C# Programming Homework 03

Chapter 03, C# Step by Step

Readings

Read chapter 3 in the *C# Step by Step* book.

Discussion Questions

Answer the discussion questions in writing for chapter 3.

1. What is a *method*? A method is a named sequence of statements. A named block of code that optionally accepts input and returns output. Can be either functions or sub-procedures, or both.
2. (Not in book) What i s the difference between a *f unction* and a *procedure/subprocedure/subroutine*? Functions return output. Procedures/subprocedures/subroutines execute a series of actions. Function: a named block of code that returns a value and doesn't take an action. A sub-procedure takes an action (eg. 'PRINT').
3. What does a *return* statement do? Returns the value of an method/expression and returns a value and completes the method/procedure. Completes the method or procedure and makes allocated memory available again.
4. What is an *expression bodied* method? Very simple methods that perform simple tasks without involving any additional logic.
5. What is the *scope* of a variable? The region of the program in which that variable is usable. Scope applies to methods as well as variables.
6. What i s a *field*? The proper C# term for a variable (defined by a class).
7. What is an *overloaded* method? A method that has the same name as another method, but more parameters or different types of parameters. Multiple instances of a method with the same name that can be differentiated by their parameters.
8. How do you call a method that requires *arguments*? By giving it a name and an access modifier and pass parameters to it inside the parentheses.
9. **How do you write a method, that i s, specify the method definition, that requires a *parameter list*? method(); give it a ~~return~~ type, Name it, create the parameter list, give the body of the method**
10. How do you specify a parameter as optional when defining a method? By using the OP method
11. How do you pass a argument to a method as a *named parameter*? you specify the name of the parameter, followed by a colon and the value to use. Positional means that the order of the arguments is correlated to the order of the parameters - (FirstParameter gets arg1, second parameter gets arg 2, third parameter gets arg3, etc.). If you don't want positional arg's you can use named arg's.
12. How do you return values from a method? Can you return multiple values from a method, and if so, how? ...by placing a return statement from within the method body.
13. What is a tuple? Two or more values within parentheses separated by a comma.
    How do you define a method that returns multiple values? Give an example of a method that returns multiple values other than the example in the book.
14. Examine the method definition on page 83 of the book. Desk check the execution of this method.
    What do you discover? The number decrements by one each cycle until it reaches 1. *This is called recursion*.
15. How does the compiler resolve an ambiguity between named arguments and optional parameters?
16. The book states: \A key feature of C# and other languages designed for the .NET Framework is the ability to interoperate with applications and components written with other technologies." What is the COM and how is the CLR dependent on the COM?