# CGT 215 Lecture 6 Methods: A Deeper Look

# Divide and Conquer

- □ Experience has shown that the best way to develop and maintain a large application is to construct it from small, simple pieces.
- □ This technique is called *divide and conquer*

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Methods

- □ Called *functions* or *procedures* in other programming languages
- Methods allow you to modularize an application by separating its tasks into selfcontained units.
- □ You have created many methods already.
- □ These methods are sometimes referred to as *user-defined methods*.

10/9/200

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Motivations for Modularizing an Application

- ☐ One is "divide and conquer"
  - Makes applications more manageable by constructing it as small, simple pieces
- □ Another is *software reusability* 
  - Existing methods can be used as building blocks to create new applications.
- □ Another reason is to avoid repeating code
  - Dividing an application into meaningful methods makes the application easier to debug and maintain.

10/9/200

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Software Engineering Observation 7.1

 Don't try to "reinvent the wheel." when possible, reuse Framework Class Library classes and methods.

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Software Engineering Observation 7.2

□ To promote software reusability, every method should be limited to performing a single, well-defined task, and the name of the method should express that task effectively. Such methods make applications easier to write, debug, maintain, and modify.

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Software Engineering Observation 7.3

☐ If you cannot choose a concise name that expresses a method's task, your method might be attempting to perform too many diverse tasks. It is usually best to break such a method into several smaller methods.

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Error-Prevention Tip 7.1

□ A small method that performs one task is easier to test and debug than a larger method that performs many tasks.

10/9/2009

CGT 215 Convright © 2009 Ronald J. Glotzhac

### static methods

- ☐ Although most methods execute on specific objects in response to method calls, this is not always the case.
- □ Sometimes a method performs a task that does not depend on the contents of any object.
- Such a method applies to the class in which it is declared as a whole and is known as a *static method*.

10/9/200

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# static methods

- □ Math.sqrt()
  - sqrt() is a static method of the Math class
  - You do not need to create an instance of the Math class in order to call sqrt()
- □ Console.WriteLine()
  - WriteLine() is a static method of the Console class
  - You do not need to create an instance of the Console class in order to call WriteLine()

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# const variables

- □ A constant is declared with the keyword const
   its value cannot be changed after the constant is declared.
  - Pi is a constant its value never changes
- ☐ Most const variables are by default static, unless declared inside of a method.

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# static variables

- ☐ Each instance of an object of a class has *separate* instance of the variables.
- ☐ This is *not* the case with *static variables*
- □ When objects of a class containing static variables are created, all the objects of that class *share one copy* of the class's *static variables*.

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Why is method Main declared static?

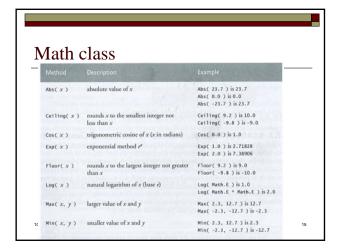
- ☐ Method Main is sometimes called the application's *entry point*.
- □ Declaring Main as static allows the execution environment to invoke Main without creating an instance of the class.
  - public static void Main( string args[] )

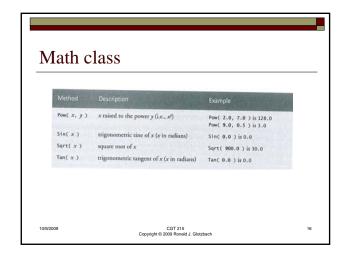
0/9/2009 CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Where do you declare static?

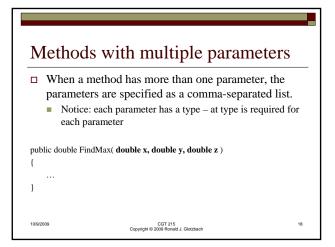
- □ Before the return type of a method:
  - public static void AddListItemMethod()
  - public static int GetValue()
- ☐ Before the class name of a variable declaration:
  - private static string name;
  - private static bool isTrue;

10/9/2009 CGT 215 Copyright © 2009 Ronald J. Glotzbach





# Declaring a method Vou need: Access modifier (optional) static Return type Name (optional) Parameters public static void Main( string args[]) public void SetGender( string gender) public string GetName()



# Calling methods

- ☐ You must provide the correct number of parameters in each method call
- □ If a method is declared with 3 parameters, you must pass 3 parameters in.
- □ Declaration:
  - $\qquad \qquad \text{public double FindMax}(\ \textbf{double}\ \textbf{x}, \textbf{double}\ \textbf{y}, \textbf{double}\ \textbf{z}\ )$
- □ Calling it:
  - FindMax( 3.2, 4.5, 1.3 );
  - FindMax(); //would be an error
  - FindMax( 2.3, 1.2 ); //would be an error

■ FindMax( 2.2, 3.3, 4.4, 5.5 ); //would be an error

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# String Concatenation (revisited)

- ☐ Use the + symbol to concatenate strings together.
- □ "hello" + "there" creates the string "hello there"
- □ "Maximum is: " + result
  - Another example of concatenation

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# **Common Errors**

- ☐ It is a syntax error to break a string literal across multiple lines in an application:
  - Wrong:
    - □ "hello how are you"

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# **Common Errors**

- □ Confusing the + operator used for string concatenation with the + operator used for addition can lead to strange results.
  - Example:

 $\Box$  "y + 2 = " + y + 2

• Results in the string "y + 2 = 52", **not** "y + 2 = 7"

y + 2 = y + (y + 2)

Results in the string "y + 2 = 7"

10/9/200

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# **Common Errors**

- □ Declaring a method outside the body of a class declaration or inside the body of another method is a syntax error.
- □ Omitting the return type in a method declaration is a syntax error.

10/9/2009

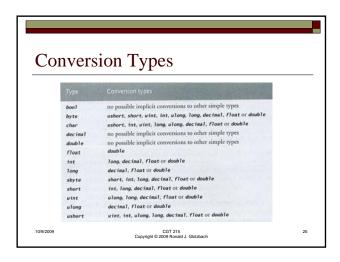
CGT 215 Copyright © 2009 Ronald J. Glotzbach

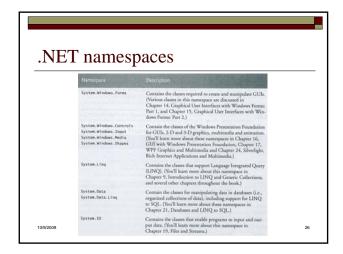
# **Common Errors**

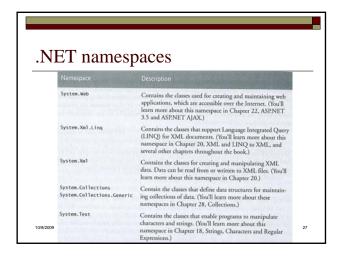
- □ Re-declaring a method parameter as a local variable in the method's body is a compilation error.
- □ Forgetting to return a value from a method that should return one is a compilation error.

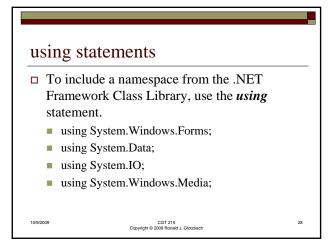
10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach









# Scope of declarations

- ☐ The scope of a parameter declaration is: the body of the method in which the declaration appears.
- ☐ The scope of a local-variable declaration is: from the point at which the declaration appears to the end of the block containing the declaration.

10/9/2009 CGT 215 29 Copyright © 2009 Ronald J. Glotzbach

# Scope of declarations

☐ The scope of a local-variable declaration that appears in the initialization section of a *for* statement's header is: the body of the for statement and the other expressions in the header.

10/9/2009 CGT 215 30 Copyright © 2009 Ronald J. Glotzbach

# Scope of declarations

- ☐ The scope of a method, property, or field of a class is the entire body of the class.
- ☐ This enables non-static methods and properties of a class to use any of the class's fields, methods, and properties, regardless of the order in which they are declared.
- ☐ Similarly, static methods and properties can use any of the static members of the class.

10/9/2009

CGT 215 novight © 2009 Ronald J. Glotzback

# Method Overloading

- ☐ Methods of the same name can be declared in the same class, as long as they have different sets of parameters (determined by the number, types, and order of the parameters)
- □ This is called *method overloading*.
- □ When an *overloaded method* is called, the C# compiler selects the appropriate method by examining the number, types and order of the arguments in the call.

10/9/2009

CGT 215
Convertebl © 2009 Populd I Glotzbach

# Overloaded methods

□ Method *calls* cannot be distinquished by return type. Therefore, the return type is not included as one of the attributes that define method overloading.

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Examples

- □ The order of parameter types is important.
  - public void Method1( int a, float b, string c)
  - public void Method1( float a, int b, string c)
  - public void Method1( string a, float b, int c)
- $\hfill\Box$  Example of calling each of the above methods:
  - Method1(3, 4.5, "hi there");
  - Method1(4.5, 3, "hi there");
  - Method1( "hi there", 4.5, 3);

10/9/2009

CGT 215 Copyright © 2009 Ronald J. Glotzbach

# Examples

- □ The number of parameters is important
  - public void MyMethod( string a )
  - public void MyMethod( string a, string b )
  - public void MyMethod( string a, string b, string c )
  - public void MyMethod( string a, string b, string c, string d )
- □ Example of calling each of the above methods:
  - MyMethod( "here", "is", "some", "text" );
  - MyMethod("here", "is", "some");
  - MyMethod( "here", "is" );MyMethod( "here" );

10/9/200

CGT 215 Copyright © 2009 Ronald J. Glotzbach

6