Chapter 5a
Methods

Chapter Objectives

- Identifying the parts of a method
- Using parameters
- Calling a method
- Relationship between parameters and arguments

General form of a Method

- Example of method header

```
private decimal commission(decimal salesAmount, Decimal)
```
A general method is reusable code that can be called from multiple event handlers
- Breaks down large sections of code into smaller units to perform a specific task
- Simplifies future program maintenance

All methods return a value
- Methods may return a value of a specific type
- Methods that do not return a specific value type return a void type.

Write a method header and enclose lines of code within a set of braces
Access can be private, public, internal, or protected
Code will not execute unless there is a call from another method

Specify a data type for the value to be returned in the method header
Supply arguments by placing a value inside the parentheses
Declare the parameter(s) that the method requires
  - Assign a data type and an identifier (name)
  - Identifier used inside the method refers to the value passed as an argument
  - When using multiple parameters, the sequence of the supplied arguments is critical
RETURNING THE RESULT OF A METHOD

A method can return one value
- Must be same data type as named in method header
- Return value is passed back with a return statement
- Keyword return is followed by variable or expression that contains the value to return
- If method header specifies void, do not use return keyword

PASSING ARGUMENTS TO METHODS

Calling or Invoking a Method
- Write the name assigned to the method followed by parentheses
  - All invoked methods have parentheses even if the return type of the method is void...in which case, use () only
- Methods defined with parameters must be invoked with arguments of the correct data type
  - The number of arguments, their sequence and data type must match the parameter list
- Arguments are usually local variables, but may be literals, constants, or properties

WRITING A COMMISSION METHOD

```csharp
private decimal commission(decimal salesAmountDecimal)
{
    // Calculate the sales commission.
    if (salesAmountDecimal < 1000M)
    {
        return 0M;
    }
    else if (salesAmountDecimal <= 2000M)
    {
        return 0.15M * salesAmountDecimal;
    }
    else
    {
        return 0.2M * salesAmountDecimal;
    }
}
```
Reference and Output Parameters (special parameters)

- Reference Parameters
  - Declared by using the `ref` modifier
  - Refers to the same location as the variable that was passed to the method
  - Calling and called methods have access to same memory location
  - OOP principles advise strongly against using reference parameters

Reference and Output Parameters (special parameters – cont’d)

- Output Parameters
  - Uses the `out` modifier in the method header
  - Allows a method to return multiple values
  - Both the method header and the call to the method must contain the `out` keyword
  - The method must explicitly assign a value to each argument declared with the `out` keyword
  - An address is sent to the method rather than the value

Breaking Calculations into Smaller Units

- Projects with many calculations are easier to understand and write if calculations are broken into small units
- Each unit should perform one program function or logic block