Chapter 1
How to get started with Visual Studio

Objectives

Applied

1. Use Visual Studio 2015 to do any of these operations:
    - Open and close an existing C# project or solution
    - Display the Form Designer for each of the forms in a project
    - Display the Code Editor for each of the forms in a project
    - Use the Solution Explorer to review the files in a project
    - Open, hide, and adjust the windows for a project
    - Build and run a project

Knowledge

1. Describe the main difference between a Windows Forms application and a Web Forms application.
2. List two additional types of .NET applications that you can develop with Visual Studio 2015.
3. Name the two languages that you can use for rapid application development with Visual Studio 2015.
4. Describe the two main components of the .NET Framework.
5. In general terms, describe the C# compiler, Microsoft Intermediate Language, the assembly, and the Common Language Runtime.
6. Describe the use of each of these windows in the Visual Studio IDE: Form Designer, Code Editor, and Solution Explorer.
7. In general terms, explain what makes it possible to select a target framework with Visual Studio 2015.

Chapter 2

How to design a Windows Forms application

Objectives

Applied

1. Given the form design and property settings for a simple application, use the Form Designer to design the form.
2. When necessary, rename the form, project, and solution files for an application.
3. Customize the Visual Studio environment for use with C# by setting the Visual Studio options and the import and export settings.

Knowledge

1. Describe the use of the Toolbox and Properties window.
2. Describe the Name and Text properties for a form or control.
3. Describe these additional properties for a control: Enabled, ReadOnly, and TextAlign.
4. Describe the way you adjust the tab order for the controls on a form, set access keys for controls, and set the default buttons for the Enter and Esc keys for a form.

Chapter 3

How to code and test a Windows Forms application

Objectives

Applied

1. Given the code for a simple application, use the skills presented in this chapter to add the code and test the application.
2. Use indentation and blank lines to make the code easier to read.
3. Use comments to document the code for the entire form or for portions of the code.
4. Use any of the help features to get the information that you need for developing an application.

Knowledge

1. In the context of object-oriented programming, describe these terms: class, object, instantiation, instance, property, method, event, and member.
2. Describe how an application responds to events.
3. Describe the use of snippets, refactoring, and annotations.
4. Distinguish between a syntax (or build) error and a runtime error.
5. Distinguish between testing and debugging.
6. Explain how a data tip can help debug a runtime error.

Chapter 4

How to work with numeric and string data

Objectives

Applied

1. Given an arithmetic expression and the values for the variables in the expression, evaluate the expression.
2. Use numeric and string data as needed within your applications. That means you should be able to do any of the following:
   - declare and initialize variables and constants
   - code arithmetic expressions and assignment statements
   - use the static methods of the Math class
   - cast and convert data from one type to another
   - use the correct scope for your variables
   - declare and use enumerations
   - work with nullable types and the null-coalescing operator

Knowledge

1. Distinguish between a variable and a constant and give the naming conventions for each.
2. Describe any of these data types: int, double, decimal, bool, and string.
3. Describe any of these terms: literal value, null value, empty string, concatenate, append, escape sequence, string literal, verbatim string literal, and nullable data type.
4. Describe the order of precedence for arithmetic expressions.
5. Distinguish between implicit casting and explicit casting.
6. Distinguish between a value type and a reference type.
7. Describe the differences in the ways that casting, the ToString method of a data structure, the Parse and TryParse methods of a data structure, and the methods of the Convert class can be used to convert data from one form to another.
8. Describe the differences between class scope and method scope.

Chapter 5

How to code control structures

Objectives

Applied

1. Given a Boolean expression and the values for the variables in the expression, evaluate the expression.
2. Use if statements and switch statements as needed within your applications.
3. Use while, do-while, and for statements as needed within your applications.

Knowledge

1. Compare the if-else and switch statements.
2. Describe the differences between while, do-while, and for loops.
3. Explain how the break and continue statements work when used within a loop.
4. Describe the use of breakpoints, the Locals window, and stepping through code as they apply to debugging errors within loops.
5. Describe block scope in the context of the control structures.

Chapter 6

How to code methods and event handlers

Objectives

Applied

1. Given the specifications for a method, write the method.
2. Give the documentation for a method including its signature, use the method.
3. Give a portion of code that can be converted to a method call and a method, use refactoring to do the conversion.

Knowledge

1. Describe the signature of a method.
2. Explain the difference between passing an argument by value and by reference.
3. Describe the use of optional arguments.
4. Describe the advantages of passing arguments by name.
5. In general terms, describe the way event wiring works in C#, what you need to do to delete an event, and how you can use one event handler to handle two or more events.

Chapter 7

How to handle exceptions and validate data

Objectives

Applied

1. Given a form that uses text boxes to accept data from the user, write code that catches any exceptions that might occur.
2. Given a form that uses text boxes to accept data and the validation specifications for that data, write code that validates the user entries.
3. Use dialog boxes as needed within your applications.

Knowledge

1. Explain what an exception is and what it means for an exception to be thrown and handled.
2. Describe the Exception hierarchy and name two of its subclasses.
3. Describe the use of try-catch statements to catch specific exceptions as well as all exceptions.
4. Describe the use of the properties and methods of an exception object.
5. Describe the use of throw statements.
6. Describe the three types of data validation that you're most likely to perform on a user entry.
7. Describe two ways that you can use generic validation methods in a method that validates all of the user entries for a form.

Chapter 8
How to work with arrays and collections

Objectives

Applied

1. Given the specifications for an application that requires the use of a one-dimensional, rectangular, or jagged array, write the code that works with the array.
2. Given the specifications for an application that requires the use of one of the collection classes presented in this chapter, write the code that works with the collection.

Knowledge

1. Distinguish between a for loop and a foreach loop.
2. Explain how the Array class can be used with an array.
3. Describe how the null-conditional operator works and when you would use it.
4. Distinguish between an untyped and a typed collection class.
5. Describe the differences between these collection classes: list, sorted list, queue, stack, and array list.

Chapter 9

How to work with dates and strings

Objectives

Applied

1. Given the date-handling requirements of an application, write the code that satisfies the requirements.
2. Given the string-handling requirements of an application, write the code that satisfies the requirements.
3. Given the formatting requirements of an application, use the Format method of the String class or interpolated strings to provide the formatting.

Knowledge

1. Describe the way a DateTime variable is stored.
2. Explain how a String object differs from a StringBuilder object.

Chapter 10

More skills for working with Windows forms and controls

Objectives

Applied

1. Given the specifications for a form that uses any of the controls presented in this chapter, design and code the form.
2. Given a form with two or more controls, set the tab order of the controls using the Tab Order view of the form.
3. Given the specifications for an application that displays custom or standard dialog boxes, design and code the application.

Knowledge

1. In general terms, describe the use of these controls: combo box, list box, radio button, check box, and group box.
2. Explain how the refactoring feature helps you change some of the occurrences of the form name in the code when you rename the file for the form, but not the occurrences in the event handlers for the form.
3. Describe the way the Program class for an application displays the first form of an application.
4. Describe how you can use the DialogResult enumeration and the Tag property to pass data between a form and a custom dialog box.
5. Describe how you can use the FormClosing event to stop a form from closing.

Chapter 11

How to debug an application

Objectives

Applied

1. Use the debugging techniques presented in this chapter to debug any unhandled exceptions or logical errors in the applications that you develop.

Knowledge

1. Describe the differences between the three Step commands that you can use to control the execution of an application.
2. Describe the primary differences between the Autos window, the Locals window, and the Watch window.
3. Describe the use of the Immediate window.
4. Describe the call stack that’s displayed in the Call Stack window.
5. Explain how you can use the Console class to display information in the Output window.

Chapter 12

How to create and use classes

Objectives

Applied

1. Given the specifications for an application that uses classes with any of the members presented in this chapter, develop the application and its classes.
2. Use the Solution Explorer and the CodeLens feature to browse the classes in a solution.
3. Use class diagrams and the Class Details window to review, delete, and start the members of the classes in a solution.
4. Use the Peek Definition window to display and edit the code for a member in another class from within the class that refers to it.

Knowledge

1. List and describe the three layers of a three-layered application.
2. Describe these members of a class: constructor, method, field, and property.
3. Describe the concept of encapsulation.
4. Explain how instantiation works.
5. Describe the main advantage of using object initializers.
6. Describe the concept of overloading a method.
7. Explain what a static member is.
8. Explain how auto-implemented properties work.
9. Explain how expression-bodied properties and methods work.
10. Describe the basic procedure for using the live code analysis feature to generate code stubs.
11. Describe the difference between a class and a structure.

Chapter 13

How to work with indexers, delegates, events, and operators

Objectives

Applied

1. Develop and use classes that have indexers, delegates, events, and overloaded operators.

Knowledge

1. Describe the use of an indexer.
2. Explain why you should validate parameters and throw exceptions in your classes.
3. Describe the use of delegates and events without and with anonymous methods and lambda expressions.
4. Describe the use of operator overloading.

Chapter 14

How to work with inheritance

Objectives

Applied

1. Use any of the features of inheritance that are presented in this chapter as you develop the classes for your applications.

Knowledge

1. Explain how inheritance is used within the .NET Framework classes and how you can use it in your own classes.
2. Explain why you might want to override the ToString, Equals, and GetHashCode methods of the Object class as you develop the code for a new class.
3. Describe the use of the protected and virtual access modifiers for the members of a class.
4. Describe the use of polymorphism.
5. Describe the use of the Type class.
6. Describe the use of explicit casting when working with the objects of a base class and its derived classes.
7. Distinguish between an abstract class and a sealed class.

Chapter 15

How to work with interfaces and generics

Objectives

Applied

1. Use .NET interfaces as you develop the classes for an application.
2. Develop and use your own interfaces.

Knowledge

1. Distinguish between an interface and an abstract class.
2. Describe the use of the .NET ICloneable, IComparable<>, and IEnumerable<> interfaces.
3. In general terms, describe the use of generic interfaces.

Chapter 16

How to organize and document your classes

Objectives

Applied

1. Add XML documentation to your classes.
2. Create and use a class library.

Knowledge

1. Describe two ways that you can code two or more classes in a single file.
2. Describe the benefits of creating and using a class library.
3. Describe the use of namespaces.
4. Describe the use of partial classes.

Chapter 17

An introduction to database programming

Objectives

Knowledge

1. Describe the hardware components of a typical multi-user system.
2. Describe the software components of a typical multi-user database application.
3. Explain how a table in a relational database is organized.
4. Explain how the tables in a relational database are related.
5. Describe the use of these SQL statements: Select, Insert, Update, Delete.
6. Describe the use of these ADO.NET components: data adapter, command, connection, data reader, dataset, data table.
7. Compare the structure of a dataset with the structure of a relational database.
8. Describe concurrency, optimistic concurrency control, and "last in wins."
9. Describe two ways that you can create ADO.NET components.

Chapter 18

How to work with data sources and datasets

Objectives

Applied

1. Use a data source to get the data that an application requires.
2. Use a DataGridView control to present the data that's retrieved by a data source.
3. Use other controls like text boxes to present the data that's retrieved by a data source.
4. Write the code for handling any data errors that result from the use of the data source or the controls that are bound to it.
5. Use the Dataset Designer to (1) view the schema for the dataset of a data source, (2) modify a query using the Query Builder, (3) preview the data for a query, or (4) review the SQL statements that are generated for a data source.

Knowledge

1. Describe the use of a connection string in an app.config file.
2. Describe the use of the Fill method of the TableAdapter object and the UpdateAll method of the TableAdapterManager object.
3. Describe the use of the EndEdit method of the BindingSource object.
4. Describe the two categories of data errors that can occur when you run an application that uses a data source.
5. Describe the use of the DataError event for a DataGridView control.
6. In general terms, describe the way the SQL statements that are generated for a data source (1) prevent concurrency errors, and (2) refresh a dataset when the database generates the keys for new rows.

Chapter 19

How to work with bound controls and parameterized queries

Objectives

Applied

1. Format the data in a bound text box by setting the properties for the control.
2. Bind a combo box to a data source.
3. Use the properties and methods of the BindingSource class to navigate and modify the rows in a dataset.
4. Create and use a parameterized query with a data source.
5. Customize a ToolStrip control by adding controls to it, deleting controls from it, and formatting the controls that are on it. Then, code the event handlers for making these controls work.
6. Customize the appearance and operation of a DataGridView control.
7. Use a DataGridView control as part of a Master/Detail form.

Knowledge

1. Explain why you might want to use code to work directly with the binding source object for a data source.
2. Describe the use of a parameterized query and the ToolStrip control that gets generated for the query.

Chapter 20

How to use ADO.NET to write your own data access code

Objectives

Applied

1. Use a connection to access a SQL Server database.
2. Use a data reader to retrieve data from a database.
3. Use data commands to execute action queries or queries that return a scalar value.
4. Use parameters to limit the data that's processed by a data command.

Knowledge

1. Describe the use of parameters with SQL statements.
2. Describe the use of a data reader.
3. Describe the use of the two types of queries that don't return result sets.

Chapter 21

How to work with files and data streams

Objectives

Applied

1. Develop an application that requires the use of text or binary files.

Knowledge

1. Distinguish between a text file and a binary file.
2. Describe the use of FileStream, StreamReader, StreamWriter, BinaryReader, and BinaryWriter objects.
3. Describe two common types of I/O exceptions.

Chapter 22

How to work with XML files

Objectives

Applied

1. Use the XmlWriter class to write an XML document.
2. Use the XmlReader class to read an XML document.
3. Use Visual Studio's XML Editor to create and edit an XML document in a project.

Knowledge

1. Identify the following in an XML document: XML declarations, start tags, end tags, root elements, parent elements, child elements, attributes, element content, and comments.

Chapter 23
How to use LINQ

Objectives

Applied

1. Use any of the LINQ features presented in this chapter to query an in-memory data structure such as an array, sorted list, or generic list.

Knowledge

1. In general terms, explain how LINQ is implemented.
2. Name the interface that a data source must implement to use LINQ with that data source.
3. Describe the three stages of a query operation.
4. Describe how the type of a query variable is determined.
5. Describe how deferred execution works.
6. Describe when a query operation results in a projection.
7. Describe when a query operation results in an anonymous type.
8. Describe the purpose of each of the following LINQ clauses: from, where, orderby, select, join.
9. Explain how extension methods and lambda expressions are used with LINQ.

Chapter 24
How to use the Entity Framework

Objectives

Applied

1. Create an Entity Data Model that contains one or more entity classes.
2. Use the Entity Data Model Designer to work with an Entity Data Model.
3. Use LINQ to Entities to retrieve data from one or more database tables.
4. Use the Entity Framework to add, modify, and delete rows in a database table.

Knowledge

1. In general terms, explain how the Entity Framework works.
2. Describe these terms as they relate to an Entity Data Model: conceptual model, storage model, mappings, object context, entity class, navigation property, and association.
3. Explain how you can provide for concurrency when using the Entity Framework.

Chapter 25

How to enhance the user interface

Objectives

Applied

1. Use the Tab control to organize an application.
2. Use a multiple-document interface for an application.
3. Add menus, toolbars, and help information to an application.

Knowledge

1. Distinguish between a single-document and a multiple-document interface.

Chapter 26

How to deploy an application

Objectives

Applied

1. Use the ClickOnce feature to deploy an application.
2. Use a Setup program created using InstallShield to deploy an application.

Knowledge

1. In general terms, describe these techniques for deploying an application: XCopy, ClickOnce, and Setup program.