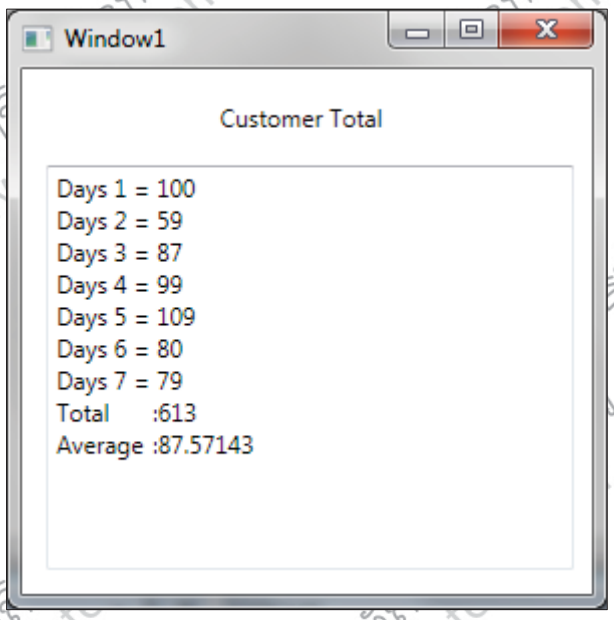
Ex:

```
#include<stdio.h>
      int n, i;
     for ( n = 0, i = 10; n != i; n++, i--)
     {
         Console.WriteLine(n +" Vs. "+i);
     }
Console.WriteLine(n + " = " + i);
```

Output:

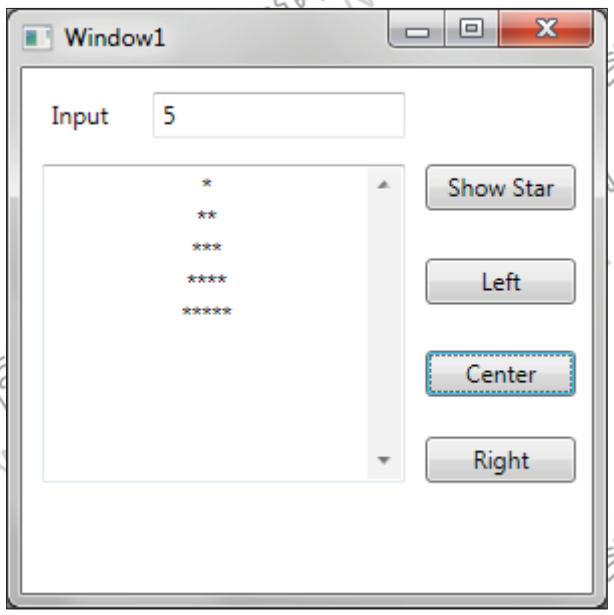
```
0 Vs. 10
1 Vs. 9
2 Vs. 8
3 Vs. 7
4 Vs. 6
5 = 5
```

6. <u>for statement Practical</u>:



```
float total=0;
for (int i = 0; i < 7; i++)
{
      string input = Interaction.InputBox("Enter Temperature in 7
      Days","Day: "+(i + 1).ToString(),"", 250, 250);
            total =total +float.Parse(input);
            textBox1.Text = textBox1.Text + "Days "+ (i+1).ToString()+
      " = "+input + "\n";
   }
textBox1.Text = textBox1.Text + "Total\t:" + total.ToString();
textBox1.Text = textBox1.Text + "\n";
  textBox1.Text =
      textBox1.Text + "Average\t:" + (total / 7).ToString();
```

7. ល្បំហាត់:



```
private void button1_Click(object sender, RoutedEventArgs e)
        {
            int loop;
            loop = int.Parse(textBox1.Text);
            String str="";
            for (int i = 1; i <= loop; i++)
            {
                for (int j = 1; j <= i; j++)
                {
                    str = str + "*";
                }
                str = str + "\n";
            }
            textBox2.Text = str;

        }
private void button2_Click(object sender, RoutedEventArgs e)
        {
            textBox2.TextAlignment = TextAlignment.Left;
        }

private void button3_Click(object sender, RoutedEventArgs e)
        {
            textBox2.TextAlignment = TextAlignment.Center;
        }

private void button4_Click(object sender, RoutedEventArgs e)
        {
            textBox2.TextAlignment = TextAlignment.Right;
        }
```

មេរៀនទី 6: សិក្សាពី Errors និង Exceptions

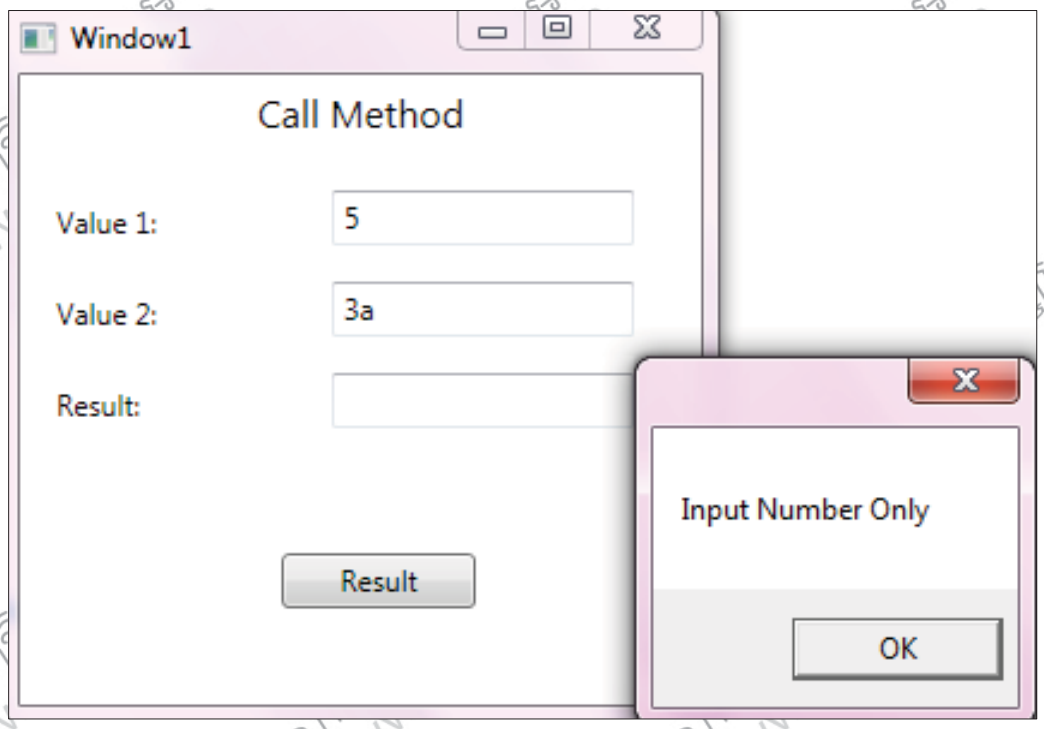
1. <u>Trying Code and Catching Exceptions</u>:

វាគឺការសរសេរកូដត្រួតពិនិត្យមើល Error ដែលអាចកើតមានឡើងនៅក្នុង Code ។ សារៈប្រយោជន៍របស់វាគឺការពារមិន Code បង្ហាញភាព Error ដោយខ្លួនឯងទេ ប៉ុន្តែកំនត់លក្ខខណ្ឌ និងមូលហេតុដែលត្រូវ Error ដើម្បីបង្ហាញព័ត៌មានឲយើងដឹង។

ដើម្បីងាយស្រួលក្នុងការគ្រប់គ្រងទៅលើ Error គឺត្រូវប្រើប្រាស់ជាមួយនឹង Exception និង Exception Handlers ដោយត្រូវបែងចែក Code ជាពីរតំបន់ដូចជា:

1. កូដសំរាប់ធ្វើការ គឺត្រូវសរសេរនៅក្នុងតំបន់របស់ try ប៉ុន្តែប្រសិនបើមានករណីដែលកូដនៅក្នុងតំបន់មាន Error កើតឡើងនោះ កូដនៅក្នុង try នឹងត្រូវរំលងចោល ដោយវាទៅដំនើការនៅក្នុងតំបន់ Catch វិញម្តង។
2. catch Handlers គឺជាតំបន់សំរាប់ដំនើការកូដនៅពេលដែលកូដនៅក្នុងតំបន់ try មាន Error កើតឡើង។ catch អាចត្រូវបានប្រើប្រាស់ចាប់ពីមួយទៅច្រើន ដើម្បីត្រួតពិនិត្យមើលពីមូលហេតុនៃ Error ឲបានច្រើនករណី។

Ex:

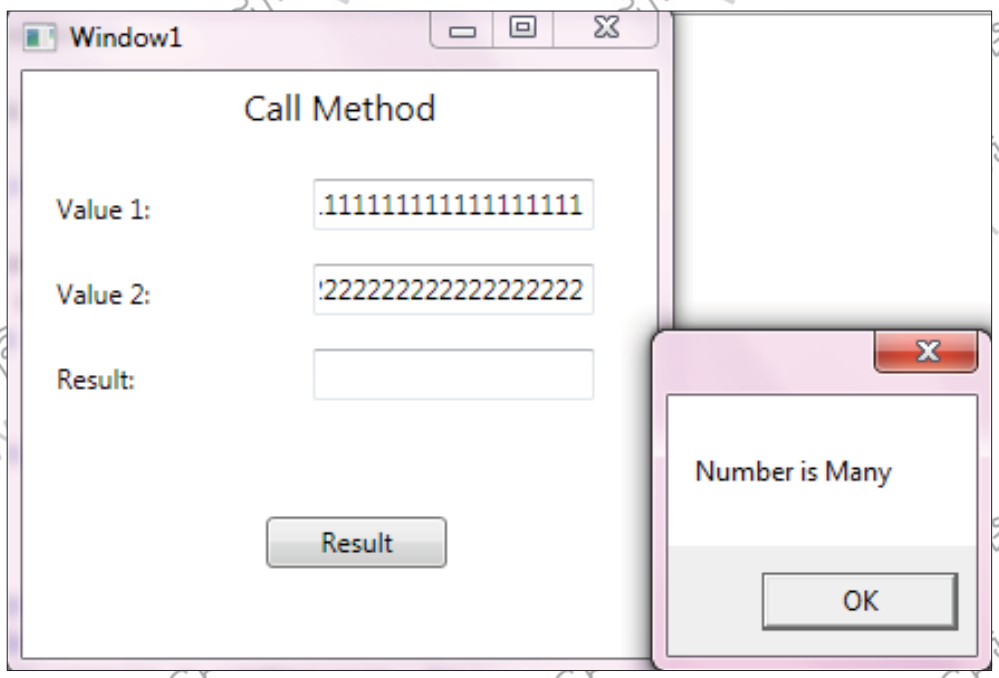


```
private int addValue(int x, int y)
{
            int sum;
            sum = x + y;
            return sum;
}

private void button1_Click(object sender, RoutedEventArgs e)
{
            try
            {
                int a = int.Parse(textBox1.Text);
                int b = int.Parse(textBox2.Text);
                int result = addValue(a, b);
                textBox3.Text = result.ToString();
            }
            catch (FormatException fEx)
            {
                MessageBox.Show("Input Number Only");
            }

}
```

2. <u>ការប្រើប្រាស់ Multiple Catch Handlers:</u>

វាគឺការសរសេរកូដត្រួតពិនិត្យមើល Error ដែលអាចកើតមានឡើងនៅក្នុង Code លើសពីមួយករណី។ តាមឧទាហរណ៍ខាងក្រោមមាន Catch Handlers ចំនួនពីរសំរាប់ប្រើប្រាស់ដើម្បីត្រួតពិនិត្យមើលពី Error ដែលអាចកើត មានឡើងនៅក្នុង Code ។

Ex:



```
private int addValue(int x, int y)
{
            int sum;
            sum = x + y;
            return sum;
}

private void button1_Click(object sender, RoutedEventArgs e)
{
            try
            {
                int a = int.Parse(textBox1.Text);
                int b = int.Parse(textBox2.Text);
                int result = addValue(a, b);
                textBox3.Text = result.ToString();
            }
            catch (FormatException fEx)
            {
                MessageBox.Show("Input Number Only");
            }
             catch (OverFlowException oEx)
            {
                MessageBox.Show("Number is Many");
            }

}
```

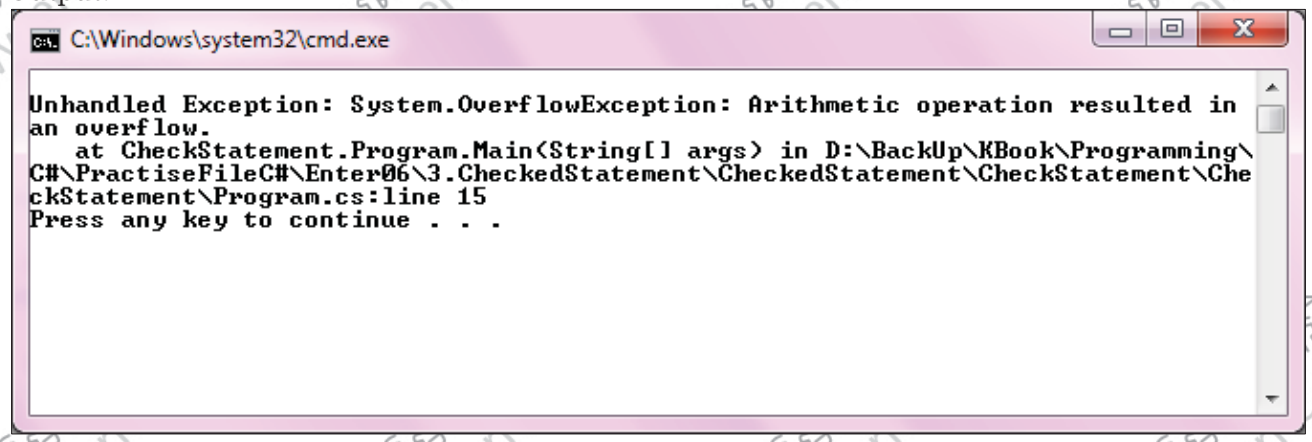
3. ការប្រើប្រាស់ Checked Statement:

Checked Statement គឺជា Block មួយសំរាប់ប្រើប្រាស់ដើម្បីធ្វើការត្រួតពិនិត្យ Error (OverflowException) ទៅលើតំលៃ Integer នៅពេលដែល Variable រក្សាទិន្នន័យលើសពី Data Type ដែលវាមាន។ មានតែ Variable ប្រភេទជា Integer ប៉ុណ្ណោះដែលត្រូវបានប្រើប្រាស់នៅក្នុង Checked Block ។

Ex:

```
int number = int.MaxValue;
checked
{
      int willThrow = number++;
      Console.WriteLine("this won't be reached");
}
```

Output:



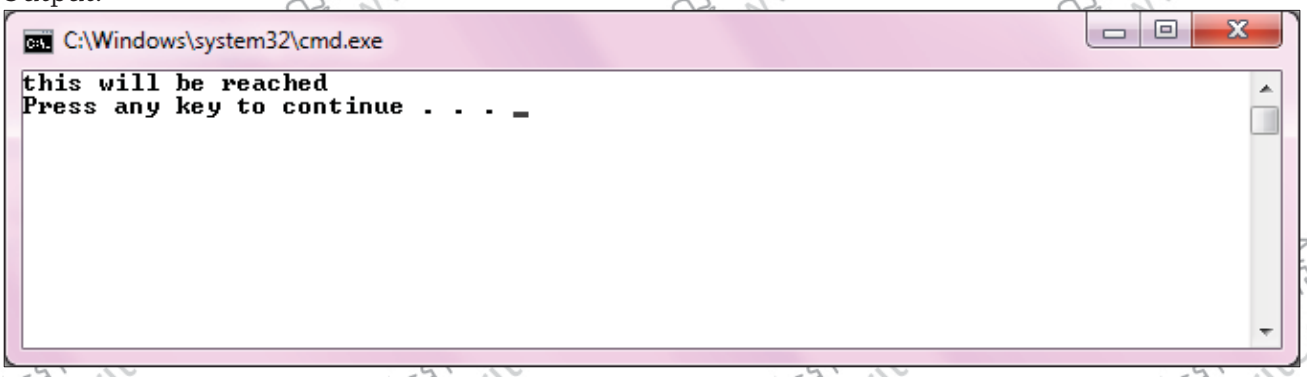
4. <u>ការប្រើប្រាស់ Unchecked Statement</u>:

Checked Statement គឺជា Block មួយសំរាប់ប្រើប្រាស់ដើម្បីធ្វើការត្រួតពិនិត្យ Error (OverflowException) ទៅលើតំលៃ Integer នៅពេលដែល Variable រក្សាទិន្នន័យលើសពី Data Type ដែលវាមាន។ មានតែ Variable ប្រភេទជា Integer ប៉ុណ្ណោះដែលត្រូវបានប្រើប្រាស់នៅក្នុង Checked Block ។

Ex:

```
int number = int.MaxValue;
unchecked
{
int wontThrow = number++;
Console.WriteLine("this will be reached");
}
```

Output:



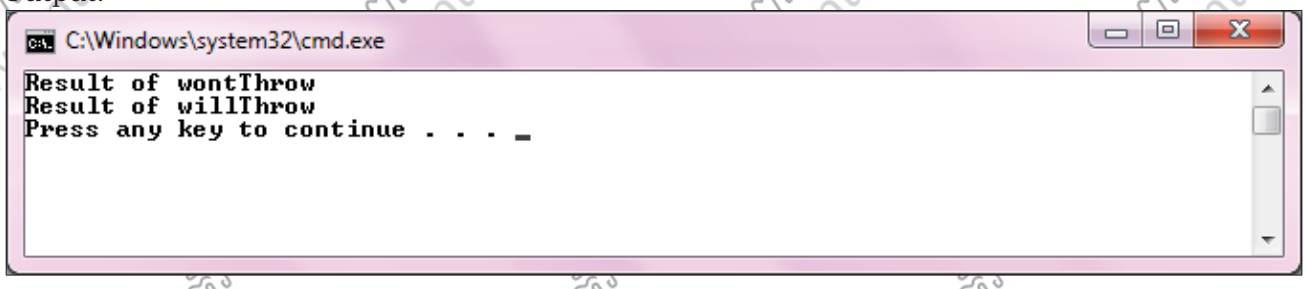
5. <u>ការប្រើប្រាស់ Checked Expression</u>:

យើងក៏អាចធ្វើការប្រើប្រាស់ checked និង unchecked keyword ដើម្បីធ្វើការគ្រប់គ្រងទៅលើ overflow ជាមួយនឹង integer expression ដោយដាក់វានៅខាងមុខ expression ទាំងនោះជាមួយ check ឬ unchecked keyword ។

Ex:

```
int wontThrow = unchecked(int.MaxValue + 1);
Console.WriteLine("Result of wontThrow");
int willThrow = checked(int.MaxValue + 1);
Console.WriteLine("Result of willThrow");
```

Output:



សូមចងចាំថាយើងមិនអាចធ្វើការប្រើប្រាស់ checked និង unchecked keyword ដើម្បីគ្រប់គ្រងទៅលើ floating-point arithmetic បានឡើយ។ checked និង unchecked keyword អាចប្រើប្រាស់ចំពោះតែ integer arithmetic ដូចជា int និង long data type ប៉ុណ្ណោះ។

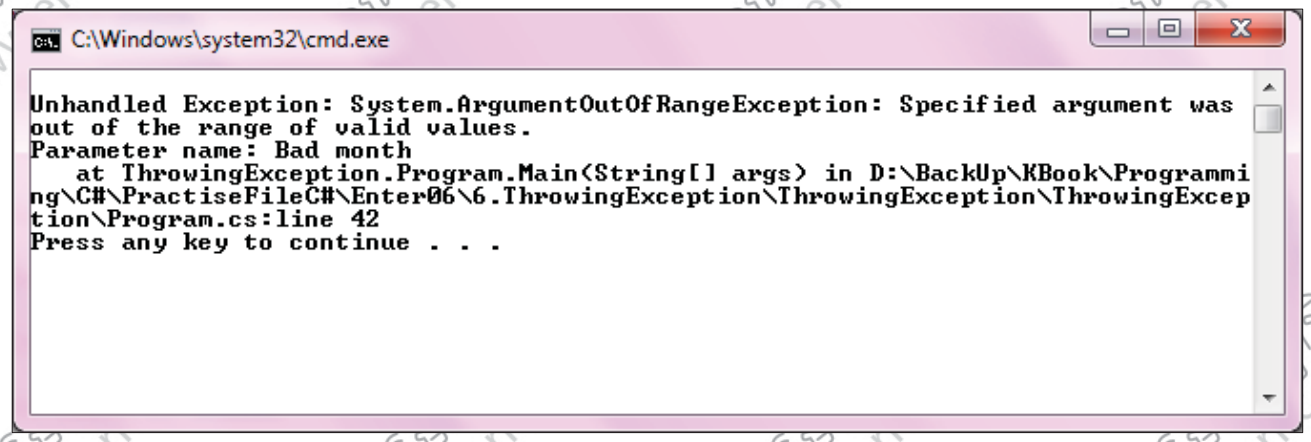
6. <u>ការប្រើប្រាស់ Throwing Exception</u>:

Throw Exception គឺត្រូវបានប្រើប្រាស់សំរាប់ត្រួតពិនិត្យទៅលើ Error ដែលកើតឡើងនៅពេលលើសពីទំហំដែលបានដាក់លក្ខខណ្ឌ។

Ex:

```
        static void Main(string[] args)
        {
            int n = 0;
            string result = "";
            {
                switch (n)
                {
                    case 1:
                        Console.WriteLine("January"); break;
                    case 2:
                        Console.WriteLine("February"); break;
                    case 3:
                        Console.WriteLine("March"); break;
                    case 4:
                        Console.WriteLine("April"); break;
                    case 5:
                        Console.WriteLine("May"); break;
                    case 6:
                        Console.WriteLine("June"); break;
                    case 7:
                        Console.WriteLine("July"); break;
                    case 8:
                        Console.WriteLine("August"); break;
                    case 9:
                        Console.WriteLine("September"); break;
                    case 10:
                        Console.WriteLine("October"); break;
                    case 11:
                        Console.WriteLine("November"); break;
                    case 12:
                        Console.WriteLine("December"); break;
                    default:
                          throw new ArgumentOutOfRangeException("Bad
                      month");
                }
            }
        }
```

Output:



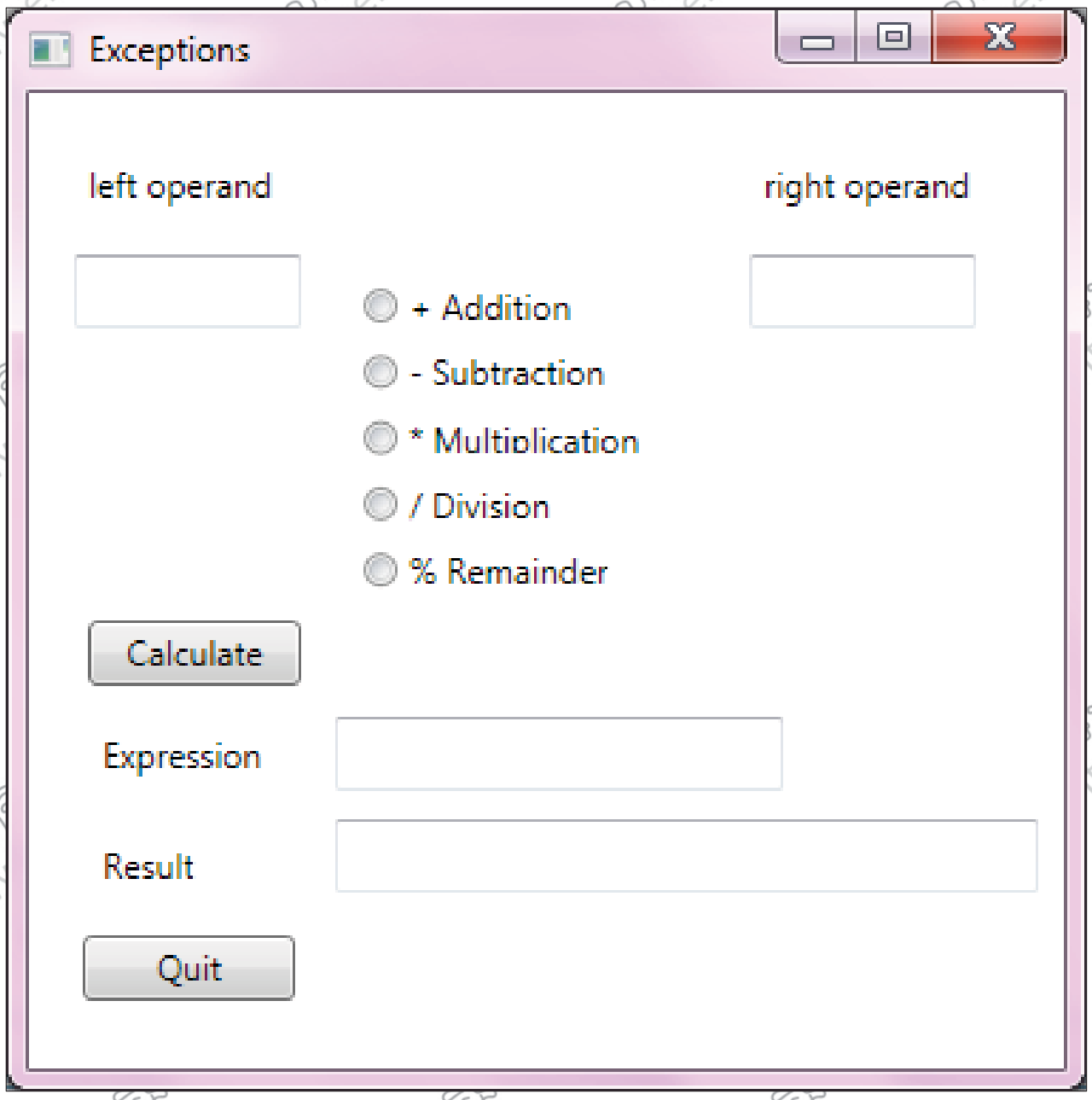
7. <u>ការប្រើប្រាស់ Finally Block</u>:

Finally Block គឺត្រូវបានដំនើការក្នុងទាំងឡាយណាដែលស្ថិតនៅក្នុងតំបន់វា បន្ទាប់ពី Try Block ឬ Catch Handler ចុងក្រោយណាមួយបានដំនើការរួចរាល់។

Ex:

```
static void Main(string[] args)
        {
            TextReader reader = null;
        try
            {
            reader = src.OpenText();
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                source.Text += line + "\n";
            }
            }
        finally
            {
                if (reader != null)
                {
                    reader.Close();
                }
            }
            }
```

8. <u>លំហាត់</u>:


Exceptions
left operand
right operand
+ Addition
- Subtraction
* Multiplication
/ Division
% Remainder
Calculate
Expression
Result
Quit