Lesson 1

Introduction to Programming

Learning Objectives

Students will learn about:

- Basics of computer programming
- Decision structures
- Repetition structures
- Exception handling

ODN Skills

Understand computer storage and data types.	1.1
Understand computer decision structures.	1.2
Identify the appropriate method for handling repetition.	1.3
Understand error handling.	1.4

Lesson Summary — Lecture Notes

You should begin this lesson by introducing yourself and the course.

A good place to start is to discuss what programming is. You can ask the students what computer applications they use frequently. Next, explain that computers respond to a predefined set of instructions in a predictable way. You should next explain that computers can only understand digital electronic signals. They cannot understand natural languages. However, multiple layers of abstractions—such as operating systems and language compilers—make it possible for humans to program computer by using a high-level programming language.

Next, introduce algorithms as a simple tool that lets you focus on creating a solution rather than worrying about the details of the programming language elements. The lesson discusses flowchart and decision tables as two tools for developing algorithms. After you have explained the basic notations, take a sample flowchart and have students follow along as you “dry-run” the algorithm. Make sure that students can identify whether they should use a flowchart or a decision table for a given problem. Encourage students to “dry-run” the program on their own with different set of values so that they take all the possible execution paths of the algorithm. The algorithms that you develop right now can be used as a basis of writing equivalent C# program in the next section.

The next section introduces the C# programming language. Make sure that the students have a working setup of Microsoft Visual C# 2010 Express or Visual Studio 2010. As you introduce the first program, make sure that students understand the basic structure of a C# program. Make sure that the students know that the Main method is a special method because it also serves as an entry point to a program.

Spend some time making sure that students understand that a variable's data type determines what value it can contain and what kind of operations may be performed on it. By end of this lesson, the students should also know about basic operators and creating simple expressions. It will be a good idea to demonstrate a few expressions and walk the students through how the value of an expression is evaluated.

The next part of the lesson covers the C# decision structures, which include the if-else and the switch statement. You may want to take a flowchart that you discussed earlier and try to covert that into an equivalent C# program using each of the decision structures. You may want to discuss different scenarios where a switch statement will be a better choice over an if-else statement and vice-versa. When discussing the switch statement make sure students understand the working of the break statement and the default case.

The next discussion covers the four control structures that allow programs to perform repetitive tasks: the while loop, the do-while loop, the for loop, and the foreach loop. Discuss the features of each control structure and give examples for situations where one structure will be a better choice over the others. Make sure that students understand the termination condition for a looping structure. Also discuss how students can use the exit and continue statements to effect the functioning of a repetition structure.

Next, discuss the concept of recursion. Give examples of a problem that can be partly defined in terms of itself. You can find examples in general life, mathematics, and in computer science. A simple example that you can discuss is the problem of moving 10 boxes when you know only how to move one box. You should encourage the students to think that this problem can be defined as moving one box followed by solving a smaller problem of moving nine boxes. Make sure that students are able to identify the base case as well as the recursive case of a recursive problem.

Finally, discuss error-handling in programs. Give examples of situations where a program can encounter an error and then talk about ways in which you can possibly handle an error case. You should encourage students to anticipate possible error cases and program in such a way that each error case is handled. Discuss the .NET Framework's supports for exception handling to raise and handle runtime errors. Make sure students understand the usage of both try-catch-finally as well as the try-finally construct.

Key Terms

algorithm—A set of ordered and finite steps to solve a given problem.

array—A collection of items in which each item can be accessed by using a unique index.

constant—A data field or local variable whose value cannot be modified.

decision table—A compact and readable format for presenting the algorithm that involves a large number of conditions.

do-while loop—Repeatedly executes a block of statements until a specified Boolean expression evaluates to false. The do-while loop tests the condition at the bottom of the loop.

exception—An error condition that occurs during the execution of a program..

flowchart—A graphical representation of an algorithm.

for loop—Combines the three elements of iteration—the initialization expression, the termination condition expression, and the counting expression—into more readable code.

foreach loop—Useful for iterating through the elements of a collection.

if—A statement that executes a given sequence of statements only if the corresponding Boolean expression evaluates to true.

if-else—A statement that allows your program to perform one action if the Boolean expression evaluates to true and a different action if the Boolean expression evaluates to false.

operator—The symbols (such as +, -, *, and /) that specify which operation to perform on one or more the operands before and returning a result.

program—A set of precise instructions to complete a task.

recursion—A programming technique that causes a method to call itself in order to compute a result.

switch—A statement that allows multi-way branching. In many cases, using a switch statement can simplify a complex combination of if-else statements.

try-catch-finally—To handle exceptions, the code that throws exceptions is placed inside a try block, and the code that handles the exceptions is placed inside a catch block, and the code that needs to be executed regardless of whether or not an exception is thrown is placed inside the finally block.

variable—A symbolic name associated with a value and whose associated value may be changed.

while loop—Repeatedly executes a block of statements until a specified Boolean expression evaluates to false.

Lesson 1

Introduction to Programming

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Review the following code snippet:

```
int n = 20;
int d = n++ + 5;
```

What will be the value of d after this code snippet is executed?

a. 25

b. 26

c. 27

d. 28

2. Review the following code snippet:

```
private static void WhileTest()
{
    int i = 1;
    while (i < 5)
    {
        Console.WriteLine("The value of i = {0}",
   i);
        i++;
    }
}
```

How many times will the while loop be executed in this code snippet?

a. 0

b. 1

c. 4

d. 5

3. Review the following code snippet:

```
int number1 = 10;
int number2 = 20;
if (number2 > number1)
Console.WriteLine("number1");
```

```
Console.WriteLine("number2");
```

What output will be displayed after this code snippet is executed?

a.

```
number1
```

b.

```
number2
```

c.

`number1`

`number2`

d.

```
number2
number1
```

4. In a switch statement, if none of the case statements match the switch expression, then control is transferred to which statement?

   a. break

   b. continue

   c. default

   d. return

5. You need to write code that closes a connection to a database and you need to make sure this code is always executed regardless of whether an exception is thrown. Where should you write this code?

   a. Within a try block

   b. Within a catch block

   c. Within a finally block

   d. Within the Main method

6. You need to store values ranging from 0 to 255. You also need to make sure that your program minimizes memory use. Which data type should you use to store these values?

   a. byte

   b. char

   c. short

   d. int

7. If you don't have a base case in your recursive algorithm, you create an infinite recursion. An infinite recursion will cause your program to throw an exception. Which exception will your program throw in such a case?

   a. OutOfMemoryException

   b. StackOverflowException

   c. DivideByZeroException

   d. InvalidOperationException

8. You are learning how to develop repetitive algorithms in C#. You write the following method:

```
private static void ForTest()
```

```
{
    for(int i = 1; i < 5;)
    {
        Console.WriteLine("The value of i = {0}",
i);
    }
}
```

How many repetitions will the for loop in this code perform?

a. 0

b. 4

c. 5

d. Infinite repetitions

9. Which of the following C# features should you use to organize code and create globally unique types?

   a. Assembly

   b. Namespace

   c. Class

   d. Data type

10. You write the following code snippet:

```
int[] numbers = {1, 2, 3, 4};
int val = numbers[1];
```

You also create a variable of the RectangleHandler type like this:

```
RectangleHandler handler;
```

What is the value of the variable val after this code snippet is executed?

a. 1

b. 2

c. 3

d. 4

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. The **switch** statement selects for execution a statement list having an associated label that corresponds to the value of an expression.
2. The **do-while** loop tests the condition at the bottom of the loop instead of at the top.
3. The only operator that takes three arguments is the **?:** operator..
4. The **foreach** loop is the most compact way to iterate through the items in a collection.
5. On a 32-bit computer, a variable of int data type takes **two** bytes of memory.

6. To access the first element of an array, you use an index of **0**.
7. **Recursion** is a programming technique that causes a method to call itself in order to compute a result.
8. **Constants** are data fields or local variables whose value cannot be modified.
9. When an algorithm involves a large number of conditions, a(n) **decision table** is a compact and readable format for presenting the algorithm.
10. A(n) **flowchart** is a graphical representation of an algorithm.

Competency Assessment

Project 1-1: Converting a Decision Table into a C# Program

You are developing an invoicing application that calculates discount percentages based on the quantity of a product purchased. The logic for calculating discounts is listed in the following decision table. If you need to write a C# method that uses the same logic to calculate the discount, how would you write such a program?

Quantity < 10	Y	N	N	N
Quantity < 50	Y	Y	N	N
Quantity < 100	Y	Y	Y	N
Discount	5%	10%	15%	20%

1. Create a new Visual C# console application project named Discount.

2. Replace the code in the class file with the following:

```
public class Discount
{
    static public void Main()
    {
        Console.Write("Please enter quantity: ");
        int quantity =
   Int32.Parse(Console.ReadLine());
        Console.WriteLine("Your discount is: {0}%",
            CalculateDiscount(quantity));
    }

    private static int CalculateDiscount(int
   quantity)
    {
        int discount;

        if (quantity >= 100)
            discount = 20;
```

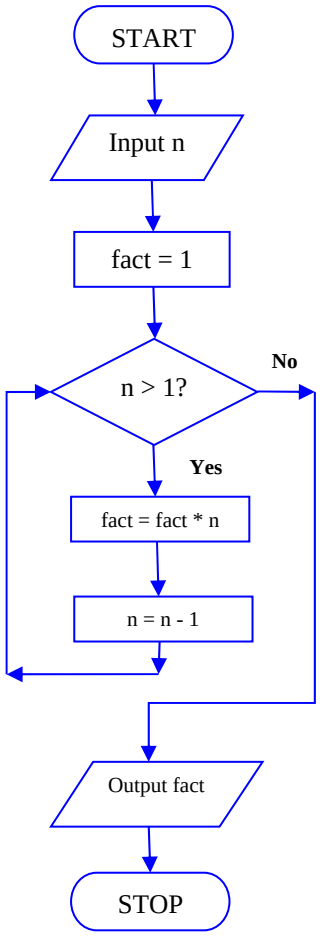
```
            else if (quantity >= 50)
                discount = 15;
            else if (quantity >= 10)
                discount = 10;
            else
                discount = 5;

            return discount;
        }
    }
```

3. Build and run the project. Enter a quantity and match the calculated discount with the decision table.

Project 1-2: Converting a Flowchart into a C# Program

You are developing a library of mathematical functions. You first develop the following flowchart describing the algorithm for calculating the factorial of a number. You need to write an equivalent C# program for this flowchart. How would you write such a program?



1. **Create a new Visual C# console application project named Factorial.**
2. **Replace the code in the class file with the following:**

```
public class Factorial
{
    static public void Main()
    {
        Console.Write("Please enter a number: ");
        int number = Int32.Parse(Console.ReadLine());
        Console.WriteLine("Factorial of {0} is: {1}",
            number, ComputeFactorial(number));
    }

    private static int ComputeFactorial(int number)
    {
        int factorial = 1;

        while (number > 1)
        {
            factorial *= number--;
        }

        return factorial;
    }
}
```

3. **Build and run the project. Enter a number and match the results of the program with the dry run of the flowchart.**

Proficiency Assessment

Project 1-3: Handling Exceptions

You are writing code for a simple arithmetic library. You decide to create a method named Divide that takes two arguments, x and y, and returns the value of x/y. You need to catch any arithmetic exceptions that might be thrown for errors in arithmetic, casting, or data type conversions. You also need to catch any other exceptions that may be thrown from the code. To address this requirement, you need to create properly structured exception-handling code. How would you write such a program?

1. **Create a new Visual C# console application project named ExceptionTest.**
2. **Replace the code in the class file with the following:**

```
class ExceptionTest
{
   public static void Main()
   {
      int x = 10;
```

```
        int y = 0;
        Console.WriteLine(Divide(x, y));
    }

    public static int Divide(int x, int y)
    {
        int results = 0;
        try
        {
            results = x/y;
        }
        catch (ArithmeticException e)
        {
            Console.WriteLine(
             "ArithmeticException Handler: {0}",
             e.ToString());
        }
        catch (Exception e)
        {
            Console.WriteLine(
             "Generic Exception Handler: {0}",
             e.ToString());
        }
        return results;
    }
}
```

3. **Build and run the project.**

Project 1-4: Creating a Recursive Algorithm

You are developing a library of utility functions for your application. You need to write a method that takes an integer and counts the number of significant digits in it. You need to create a recursive program to solve this problem. How would you write such a program?

1. **Create a new Visual C# console application project named Reverse.**
2. **Replace the code in the class file with the following:**

```
public class Reverse
{
    static public void Main()
    {
        Console.Write("Please enter a number: ");
        int number = Int32.Parse(Console.ReadLine());
        Console.WriteLine("The number {0} has {1}
    digits",
            number, CountDigits(number));
```

```
    }

    private static int CountDigits(int number)
    {
        if (number / 10 == 0)
            return 1;
        else
            return 1 + CountDigits (number / 10);
    }
}
```

3. Build and run the project. Enter a number, and check the number of digits calculated by the program.

Lesson 2

Introduction to Object-Oriented Programming

Learning Objectives

Students will learn about:

- Objects
- Values and References
- Encapsulation
- Inheritance
- Polymorphism
- Interfaces

ODN Skills

• Understand the fundamentals of classes.	2.1
• Understand computer storage and data types.	1.1
• Understand encapsulation.	2.4
• Understand inheritance.	2.2
• Understand polymorphism.	2.3
• Understand encapsulation.	2.4

Lesson Summary — Lecture Notes

Lesson 2 introduces students to the concepts of object-oriented programming. A good way to start the discussion is to have students think about the day-to-day objects they interact with (for example, a music player or a car) and talk about the properties and behaviors of those objects. Then extend the discussion to the objects as they relate to programming.

Start the discussion by presenting the various features of a class, such as data fields, properties, and methods. Then talk about how to create instances of a class. Ensure that students can distinguish between a class and an object. At this point, introduce the this keyword followed by a discussion on the static members.

The next section talks about the value types and reference types. This section introduces the struct and talks about how memory is allocated differently for a value type and reference type. You can demonstrate how assigning a reference is different from assigning a value. You may want to present multiple examples here to ensure students correctly understand the difference between references and values.

In the next section, you will discuss the concept of encapsulation. Here again you can provide a real-world example and then show how that applies to computer objects. Have students think from two different points of view – the class developer point of view and the class user point of view. Next, talk about the access modifiers and how the different access modifiers can limit the access of a data item or a method to only certain areas of the code.

Next, you talk about the concept of inheritance. Emphasize on the benefits of reusing the code. Talk about how the concept of encapsulation goes hand-in-hand with inheritance. Again, take an example from the real world and then segue to programming. Make sure students know how to override the behavior of the base classes. Give an example that creates a chain of inheritance. Explain to students that C# doesn't support inheriting from multiple objects.

Polymorphism is the final concept covered in this lesson. Make sure students understand that programmers don't need to know the exact type of an object in advance. The run-time determines the exact type at runtime and calls the methods or properties specific to the type at runtime. Students need to know how and when to use the new, virtual, and the overrides keywords to provide specific functionality. Finally, talk about interfaces. Make sure students can distinguish when to use inheritance and when to use an interface.

Key Terms

accessors—The accessor of a property contains the statements associated with getting or setting the property.

abstract class—Provides a common definition of a base class that can be shared by multiple derived classes.

constructor—A special class method that is executed when a new instance of a class is created. It is used to initialize the data members of the object.

delegate–A special object that can hold a reference to a method with a specific signature.

encapsulation—An information-hiding mechanism that makes code easy to maintain and understand.

event–A way for a class to notify other classes or objects when something of interest happens. The class that sends the notification is called a publisher of the event. The class that receives the notification is called the subscriber of the event.

inheritance—A feature of object-oriented programming that allows you to develop a class once and then reuse that code over and over as the basis of new classes.

interface—Used to establish a contract through which objects can interact with each other without knowing the implementation details.

polymorphism—The ability of derived classes to share common functionality with base classes but still define their own unique behavior.

property–A class member that can be accessed like a data field but contain code like a method.

reference type—Only stores a reference to an actual value in the memory.
value type–Directly stores a value in the memory.

Lesson 2

Introduction to Object-Oriented Programming

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You want to restrict the access for a method to the containing class or to a class that is derived from the containing class. Which access modifier should you use for this method?

   a. public

   b. private

   c. protected

   d. internal

2. In a class, you defined a method called Render. This method provides functionality to render bitmap files on the screen. You would like the derived classes to supersede this functionality to support the rendering of additional image formats. You also want the Render method of the derived classes to be executed even if a derived class is cast as the base class. Which keyword should you use with the definition of the Render method in the base class?

   a. abstract

   b. virtual

   c. new

   d. overrides

3. You defined a class AdvMath that defines advanced mathematical functionality. You do not want the functionality of this class to be inherited into derived classes. What keyword should you use to define the AdvMath class?

   a. sealed

   b. abstract

   c. private

   d. internal

4. You need to provide query functionality to several of your classes. Each class's algorithm for the query will likely be different. Also, not all the classes have an "is-a" relationship with each other. How should you support this functionality?

   a. Add the query functionality to a base class with public access modifier.

   b. Have all the classes inherit from an abstract base class and override the base-class method to provide their own query functionality.

c. Have all the classes inherit from a base class that provides the query functionality.

d. Create a common interface that is implemented by all the classes.

5. Which of the following class elements should you use to define the behavior of a class?

a. Method

b. Property

c. Event

d. Delegate

6. You are writing code for a class named Product. You need to make sure that the data members of the class are initialized to their correct values as soon as you create an object of the Product class. The initialization code should be always executed. What should you do?

a. Create a static method in the Product class to initialize data members.

b. Create a constructor in the Product class to initialize data members.

c. Create a static property in the Product class to initialize data members.

d. Create an event in the Product class to initialize data members.

7. You are creating a new class named Square that is derived from the Polygon class. The Polygon class has the following code:

```
class Polygon
{
    public virtual void Draw()
    {
        // additional code...
    }
}
```

The Draw method in the Square class should provide new functionality but also hide the Polygon class implementation of the Draw method. Which code segment should you use to accomplish this?

a.

```
class Square : Polygon
{
    public override void Draw()
    {
        // additional code ...
    }
}
```

b.

```
class Square : Polygon
{
    public new void Draw()
```

```
    {
        // additional code ...
    }
}
```

c.

```
class Square : Polygon
{
    public virtual void Draw()
    {
        // additional code ...
    }
}
```

d.

```
class Square : Polygon
{
    public static void Draw()
    {
        // additional code ...
    }
}
```

8. You are creating a new class named Rectangle. You write the following code:

```
class Rectangle : IComparable
{
    public double Length { get; set; }
    public double Width { get; set; }

    public double GetArea()
    {
        return Length * Width;
    }

    public int CompareTo(object obj)
    {
        // to be completed
    }
}
```

You need to complete the definition of the CompareTo method to enable comparison of the Rectangle objects. Which of the following code should you write?

a.

```
public int CompareTo(object obj)
{
    Rectangle target = (Rectangle)obj;
    double diff = this.GetArea() -
target.GetArea();
```

```
        if (diff == 0)
            return 0;
        else if (diff > 0)
            return 1;
        else return -1;
    }
```

b.

```
    public int CompareTo(object obj)
    {
        Rectangle target = (Rectangle)obj;
        double diff = this.GetArea() -
target.GetArea();

        if (diff == 0)
            return 1;
        else if (diff > 0)
            return -1;
        else return 0;
    }
```

c.

```
    public int CompareTo(object obj)
    {
        Rectangle target = (Rectangle)obj;

        if (this == target)
            return 0;
        else if (this > target)
            return 1;
        else return -1;
    }
```

d.

```
    public int CompareTo(object obj)
    {
        Rectangle target = (Rectangle)obj;

        if (this == target)
            return 1;
        else if (this > target)
            return -1;
        else return 0;
    }
```

9. You are writing code for a new method named Process:

```
void Process(object o)
{


}
```

The code receives a parameter of type object. You need to cast this object into the type Rectangle. At times, the value of o that is passed to the method might not be a valid Rectangle value. You need to make sure that the code does not generate any System.InvalidCastException errors while doing the

conversions. Which of the following lines of code should you use inside the Process method to accomplish this goal?

a. `Rectangle r = (Rectangle) o;`

b. `Rectangle r = o as Rectangle;`

c. `Rectangle r = o is Rectangle;`

d.
```
Rectangle r = (o != null) ? o as rectangle :
        (Rectangle) o;
```

10. You are writing code to handle events in your program. You define a delegate named RectangleHandler like this:

```
public delegate void RectangleHandler(Rectangle
        rect);
```

You also create a variable of the RectangleHandler type as follows:

```
RectangleHandler handler;
```

Later in the program, you need to add a method named DisplayArea to the method invocation list of the handler variable. The signature of the DisplayArea method matches the signature of the RectangleHandler method. Any code that you write should not affect any existing event-handling code. Given this restriction, which of the following code should you write?

a. `handler = new RectangleHandler(DisplayArea);`

b. `handler = DisplayArea;`

c. `handler += DisplayArea;`

d. `handler -= DisplayArea;`

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. A(n) **class** is a blueprint of an object.
2. A class that does not provide a complete implementation must be declared with the keyword **abstract**.
3. Classes that want to support comparison must implement the IComparable interface and then provide a body for the **CompareTo** method.
4. You can use the **is** operator to check whether it is legal to cast one type to another type.
5. Three main features of an object-oriented programming language are **encapsulation**, **inheritance**, and **polymorphism**.
6. You can use **namespaces** to group related classes in order to reduce name collisions.
7. The **this** keyword refers to the current instance of a class.
8. A(n) **delegate** is a type that references a method.
9. A(n) **struct** is a value type, whereas a(n) **class** is a reference type.
10. You can use the **static** keyword to declare a member that belongs to the class itself rather than to a specific object.

Competency Assessment

Scenario 2-1: Creating Properties

You need to create a class named Product that represents a product. The class has a single property named Name. Users of the Product class should be able to get and set the value of the Name property. However, any attempt to set the value of Name to an empty string or a null value should raise an exception. Also, users of the Product class should not be able to access any other data members of the Product class. How will you create such a class?

1. Create a new Visual C# class named Product.cs.

2. Replace the code for the Product class with the following code:

```
public class Product

{
    private string name;
    public string Name
    {
        get
        {
            return name;
        }
        set
        {
            if (string.IsNullOrEmpty(value))
            {
                throw new ArgumentException(
                "Name cannot be null or empty string",
                "Name");
            }
            name = value;
        }
    }
}
```

3. Select Build > Build Solution to compile the code.

Scenario 2-2: Creating a Struct

You are developing a game that needs to represent the location of a target in the three-dimensional space. The location is identified by the three integer values denoted x, y, and z. You will create thousands of these data structures in your program and you need a lightweight, efficient way to store this data in memory. Also, it is unlikely that you will need to inherit any other types from this location type. How should you represent the location in your program?

1. Create a new Visual C# class file named Location.cs

2. Replace the code for the Location class with the following code:

```
public struct Location
{
    public int x, y, z;

    public Location(int x, int y, int z)
    {
        this.x = x;
        this.y = y;
        this.z = z;
    }
}
```

3. Select Build > Build Solution to compile the code.

Proficiency Assessment

Project 2-1: Overriding the ToString Method

You are writing code for a Product class. The Product class contains the name and price of a product. You need to override the base class (System.Object) method ToString to provide information about the objects of the product class to the calling code. What code do you need to write for the Product class in order to meet this requirement?

1. Create a new Visual C# class named Product.cs.

2. Replace the definition of the Product class with the following code:

```
class Product
{
  string name;
  double price;
  Product(string name, double price)
  {
     this.name = name;
     this.price = price;
  }
  public override string ToString()
  {
     string s = price.ToString();
     return "Product: " + name + " " + s;
  }
}
```

3. Select Build > Build Solution to compile the code.

Project 2-2: Creating and Handling Events

You are writing code for creating and handling events in your program. The class SampleClass needs to implement the following interface:

```
public delegate void SampleDelegate();

public interface ISampleEvents
{
    event SampleDelegate SampleEvent;
    void Invoke();
}
```

You need to write code for the SampleClass and for a test method that creates an instance of the SampleClass and invokes the event. What code should you write?

1. Create a new Visual C# class named SampleClass.cs, then replace the code for the Sample class with the following code:

```
public class SampleClass : ISampleEvents
{
    public event SampleDelegate SampleEvent;
    public void Invoke()
    {
        if (SampleEvent != null)
            SampleEvent();
    }
}
```

2. Add another class named TestClass to the file:

```
public class TestClass
{
    static public void TestHandler()
    {
        Console.WriteLine(
"TestHandler is called when the event fires.");
    }

    static public void Main()
    {
        ISampleEvents se = new SampleClass();

        se.SampleEvent += new
   SampleDelegate(TestClass.TestHandler);
        se.Invoke();
    }
}
```

3. Build the code and run the project to verify that the event handler is executed.

Lesson 3

Understanding General Software Development

Learning Objectives

Students will learn about:

- Application Lifecycle Management
- Testing
- Data Structures
- Sorting Algorithms

ODN Skills

- Understand application lifecycle management. 3.1
- Understand algorithms and data structures. 3.3

Lesson Summary — Lecture Notes

This lesson covers two separate topics:

1. Application lifecycle management.
2. Algorithms and data structures.

You should emphasize the fact that software development is only one activity in the complete application lifecycle. A whole range of people perform various activities in order to turn an idea into reality. This lesson discusses activities such as design, development, testing, and release.

The next topic covers the software testing activity in detail. At the end of the discussion, students should before familiar with approaches to testing. You should point out that complex business applications have a large number of possible execution paths. Many of these execution paths depend on the runtime data. It is almost impossible to guarantee that the application will have zero defects. As part of testing, you verify that the software works as expected for the known execution paths.

The next section of the lesson discusses data structures. The objective here is to use data structures as a tool for understanding storage patterns and for problem solving. The section discusses the following data structures: array, queues, stacks, and linked lists. You should talk about the common operations for each of these data structures. For each operation, help students visualize how the operation affects the internal storage of the data structure. Also discuss the algorithm that the data structure may use for each of these operations.

You may also point out that an implementation for each of these data structures is also available as part of the .NET Framework class libraries.

The final topic in this lesson covers sorting algorithms. This section talks about two sorting algorithms, BubbleSort and QuickSort. The BubbleSort algorithm uses a series of comparison and swap operations to arrange the elements of a list in the correct order. The QuickSort algorithm uses partitioning and comparison operations to arrange the elements of a list in the correct order.

You may highlight that there are different approaches to solving the same problem and the solution that you choose may depend on your specific requirements. You should take some time to compare the BubbleSort algorithm with QuickSort but at this level there is no need to get into the discussion of the BigO notation and complexity analysis.

Key Terms

arrays—A collection of items stored in a contiguous memory location and addressed using one or more indices.

black-box testing—Mostly used to make sure a software application covers all its requirements.

BubbleSort—A sorting algorithm that uses a series of comparison and swap operations to arrange the elements of a list in the correct order.

linked lists—A collection of nodes arranged so that each node contains a link to the next node in the sequence.

QuickSort—A sorting algorithm that uses the partitioning and comparison operations to arrange the elements of a list in the correct order.

queues—A collection of items in which the first item added to the collection is the first one to be removed.

regression testing—A type of testing that ensures the functionality that was already known to work correctly is still working with every new build.

stacks—A collection of items in which the last item added to the collection is the first one to be removed.

unit testing—Verifies the functionality of a unit of code.

white-box testing—Used to make sure that each method or function works as expected.

Lesson 3

Understanding General Software Development

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. The product that you are developing is not yet finished, but you would like to release the product to a wider customer audience for feedback and testing. Under which of the following testing levels would this activity fall?

   a. Integration testing

   b. System testing

   c. Acceptance testing

   d. Regression testing

2. The testers of a software application have access to its source code and they plan to write test cases that ensure that the methods return correct values. Which of the following testing levels will this activity fall under?

   a. Integration testing

   b. Unit testing

   c. Alpha testing

   d. Beta testing

3. Which of the following data structures allows direct access to all of its items?

   a. Array

   b. Stack

   c. Queue

   d. Linked list

4. Which of the following activities in the application lifecycle is used by an architect to create the technical blueprint of a system?

   a. Requirements analysis

   b. Design

   c. Development

   d. Maintenance

5. In your application, you are using a queue data structure to manipulate information. You need to find which data item will be processed next, but you don't want to actually process that data item yet. Which of the following queue operations will you use?

   a. Enqueue

b. Dequeue

c. Peek

d. Contains

6. You are developing a program that requires you to track the method calls. You can only invoke one method at a time. However, a method call may in turn invoke other methods. When a method ends, it returns control back to the calling method. Which data structure should you use to keep track of these method calls?

   a. Queue

   b. Array

   c. Linked list

   d. Stack

7. You are developing a program that simulates a job queue. Often, the jobs come faster than you can process them, and in such case, the jobs wait for their turn to be processed. You need to make sure that the job that arrived first is the first to be processed as well. Which of the following data structures is best suited for this requirement?

   a. Array

   b. Queue

   c. Linked list

   d. Stack

8. You write the following code in a program:

```
int[] numbers = {2, 3, 1, 4};
numbers[2] = 4;
```

   What will be the contents of the array after the second statement is executed?

   a. {2, 4, 1, 4}

   b. {2, 3, 4, 4}

   c. {2, 4, 1, 2}

   d. {4, 3, 1, 4}

9. You are developing a program that performs frequent insert and delete operations on the data. Your requirement also dictates the capability to access previous and next records when the user presses the previous or next button. Which of the following data structures will best suit your requirements?

   a. Array

   b. Circular linked list

   c. Linked list

   d. Doubly linked list

10. You are developing a program that performs frequent insert and delete operations on the data. The data items need to be accessed like a stack with

last-in, first-out functionality. Your solution must require as little memory as possible. Which of the following data structures will best suit these requirements?

a. Array

b. Circular linked list

c. Linked list

d. Doubly linked list

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. In **white-box** testing, testers use their knowledge of system internals to assess the system.
2. Usually, with every new fix, software testers run a battery of **regression tests** to make sure that all functionality that was known to be working is still working.
3. The BubbleSort algorithm uses a series of **comparison** and **swap** operations to arrange the elements of a list in the correct order.
4. A(n) **stack** is a collection of items in which the last item added to the collection is the first one to be removed.
5. **Requirements analysis** is the process of determining the detailed business requirements for a new software system.
6. A linked list is a collection of nodes such that each node contains a(n) **reference (or link)** to the next node in the sequence.
7. The **enqueue** operation adds an item to a queue, whereas the **dequeue** operation removes an item from a queue.
8. The QuickSort algorithm uses **partitioning** and comparison operations to arrange the elements of a list in the correct order.
9. A(n) **business analyst** is responsible for analyzing business needs and converting them into requirements that can be executed by the development team.
10. Alpha testing and beta testing both are part of the **acceptance** testing of a system.

Competency Assessment

Project 3-1: Using Arrays

You are writing a program that uses a two-dimensional array. The array has four rows and five columns. You need to print the largest element in each row of the array. How would you write such a program?

1. Create a new project based on the Console Application template. Name the project Project03_01.

2. Replace the code for the Program.cs file with the following:

```
using System;
namespace Project03_01
{
```

```
class Program
{
    static void Main(string[] args)
    {
        int[,] numbers = new int[,]
        {
            { 11, 7, 50, 45, 27  },
            { 18, 35, 47, 24, 12 },
            { 89, 67, 84, 34, 24 },
            { 67, 32, 79, 65, 10 }
        };

        for (int row = 0;
             row < numbers.GetLength(0);
             row++)
        {
            Console.WriteLine(
              "Maximum number in the row {0}: {1}",
                row, FindMax(row, numbers));
        }
        Console.WriteLine(
        "Press any key to continue...");
        Console.ReadKey();
    }

    static int FindMax(int row, int[,] numbers)
    {
        int max = numbers[row, 0];
        for (int column = 0;
             column < numbers.GetLength(1);
             column++ )
        {
            if (numbers[row, column] > max)
                max = numbers[row, column];
        }
        return max;
    }
}
}
```

3. **Select Debug > Start Debugging (or press F5) to run the project. You will see the following output displayed in the console window:**

```
Maximum number in the row 0: 50
Maximum number in the row 1: 47
Maximum number in the row 2: 89
```

```
Maximum number in the row 3: 79
```

Project 3-2: Using Queues

You are writing a program that uses two queues. The data in each queue is already in ascending order. You need to process the contents of both queues in such a way that the output is printed on the screen in sorted order. How would you write such a program?

1. **Create a new project based on the Console Application template. Name the project Project03_02.**
2. **Replace the code for the Program.cs file with the following:**

```
using System;
using System.Collections;

namespace Project03_02
{
    class Program
    {
        static void Main(string[] args)
        {
            Queue first = new Queue();
            first.Enqueue(7);
            first.Enqueue(11);
            first.Enqueue(45);
            first.Enqueue(50);

            Queue second = new Queue();
            second.Enqueue(12);
            second.Enqueue(32);
            second.Enqueue(65);
            second.Enqueue(67);

            ProcessInOrder(first, second);

            Console.WriteLine(
                "Press any key to continue...");
            Console.ReadKey();
        }

        static void ProcessInOrder(Queue first,
            Queue second)
        {
            while (first.Count > 0 || second.Count > 0)
            {
                if (first.Count == 0)
```

```
                    {
                        Console.WriteLine(second.Dequeue());
                        continue;
                    }

                    if (second.Count == 0)
                    {
                        Console.WriteLine(first.Dequeue());
                        continue;
                    }

                    if ( (int) first.Peek()
                        >= (int) second.Peek())
                    {
                        Console.WriteLine(
                            second.Dequeue());
                    }
                    else
                    {
                        Console.WriteLine(first.Dequeue());
                    }
                }
            }
        }
    }
```

3. **Select Debug > Start Debugging (or press F5) to run the project. Notice that the numbers are displayed in ascending order in the console window.**

Proficiency Assessment

Project 3-3: Using Stacks

You are writing a program that uses two stacks. The data in each stack is already in descending order. You need to process the contents of both stacks in such a way that the output is printed on the screen in ascending order. How would you write such a program?

1. **Create a new project based on the Console Application template. Name the project Project03_03.**
2. **Replace the code for the Program.cs file with the following:**

```
using System;
using System.Collections;

namespace Project03_03
{
```

```
class Program
{
    static void Main(string[] args)
    {
        Stack first = new Stack();
        first.Push(50);
        first.Push(45);
        first.Push(11);
        first.Push(7);

        Stack second = new Stack();
        second.Push(67);
        second.Push(65);
        second.Push(32);
        second.Push(12);

        ProcessInOrder(first, second);

        Console.WriteLine(
            "Press any key to continue...");
        Console.ReadKey();
    }

    static void ProcessInOrder(Stack first,
        Stack second)
    {
        while (first.Count > 0 || second.Count > 0)
        {
            if (first.Count == 0)
            {
                Console.WriteLine(second.Pop());
                continue;
            }

            if (second.Count == 0)
            {
                Console.WriteLine(first.Pop());
                continue;
            }

            if ((int)first.Peek()
                >= (int)second.Peek())
            {
                Console.WriteLine(
                    second.Pop());
```

```
                }
                else
                {
                    Console.WriteLine(first.Pop());
                }
            }
        }
    }
}
```

3. **Select Debug > Start Debugging (or press F5) to run the project. Notice that the numbers are displayed in ascending order in the console window.**

Project 3-4: Using Linked Lists

You are writing a program that stores a list of product names in a linked list. The user will enter a product name and your program needs to check whether the linked list contains the given product. How would you write such a program?

1. **Create a new project based on the Console Application template. Name the project Project03_04.**
2. **Replace the code for the Program.cs file with the following:**

```
using System;
using System.Collections.Generic;

namespace Project03_04
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] words = {
                    "Konbu", "Tofu",
                    "Pavlova", "Chocolate",
                    "Ikura" };
            LinkedList<string> list =
                new LinkedList<string>(words);

            Console.WriteLine("Enter product to find:");
            Console.WriteLine(
                "Type <end> to end the program");
            string searchText = Console.ReadLine();
            while (searchText != "<end>")
            {
                if (list.Contains(searchText))
```

```
                    {
                        Console.WriteLine(
                            "The search text was found");
                    }
                    else
                    {
                        Console.WriteLine(
                         "The search text was NOT found");
                    }

                    Console.WriteLine(
                      "Enter product to find:");
                    Console.WriteLine(
                        "Type <end> to end the program\n");
                    searchText = Console.ReadLine();
                }
            }
        }
    }
```

3. **Select Debug > Start Debugging (or press F5) to run the project. Type a product name to find in the console window and press the Enter key. The program will display a message saying that the product was found or not found. Notice that the search is case sensitive. Type <end> and press Enter to end the program.**

Lesson 4

Understanding Web Applications

Learning Objectives

Students will learn about:

- Web Page Development
- ASP.NET Application Development
- IIS Web Hosting
- Web Services Development

ODN Skills

- Understand Web page development. 4.1
- Understand Microsoft ASP.NET Web application development. 4.2
- Understand Web hosting. 4.3
- Understand Web services. 4.4

Lesson Summary — Lecture Notes

The focus of Lesson 4 is on the basics of Web development. Before you start, make sure students have a working setup of either Visual Studio 2010 or Visual Web Developer Express edition. You will also need Internet Information Services (IIS) to demonstrate the administration of Web sites and virtual directories.

The first part of the lesson covers the basic Web development concepts. Talk about a Web page and how the Web page is delivered from a server and displayed on the user's computer in a Web browser. Next, talk about HTML, CSS and JavaScript. Emphasize the importance of separating markup, formatting, and code from each other. Ensure students understand that the HTML, CSS and JavaScript are all executed on the client-side within a Web browser.

The next section covers the basics of Web development by using ASP.NET. Start this section by discussing how ASP.NET pages are executed. Next, create a simple ASP.NET page and walk students through the basic structure of the program and its execution. Help students understand what code is executed on the server and what code is sent to the client-side for execution. Next, help students understand how to configure virtual directory on IIS and how you'll deploy a Web site on IIS.

The next topic in this section covers state management. Discuss why you need to think about state management in Web development. Discuss both the client-side and the server side state management techniques. Talk about the features of each of the techniques and help students identify the scenarios where one state management technique is better than the other.

The final section covers the basics of developing and consuming Web services. In this section, first discuss how a Web service is different from a Web application. Next, discuss some popular examples of Web services. Next, talk about the key technologies that make the Web services possible—XML, HTTP, SOAP, and so on. Describe WSDL and how it is used. Walk students through developing a simple Web service by using Visual Studio/Visual Web Developer Express Edition. First show students how to test a Web service from within the development environment. Then show them how to call a Web service from a client application. Make sure students understand what a Web reference is and how a Web reference is different from referencing a local assembly.

Key Terms

cascading style sheets (CSS)—Enable you to store a Web page's style and formatting information separate from the HTML code. This separation makes it easier to update the look and feel of your Web site.

Hypertext Markup Language (HTML)—The language used by Web servers and browsers to describe a Web page.

Internet Information Services (IIS)—An integral part of Windows Server operating systems that provides functionality for hosting Web sites.

JavaScript—A client-side scripting language that runs inside Web browsers to help create interactive Web pages.

SOAP—The protocol for exchanging structured information in a Web service communication between two remote computers.

state management—An important issue for Web applications because of the disconnected nature of HTTP. There are both client-side techniques and server-side techniques available for state management.

Web service—A software component that can be accessed over a network using standard network protocols such as HTTP. Web services are described using the Web Service Description Language (WSDL).

Web Service Description Language (WSDL)—An XML-based language for describing Web services.

Lesson 4

Understanding Web Applications

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You write the following code for your Web page:

```
<html>
   <head>
      <title>Sample Page</title>
      <style type="text/css">
      div
      {
          font-family: Verdana;
          font-size: 9pt;
      }
      </style>
   </head>
   <body>
      <div style=
         "font-weight: bold; font-size: 12pt;">
      Sample Text</div>
   </body>
</html>
```

What would be the style for the text displayed as part of the element?

   a. **Font family: Verdana; font weight: bold; font size: 12pt**

   b. Font family: Verdana; font weight: bold; font size: 9pt

   c. Font family: Verdana; font size: 12pt

   d. Font family: Verdana; font size: 9pt

2. You are developing a mapping Web site that allows users to interactively explore maps using actions such as panning and zooming. You want the Web site to be responsive and accessible in most modern Web browsers. However, you do not want users to have to install additional plug-ins in order to use your Web site. Which of the following technologies should you use to display maps?

   a. HTML

   b. Server-side programming technology such as ASP.NET

   c. Adobe Flash

   d. JavaScript

3. Your ASP.NET page contains a page-level variable of Customer type. You want to preserve the value of this variable across page postbacks, but you

do not need this variable in any other page in the application. Which of the following state-management techniques is the best way to achieve this?

a. Query strings

b. Cookies

c. ViewState

d. Session

4. You are developing a Web application for an online bank. Your application enables users to access their account information and transactions from within a Web browser. When a user logs onto the Web application, you want to show the username and account balance on all pages of the application until the user logs off. You also want this application to be safe from malicious users. Which of the following state-management techniques should you use?

   a. Cookies

   b. ViewState

   c. ViewState with encryption

   d. Session

5. You are developing a Web form to display weather information. When a user requests the Web form, the form needs to perform some initialization to change its appearance and assign values to some controls. Where should you put the code?

   a. In the PreInit event handler of the Page class

   b. In the Init event handler of the Page class

   c. In the Load event handler of the Page class

   d. In the PreRender event handler of the Page class

6. You want to display values of C# expressions in an ASP.NET page. Which of the following types of code blocks should you use to enclose the expression?

   a. <script runat="server">…</script>

   b. <script>…</script>

   c. <%= … %>

   d. <form>…</form>

7. You have developed a timesheet application that will be used by all employees in your company. You used ASP.NET to develop this application and have deployed it on the company's Web server. What must all employees of the company install on their computers before they can access the timesheet application?

   a. .NET Framework Redistributable

   b. .NET Framework Software Development Kit

   c. Visual Studio

   d. A Web browser

8. Your client application calls a Web service that performs complex, time-consuming calculations. A user complains that while results are being returned, the user interface freezes momentarily. Which approach should you take to solve this issue?

a. You should install a better processor on the Web server.

b. You should install a better processor on the client computer.

c. You should upgrade to a faster Internet connection.

d. You should use asynchronous calls to invoke the Web service.

9. You have created an ASP.NET Web service that converts one currency into another. One of the methods in your Web service is defined with the following code:

```
public double Convert(double amount,
    string from, string to)
{
    // code to perform currency conversion
}
```

The users of the Web service report that they can set a reference to the Web service but the Convert method is not available to them. What could be the problem?

a. The .asmx file for the Web service is not available on the Web server.

b. The Web service class is not marked with the WebService attribute.

c. The Convert method is not marked with the WebMethod attribute.

d. Web services can only expose methods that return text values.

10. You are working on two Visual Studio projects. The first project is a Web service that returns a DataSet object belonging to the System.Data namespace. The second project accesses the Web service created by the first project. Which project in this scenario requires a reference to the System.Data namespace?

a. The Web service project.

b. The client project that accesses the Web service.

c. Both the client project and the Web service project.

d. Neither the client project nor the Web service project.

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. In the HTML anchor tag (<a>), the **href** attribute specifies the target URL.
2. You can put CSS code in a separate file and link it to a Web page through use of the HTML **<link>** element.
3. The JavaScript code on a Web page is executed on the **client-side**.
4. You can use a(n) **<noscript>** element to display a specific message to users when their browser is not running JavaScript.
5. You can disable ViewState at the page level by setting the **EnableViewState** attribute of the Page directive to false in the ASP.NET page.
6. The **Application** state is used to store data that is used globally throughout an application, as opposed to the **Session** state, which stores data for a user session.
7. A Web application is accessed using a(n) **virtual directory** name instead of a physical folder name.

8. You must mark classes with the **WebService** attribute to expose them as a Web service.
9. Of all the methods in a Web service class, only those marked with **WebMethod** attributes are exposed as Web service methods.
10. SOAP relies on **XML** as its message format and uses **HTTP** for message transmission.

Competency Assessment

Project 4-1: Using JavaScript and HTML

You are developing a Web page that provides a responsive user interface. You decide to display an image on the page. When the user moves her mouse over the image, the original image is replaced with a new image. Then, when the mouse moves out of the image area, the original image is displayed again. You hope to accomplish this using client-side JavaScript and HTML code. How would you create a Web page that works as described here?

1. **Open Visual Studio. Create a new project based on the ASP.NET Empty Web Application template. Name the project Project04_01.**
2. **Add a new HTML page named default.htm to the project.**
3. **Add two image files (RedBox.png and BlueBox.png) to the project. These image files are a square filled with a solid color. RedBox.png has a square filled with red. Similarly, the BlueBox.png file is a square filled with blue. You can create these files using the Windows Paint application.**
4. **Replace the default code in the HTML page with the following:**

```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
   <head>
        <title></title>
       <script type="text/javascript"
               language="javascript">
         <!--
           function SetMouseOverImage() {
               document.boxImage.src = "BlueBox.png";
           }

           function SetMouseOutImage() {
               document.boxImage.src = "RedBox.png";
           }
         -->
       </script>
```

```
    </head>
    <body>
        <p align="center">
            <img name="boxImage" src="RedBox.png"
                 alt="this image changes on mouse over"
                 width="133" height="100"
                 onmouseover="SetMouseOverImage()"
                 onmouseout="SetMouseOutImage()" />
         </p>
         <p>
         Move the mouse over the red color box above.
         You will see that the image changes to the
         blue color. Now, move the mouse out of the image
         area. The image will change back to its original
         red color.
         </p>
    </body>
</html>
```

5. **Select Debug > Start Debugging (or press F5) to run the project. Move the mouse pointer over the image. You should see the red-color image being replaced by a blue-color image. Then, when you move the mouse pointer out of the image area, the red-color image should be restored.**

Project 4-2: Using Query Strings

You are developing a portion of a Web site that allows users to enter their names and email addresses to subscribe to an email newsletter. Your solution consists of two Web pages. The first page collects the user name and email address and then transfers control to a second page. The second page accepts the name and the email address as query-string parameters and displays a confirmation message to the user. You need to write code for these two pages. What code will you write to accomplish this requirement?

1. **Create a new project based on the ASP.NET Empty Web Application template. Name the project Project04_02.**
2. **Add a new item to the project based on the Web Form template. Name the Web form default.aspx.**
3. **Replace the HTML in the default.aspx file with the following code:**

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="default.aspx.cs"
    Inherits="Project04_02._default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Enter Name: <asp:textbox ID="NameTextBox"
   runat="server" />
    </div>
        <div>
        Enter Email: <asp:textbox ID="EmailTextbox"
   runat="server" />
    </div>
    <div />
    <div>
        <asp:Button ID="SubscribeButton" runat="server"
          Text="Subscribe"
   onclick="SubscribeButton_Click"/>
    </div>
    </form>
</body>
</html>
```

4. **Switch to the code view for the Web page (default.aspx.cs) and replace the code with the following:**

```
using System;

namespace Project04_02
{
    public partial class _default : System.Web.UI.Page
    {
        protected void SubscribeButton_Click(
            object sender, EventArgs e)
        {
            Response.Redirect(
                String.Format(
                "Confirm.aspx?Name={0}&Email={1}",
                NameTextBox.Text,
                EmailTextbox.Text));
        }
    }
}
```

5. **Add a new item to the project based on the Web Form template. Name the Web form Confirm.aspx.**
6. **Switch to the code view for the Web page (Confirm.aspx.cs) and replace the code with the following:**

```
using System;

namespace Project04_02
{
    public partial class Confirm : System.Web.UI.Page
    {
        protected void Page_Load(
               object sender, EventArgs e)
        {
            if (Request.QueryString["Email"] != null
                && Request.QueryString["Name"] != null)
            {
                Response.Write(String.Format(
                    "{0}, you are subscribed at: {1}",
                    Request.QueryString["Name"],
                    Request.QueryString["Email"]));
            }
        }
    }
}
```

7. **Select Debug > Start Debugging (or press F5) to run the project. Enter a name and email address and click the Subscribe button. Notice that you get transferred to the Confirm.aspx page. Take note of the URL in the Web browser's address bar. The query-string parameters name and email are part of the URL. Also notice that the Confirm.aspx page retrieves that query string to display a customized conformation message.**

Proficiency Assessment

Project 4-3: Calling a Web Service Asynchronously

The proxy class generated by Visual Studio for a Web service includes methods for calling the Web service synchronously as well as asynchronously. By default, the application uses the synchronous method. If you prefer asynchronous invocation, you need to call the asynchronous version of the method. The asynchronous versions do not wait for the Web service to return a response and use a callback mechanism to get a response when it is ready. Asynchronous invocation of a Web service may help client applications be more responsive. You want to call the ToLower method of the previously

created TextWebService in an asynchronous fashion. What code would you write for asynchronously invoking a Web service?

1. **Create a new ASP.NET Web Application project named ToLowerAsync.**
2. **Add a Web Reference to the TextWebService Web service. Name the reference textWebService.**
3. **Change the code of the Default.aspx to the following code. Notice here that the Page attribute has an Async property set to true. This property must be set to true before a page can start an asynchronous operation:**

```
<%@ Page Title="Home Page" Language="C#"
    AutoEventWireup="true"
    CodeBehind="Default.aspx.cs"
    Async="true"
    Inherits="ToLowerAsync._Default" %>
<html>
    <head><title>TextWebService Client
    </title></head>
  <body>
    <form id="Form1" runat="server">
        <h2>Test Form For TextWebService</h2>
          <p>
            <asp:TextBox ID="TextBox1"
               runat="server"
               Text="enter text" />
            <br />
            <asp:Button ID="Button1"
               runat="server"
               Text="Async ToLower"
               onclick="Button1_Click" />
          </p>
          <p>
            <strong>Results from ToLower
   method:</strong><br />
            <asp:Label ID="toLowerLabel"
               runat="server"
               Text="Label" ForeColor="Green" />
          </p>
    </form>
  </body>
</html>
```

4. **Add the following code to the code-behind for the Web Form:**

```
protected void Button1_Click(
    object sender, EventArgs e)
{
    var webService =
        new textWebService.TextWebService();
    webService.ToLowerCompleted +=
        new textWebService
            .ToLowerCompletedEventHandler(
           webService_ToLowerCompleted);
    webService.ToLowerAsync(TextBox1.Text);
}

void webService_ToLowerCompleted(object sender,
    textWebService.ToLowerCompletedEventArgs e)
{
    toLowerLabel.Text = e.Result;
}
```

5. **Run the Web application, enter some sample text, and click the Async ToLower button. The final value in the label is displayed by the webService_ToLowerCompleted callback method.**

Project 4-4: Using Session State

You are developing a portion of a Web site that allows users to enter their names and email addresses to subscribe to an email newsletter. Your solution consists of two Web pages. The first page collects the user name and email address, adds them to the session state, and transfers control to a second page. The second page retrieves the name and the email address from the session state and displays a confirmation message to the user. You need to write code for these two pages. What code will you write to accomplish this requirement?

1. **Create a new project based on the ASP.NET Empty Web Application template. Name the project Project04_04.**
2. **Add a new item to the project based on the Web Form template. Name the Web form default.aspx.**
3. **Replace the HTML in the default.aspx file with the following code:**

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="default.aspx.cs"
    Inherits="Project04_04._default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Enter Name: <asp:textbox ID="NameTextBox"
        runat="server" />
    </div>
        <div>
        Enter Email: <asp:textbox ID="EmailTextbox"
        runat="server" />
    </div>
    <div />
    <div>
        <asp:Button ID="SubscribeButton" runat="server"
          Text="Subscribe"
          onclick="SubscribeButton_Click"/>
    </div>
    </form>
</body>
</html>
```

4. **Switch to the code view for the Web page (default.aspx.cs) and replace the code with the following:**

```
using System;

namespace Project04_04
{
    public partial class _default : System.Web.UI.Page
    {
        protected void SubscribeButton_Click(
            object sender, EventArgs e)
        {
            // Add name and email to the session state
            Session["Name"] = NameTextBox.Text;
            Session["Email"] = EmailTextbox.Text;

            Response.Redirect("Confirm.aspx");
        }
    }
}
```

5. **Add a new item to the project based on the Web Form template. Name the Web form Confirm.aspx.**

6. **Switch to the code view for the Web page (Confirm.aspx.cs) and replace the code with the following:**

```
using System;

namespace Project04_04
{
    public partial class Confirm : System.Web.UI.Page
    {
        protected void Page_Load(
               object sender, EventArgs e)
        {
            if (Session["Email"] != null
                && Session["Name"] != null)
            {
                Response.Write(String.Format(
                    "{0}, you are subscribed at: {1}",
                    Session["Name"],
                    Session["Email"]));
            }
        }
    }
}
```

7. **Select Debug > Start Debugging (or press F5) to run the project. Enter a name and email address and click the Subscribe button. Notice that you are transferred to the Confirm.aspx page. Take note of the URL in the Web browser's address bar. The URL does not contain any query-string parameters. Also notice that the Confirm.aspx page retrieves that name and email from the session state to display a customized confirmation message.**

Lesson 5

Understanding Desktop Applications

Learning Objectives

Students will learn about:

- Windows Forms Applications
- Console-Based Applications
- Windows Services

ODN Skills

- Understand Windows Forms applications. 5.1
- Understand console-based applications. 5.2
- Understand Windows services. 5.3

Lesson Summary — Lecture Notes

Lesson 5 is about development of desktop applications. These are the applications that run locally on the user's computer. This chapter covers three different types of applications:

1. Windows applications.
2. Console-based applications.
3. Windows services.

You'll start the lesson by discussing Windows applications. Provide examples of some common Windows applications and then compare Windows applications with Web applications.

Next, demonstrate how to create a simple Windows application. You can walk the students through the Windows Designer, show how to drag and drop controls from the Toolbox, and how to modify properties and attach event-handlers. Make sure students understand how event-handling works and how to handle events.

Next, explain how visual inheritance allows you to reuse existing functionality and layout for Windows Forms. Help students see visual inheritance in action by demonstrating a sample.

Next, demonstrate how a multiple document interface (MDI) application looks and behaves. Discuss the scenarios whereby you may want to create an MDI application rather than a Single Document Interface (SDI) application. Demonstrate how to create a MDI application.

In the next section, talk about console-based applications. Start by discussing some common usage scenarios for a console-based application. Next, demonstrate how to develop and run a simple console application. Finally, talk about how to generate output and accept input from a console-based application.

The final section covers the development of Windows services. Start by discussing some common scenarios for Windows services. Emphasize that Windows services are ideal for creating long-running applications that run in the background and do not have any user interface. Next, talk about the ServiceBase class and some of the common functionality provided by this class. Next, discuss how a Windows service is installed. Finally, demonstrate how you can interact with a Windows service by using the Services tool.

Key Terms

command-line parameters—The parameters sent to a program being called.

Console—Programs that are designed to be used via a text-only interface.

delegates—The Windows Forms event model uses .NET Framework delegates to bind events to their respective event handlers.

events—Actions that are generated as a result of user actions and make programs interactive.

installer—A program that installs different types of applications onto a computer.

visual inheritance—Allows you to reuse existing functionality and layout for Windows Forms.

Windows service—A long-running application that runs in the background and doesn’t require any user intervention.

Lesson 5

Understanding Desktop Applications

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You need to design a Windows service that cannot be paused. Which of the following options will help you accomplish this task?

   a. Set the CanPauseAndContinue property of the Windows service to False.

   b. Set the CanPauseAndContinue property of the Windows service to True.

   c. Set the CanStart property of the Windows service to True, but set the CanShutdown property to False.

   d. Do not override the OnPause and OnContinue methods in the Windows service.

2. You have developed a Windows service and need to install this service in order to install its functionality. Which of the following options should you choose to accomplish this task?

   a. Use the Visual Studio Server Explorer.

   b. User the Services node in the Computer Management window.

   c. Use InstallUtil.exe.

   d. Use gacutil.exe.

3. You have developed a Windows service. This service needs to run as a nonprivileged user in order to minimize any possible security risk. Which of the following accounts should you use for running this Windows service?

   a. LocalSystem

   b. NetworkService

   c. LocalService

   d. User (where the UserName property is set to a member of administrator role)

4. You are designing a Windows service application that contains only one Windows service. You would like this service to be started automatically when the computer is restarted. Which of the following classes should you use to specify this setting?

   a. ServiceBase

   b. ServiceInstaller

   c. ServiceProcessInstaller

   d. ServiceController

5. You need to change the display and behavior of a Windows Form so that the form can contain multiple child windows. What should you do?

   a. Set the IsMdiContainer property of the form to True.

   b. Set the MdiParent property for all the child windows.

   c. Set the MdiChild property of the form.

   d. Set the IsMdiChild property of the form.

6. You are developing a Windows Form that responds to mouse events. When the mouse moves, you need to invoke the method Form1_HandleMouse. Any code that you write should not affect any existing event-handling code. What statement should you use to attach the event handler to the event?

   a.

```
this.MouseDown = new MouseEventHandler
         (Form1_HandleMouse);
```

   b.

```
this.MouseMove = new MouseEventHandler
         (Form1_HandleMouse);
```

   c.

```
this.MouseDown += new MouseEventHandler
         (Form1_HandleMouse);
```

   d.

```
this.MouseMove += new MouseEventHandler
         (Form1_HandleMouse);
```

7. You are developing a Windows form with a multiple document interface (MDI). You need to write code that arranges the child windows vertically within the client region of the MDI parent form. Which of the following statements should you use?

   a.

```
LayoutMdi(MdiLayout.TileVertical);
```

   b.

```
LayoutMdi(MdiLayout.Cascade);
```

   c.

```
MdiLayout(LayoutMdi.TileVertical);
```

   d.

```
MdiLayout(LayoutMdi.Cascade);
```

8. You are developing an application that will be run from the command line. Which of the following methods would you use for output to the command line?

   a. Console.Read

   b. Console.Write

   c. File.Read

   d. File.Write

9. You want to develop an application that displays a visual surface capable of displaying a variety of controls, such as text boxes, buttons, and menus. The application should also allow multiple child windows to reside under a single parent window. Which of the following types of applications should you develop?

   a. Console-based application.

   b. Windows service application.

   c. Single document interface (SDI) application.

   d. Multiple document interface (MDI) application.

10. You are extending an existing Windows application. You would like to create a new form that derives its visual characteristics (including size, color, and some controls) from a previously created form. Which technique should you use to create the new form?

    a. Visual inheritance.

    b. Visual encapsulation.

    c. Visual abstraction.

    d. Visual polymorphism.

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. Use the **Description** property of the ServiceInstaller class to specify a brief comment that explains the purpose of the service.
2. The **Account** property of the **ServiceProcessInstaller** class indicates the account type under which a Windows service will run.
3. The **Source** property of the EventLog class is use to specify the application name to use when writing to an event log.
4. **Visual inheritance** allows you to reuse existing functionality and layout for Windows Forms.
5. **Multiple document interface (MDI)** applications are applications in which multiple child windows reside under a single parent window.
6. A(n) **Windows service** is ideal for creating long-running applications that run in the background and do not have any user interface.
7. **Console-based applications** do not have a graphical user interface and use a text-mode console window to interact with the user.
8. **Multiple document interface (MDI)** applications provide their own window management functionality, whereas **single document interface (SDI)** applications rely on the operating system for window management.
9. A delegate can be bound to any method whose signature matches that of the **delegate declaration**.
10. The **multicast delegates** can be bound to more than one method, allowing one-to-many notifications when an event is fired.

Competency Assessment

Project 5-1: Using Visual Inheritance

You need to create a Windows Form similar to the one you created in the VisualInheritance exercise. However, this time, the requirement is that the background color of this form must match the currently selected color of the user's desktop. How would you develop such a form?

1. **Open the Windows Application project named WindowsFormsDesign.**
2. **Add a new Windows Form based on the Inherited Form template. Name the inherited form BackColor.cs.**
3. **In the Inheritance Picker dialog box, select InheritedForm.**
4. **In the Designer view of the BackColor form, set the Text property to "BackColor Form."**
5. **In the Designer view, select the BackColor form. From the Properties window, set the BackColor property to Desktop. Set the ForeColor property to HighlightText.**
6. **Open Program.cs and modify the Main method as shown below to make sure that BackColor form is launched when you run the application:**

```
[STAThread]
static void Main()
{
   Application.EnableVisualStyles();
   Application
    .SetCompatibleTextRenderingDefault(false);
   Application.Run(new BackColor());
}
```

7. **Build and run the project.**

Project 5-2: Handling the MouseDown Event

You are developing a game that allows users to hit a target area on a Windows Form with their mouse. You need to develop an experimental form that displays the X and Y coordinates of the location clicked by the user in the form's title bar. How should you achieve this?

1. **Create a new Windows Forms project with the name MouseDown.**
2. **In the Properties window of the form, click the Event icon and look for an event named MouseDown. Double-click the row containing the MouseDown event. Modify the**

MouseDown event handler created by Visual Studio to the following:

```
private void Form1_MouseDown(object sender,
   MouseEventArgs e)
{
    Text = String.Format(
      "X = {0}, Y = {1}", e.X, e.Y);
}
```

3. **Build and run the project. Click on the client area of the form and notice that the text in the title bar shows the coordinates of the point at which the mouse button was pressed.**

Proficiency Assessment

Project 5-3: Working with Console Input

You are developing a program that manipulates text. You need to write a console-based application that accepts text from the user and converts the text to upper-case letters. What code do you need to write to meet this requirement?

1. **Create a new ConsoleApplication and name it ToUpper.**
2. **Add the following code to the Main method:**

```
static void Main(string[] args)
{
    int character = Console.Read();

    while (character != -1)
    {
        if (Char.IsLetter((char)character))
           Console.Write(((char)character)
               .ToString().ToUpper());
        else
            Console.Write( (char) character);

        character = Console.Read();
    }
}
```

3. **Build and run the application. Type some letters on the command line and press Enter. You should see that the letters are converted to upper case. Press Ctrl+Z (end of input) to end the program.**
4. **Copy DisplayFile.exe from the previous exercise and ToUpper.exe to the same folder. Run the following command from the command line:**

```
DisplayFile Sample.txt | ToUpper
```

Notice that the generated output is the contents of the Sample.txt file converted to uppercase letters. The '|' character works as a pipe between commands, so the console output from the DisplayFile.exe works as console input to the ToUpper.exe. By using pipes, you can chain multiple simple commands to accomplish complicated tasks.

Project 5-4: Using the Net Utility (net.exe)

The net.exe command line utility comes installed with Windows. This utility allows you to perform various networking commands, including control of Windows services. Say you want to use net.exe to work with the previously created FirstService Windows service. What steps must you take in order to pause, stop, and start a Windows service using the net.exe utility?

1. **Open the command prompt. To enumerate a list of all installed services, issue the following command:**

```
net start
```

2. **To start the FirstService Windows service, give the following command:**

```
net start FirstService
```

3. **To pause FirstService, give the following command:**

```
net pause FirstService
```

4. **To resume the paused FirstService, give the following command:**

```
net continue FirstService
```

5. **To stop FirstService, give the following command:**

```
net stop FirstService
```

6. **Check the Application event log to see the messages generated by FirstService.**

Lesson 6

Understanding Databases

Learning Objectives

Students will learn about:

- Relational Database Management Systems
- Database Query Methods
- Database Connection Methods

ODN Skills

- Understand relational database management systems. 6.1
- Understand database query methods. 6.2
- Understand database connection methods. 6.3

Lesson Summary — Lecture Notes

Lesson 6 introduces students to basic database concepts. Start the discussion by emphasizing that data is at the heart of most business applications. To be effective at work, the students will need to know how to connect to various data sources and how to retrieve and update data in these types of data sources.

The next topic discusses the database design process. This lesson uses entity relationship (ER) diagram as a modeling technique. You should demonstrate how ER diagrams can help the students in determine what data needs to be stored in a database.

The next topic is about the process of data normalization. This lesson discusses three normalization forms:

1. The first normal form (1NF).
2. The second normal form (2NF).
3. The third normal form (3NF).

Discuss the requirements for each form and help students understand how the normalization process can ensure the data integrity by eliminating insert, delete, and update anomalies.

The next topic covers structured query languages. In this section, help students understand how to write the SELECT, INSERT, UPDATE, and DELETE statements. Discuss both ad-hoc queries and stored procedures. Ensure that students understand when to use a stored procedure and when to use ad-hoc queries.

The final section discusses how to work with flat files, XML data, and datasets. Demonstrate how to read from and write to text files, binary files, and XML files. Make

sure students understand when to use each type of file for storing data. Also make sure they understand dataset is an in-memory representation of a database. Emphasize that datasets are helpful for developing applications that need to work while disconnected from the network.

Key Terms

entity relationship diagram—Used to model entities, their attributes, and the relationships among entities. They can help you determine what data needs to be stored in a database.

first normal form (1NF)—A step in the database normalization that ensures that none of the columns in the table have multiple values in the same row of data.

flat files—A database table that is stored inside a stand-alone disk file.

normalization—A data normalization process ensures that a database design is free of any problems that could lead to loss of data integrity.

parameterized stored procedure—AllowS you to pass runtime arguments to the SQL Server.

second normal form (2NF)—A step in the database normalization that requires that all non-key columns are functionally dependent on the entire primary key.

stored procedure—A collection of SQL statements and programming logic that are stored on the database server as named objects.

Structured Query Language (SQL)—The language used by most database systems to manage the information in their databases.

third normal form (3NF)—A step in the database normalization that requires that there is no functional dependency among the non-key attributes.

XML (eXtensible Markup Language)—A text-based format for representing structured data.

Lesson 6

Understanding Databases

Knowledge Assessment

Multiple Choice

Circle the letter or letters that correspond to the best answer or answers.

1. Your application needs to store the product image out to a disk file. You'd like to minimize the size of this disk file. Which of the following objects should you use to write the file?

   a. FileStream

   b. StreamWriter

   c. BinaryWriter

   d. XmlWriter

2. Your C# program needs to return the total number of customers in a database. The program will be used several times a day. What is the fastest way to return this information from your program?

   a. Write a SQL query and use the SqlCommand.ExecuteScalar method to execute the query.

   b. Create a stored procedure to return the total number of customers, then use the SqlCommand.ExecuteScalar method to execute the stored procedure.

   c. Write a SQL query and use the SqlDataAdapter.Fill method to execute the query.

   d. Create a stored procedure to return the total number of customers, then use the SqlDataAdapter.Fill method to execute the stored procedure.

3. You need to modify the records in a Products table by marking certain products as Discontinued. However, you need to do this only when the UnitsInStock and UnitsOnOrder are both zero. Which of the following SQL statements should you use?

   a. INSERT

   b. SELECT

   c. UPDATE

   d. DELETE

4. You need to update the Region fields for customers in Japan. You write the following SQL UPDATE statement:

```
UPDATE Customers
SET Region = 'EastAsia'
```

   You test the query on a test database and find that more records were affected than you expected. You need to correct the SQL statement. What should you do?

   a. Add a WHERE clause to the UPDATE statement.

b. Add an additional SET clause to the UPDATE statement.

c. Add a GROUP BY clause to the UPDATE statement.

d. Add a HAVING clause to the UPDATE statement.

5. You are developing an application that needs to retrieve a list of customers from a SQL Server database. The application should move through the list sequentially once, processing each customer's record. Which of the following classes should you use to hold the customer list in order to achieve maximum performance?

   a. DataSet

   b. DataTable

   c. DataView

   d. SqlDataReader

6. The application you are developing needs to read data from a flat file that include items such as a five-digit integer key, followed by a 20-character customer name, followed by two date and time fields. Which of the following classes should you use?

   a. FileStream

   b. StreamReader

   c. BinaryReader

   d. DataReader

7. You are developing an application that will need to copy data from a SQL Server view to a DataSet. You name the DataSet object dsData. Which of the following methods should you use to copy the data?

   a. Fill

   b. InsertCommand

   c. SelectCommand

   d. Update

8. You are developing an application that manages customers and their orders. Which of the following situations is not a good candidate for implementation with stored procedures in your application?

   a. Retrieving the list of all customers in the database.

   b. Retrieving the list of all orders for particular customers.

   c. Inserting a new order into the Orders table.

   d. Ad hoc querying by the database administrator.

9. Your application connects to a SQL Server database that contains a table called Employees with the following columns:

   EmployeeID (int, identity)

   EmployeeType (char(1))

   EmployeeDate (datetime)

   You need to write a query that deletes all rows from the table where the EmployeeType value is either C or T. You do not want to delete any other rows. Which statement should you use?

   a.

```
DELETE FROM Employees
```

```
WHERE EmployeeType LIKE '[CT]'
```

b.

```
DELETE FROM Employees
WHERE EmployeeType LIKE '[C-T]'
```

c.

```
DELETE FROM Employees
WHERE EmployeeType LIKE 'C' OR 'T'
```

d.

```
DELETE * FROM Employees
WHERE EmployeeType IN ('C', 'T')
```

10. Your application includes a SqlDataAdapter object named sqlDataAdapter that connects to the Employees table. Based on this SQLDataAdapter, your application also includes a DataSet object dsEmployees. What line of code should you use to load the data from the database into the DataSet object?

a.

```
dsEmployees = sqlDataAdapter.Fill("Employees");
```

b.

```
sqlDataAdapter.Fill("dsEmployees", "Employees");
```

c.

```
sqlDataAdapter.Fill(dsEmployees);
```

d.

```
sqlDataAdapter.Fill(dsEmployees, "Employees");
```

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. In order for a table to be in the **first normal form (1NF)**, none of the columns should have multiple values in the same row of data.
2. The **second normal form (1NF)** requires that all non-key columns are functionally dependent on the entire primary key.
3. The **third normal form (1NF)** requires that there is no functional dependency among the non-key attributes.
4. The basic building blocks for an entity relationship diagram are **Entity**, **Attribute**, and **Relationship**.
5. The **WHERE** clause in a SELECT statement evaluates each row for a condition and decides whether to include it in the result set.
6. The object used with the *using* statement must implement the **IDisposable** interface.
7. T-SQL's **CREATE PROCEDURE** statement can be used to create a stored procedure.
8. In the process of **normalization**, you apply a set of rules to ensure that your database design helps with data integrity and ease of maintenance in the future.

9. You find classes for working with streams and backing stores in the **System.IO** namespace.
10. The **XML** format is a hierarchical data-representation format.

Competency Assessment

Project 6-1: Creating an Entity-Relationship Diagram

A company has a number of employees, and each employee may be assigned to one or more projects. In addition, each project can have one or more employees working on it. Draw an entity-relationship diagram for this situation.

1. **As the first step, you must identify the entities. The two entities of interest based on the given description are Employee and Project.**
2. **Next, you need to identify the relationships between entities. Based on the description, Employees have an "assigned to" relationship with projects. The "assigned to" relationship is many-to-many because many employees can work on a single project and one employee can work on many projects.**
3. **Finally, you can draw the entity-relationship diagram based on the above analysis.**

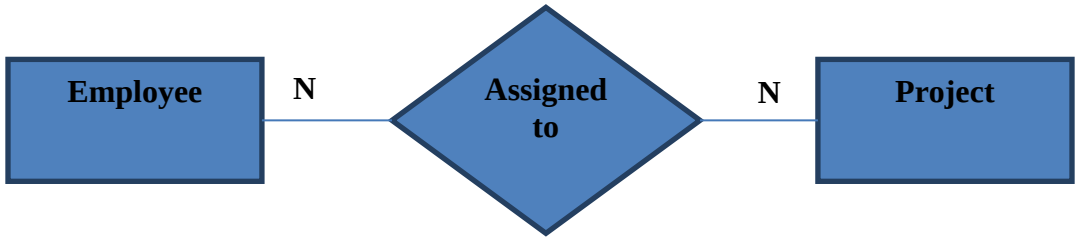


Figure 6-1
An entity-relationship diagram.

Project 6-2: Creating a Stored Procedure

You often need to generate a list of customers from a given country. Therefore, you decide to create a stored procedure that accepts the name of country as a parameter and returns all the customers from that country. How would you go about doing this?

1. **Open Server Explorer and select the Northwind database. Right-click the Stored Procedure node and select the Add New Stored Procedure option.**
2. **In the stored procedure designer, replace the boilerplate text with the following code:**

```
CREATE PROCEDURE GetCustomersFromCountry
(
        @country nvarchar(15)
```

```
)
AS
        SELECT * FROM Customers
        Where Country = @country
RETURN
```

3. **Save the stored procedure. The stored procedure is now added to the database.**
4. **To execute the stored procedure, right-click the procedure in the Solution Explorer and select Execute. In the Run Stored Procedure dialog box, type the name of the desired country and click OK. The results should be displayed in the Output window.**

Competency Assessment

Project 6-3: Normalizing Tables

You are converting an entity-relationship diagram into tables. You come up with the following table design:

Books

BookId	BookName	CategoryId	CategoryName
1	Cooking Light	1001	Cooking
2	Prophecy	1002	Mystery & Thriller
3	Shift	1003	Business
4	The Confession	1002	Mystery & Thriller

You need to apply normalization rules to ensure data integrity. How would you ensure that the Books table is in the third normal form?

1. **First, apply the first normal form. Each column in the Books table stores a single value and there are no repeating groups. Therefore, the Books table conforms to the first normal form.**
2. **Next, apply the second normal form. If a table satisfies 1NF and has only a single column in its primary key, then the table also conforms to 2NF. Thus, the Books table also conforms to the second normal form.**
3. **Next, apply the third normal form. The 3NF requires that there is no functional dependency between the non-key attributes. In the case of Books table, CategoryName is functionally dependent on CategoryId, and both CategoryName and CategoryId are non-key attributes. As a result, the Books table does not conform to 3NF. To fix this issue, the**

violating data needs to be stored in a different table. The new design should consist of the following two tables:

Books

BookId	BookName	CategoryId
1	Cooking Light	1001
2	Prophecy	1002
3	Shift	1003
4	The Confession	1002

Categories

CategoryId	CategoryName
1001	Cooking
1002	Mystery & Thriller
1003	Business

Finally, you can apply all the normalization rules again on both the Books and Categories tables to ensure that they satisfy the 3NF. In this case, they do.

Project 6-4: Creating and Handling Events

You are working on an application that requires you to save customer information from the Customers table of the Northwind database into an XML file. This XML file will be used for various data integration tasks. You need to make sure that the root node of the XML is called Customers. The root node will then have a Customer node for each customer in the Customers table. How should you accomplish this task?

1. **Create a new Visual C# Console application named DataSetXml.**
2. **Modify the code in the Program class to the following. Be sure to change the connection string to match the local path of the database file on your computer:**

```
static void Main(string[] args)
{
    SaveDataSetAsXml();
}

static private void SaveDataSetAsXml()
{
```

```
        try
        {
            // Change the connection string
            // to match with your system.
            string connectionString =
            @"Data Source=.\SQLEXPRESS;" +
            @"AttachDbFilename=" +
            @"c:\SqlSampleDB\NORTHWND.MDF;" +
            @"Integrated Security=True;" +
            @"Connect Timeout=30;User Instance=True";

            SqlConnection connection =
              new SqlConnection(connectionString);
            string commandText = "SELECT CustomerId, "
                + "CompanyName, ContactName, "
                + "Phone FROM Customers";

            DataSet ds = new DataSet("Customers");
            SqlDataAdapter sda = new SqlDataAdapter(
              commandText, connection);
            sda.Fill(ds, "Customer");
            ds.WriteXml("Customers.xml",
              XmlWriteMode.IgnoreSchema);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
```

3. **Add the following using directives to your program:**

```
using System.Data.SqlClient;
using System.Data;
```

4. **Build and run the project. Open the output XML file. You should see XML in the following format:**

```
<?xml version="1.0" standalone="yes"?>
<Customers>
  <Customer>
    <CustomerId>ALFKI</CustomerId>
    <CompanyName>Alfreds Futterkiste</CompanyName>
    <ContactName>Maria Anders</ContactName>
    <Phone>030-0074321</Phone>
  </Customer>
  <Customer>
    <CustomerId>ANATR</CustomerId>
```

```
    <CompanyName>Ana Trujillo Emparedados y
   helados</CompanyName>
    <ContactName>Ana Trujillo</ContactName>
    <Phone>(5) 555-4729</Phone>
  </Customer>
  <!-- more customers will go here -->
<Customers>
```