Lesson 1

Understanding Object Oriented Programming

Learning Objectives

Students will learn about:

- Basics of object oriented programming
- Encapsulation
- Inheritance
- Polymorphism
- Interfaces
- Namespaces
- Creating class libraries

ODN Skills

Understand object oriented concepts in the .NET Framework	2.2
Understand .NET class hierarchies	2.1
Understand .NET namespaces	2.3
Understand and create class libraries	2.4

Lesson Summary — Lecture Notes

Lesson 1 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.

In the next section, discuss the concept of encapsulation. Here again, 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, 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 next 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.

Next, talk about interfaces. Make sure students can distinguish when to use inheritance and when to use an interface. Also discuss a common interface such as IComparable, which is already defined in the System namespace.

The next section discusses how to organize code and creating globally unique class names by using namespaces. Make sure students understand how to create a hierarchy of namespaces for organizing a large number of classes. You can present examples from the .NET class library to demonstrate how the namespace organization works. Discuss a sample of commonly used namespaces in the .NET Framework.

In the final topic of this lesson, discuss the advantages of packaging code into reusable class libraries. Reiterate the concept of namespace and assemblies and how they can be helpful tools in creating a good class library.

Key Terms

abstract class - An abstract class provides a common definition of the base class that can be shared by multiple derived classes.

access modifier - An access modifier specifies what region of the code will have access to a field or method. For example, a public access modifier will not limit access but the access modifier private will limit access to just the class in which the field is defined.

accessors - The accessors of a property contain the code associated with reading for writing the property value. A property can have two accessors. The `get` accessor is used to return the property value. The `get` accessor is used to assign a new value to the property.

assembly - An assembly is a unit of executable code that can be independently versioned and installed.

auto implemented properties - Auto implemented properties enable you to quickly specify a property of a class without having to write code to `get` and `set` the property.

base class - Inheritance enables you to create new classes that reuse, extend, and modify the functionality defined in existing classes. The class whose functionality is inherited is a base class.

class - A class is the template from which individual objects are created.

class library - A class library is a collection of functionality defined in terms of classes, interfaces and other types that can be reused to create applications, components, and controls.

constructor - A constructor is a block of code used to initialize the data members of the object.

derived class - Inheritance enables you to create new classes that reuse, extend, and modify the functionality defined in existing classes. The class that inherits functionality is a derived class.

encapsulation - Encapsulation is an information-hiding mechanism that makes code easy to maintain and understand.

events - Events provide communication between the objects.

inheritance - Inheritance is an object-oriented programming technique that allows you to create new classes that reuse and extend functionality of existing classes.

interface - An interface is used to establish contracts through which the objects interact with each other without knowing the implementation details.

method - A method is a block of code containing a series of statements that can be called from elsewhere in a program. Methods specify an object's behavior.

namespace - Namespace is a language element that allows you to organize code and create globally unique class names.

new – The new keyword is used to create a new instance of a class. When used in a class definition, the new keyword creates a new member with the same name in the derived class and hides the base class implementation.

object oriented programming - Object oriented programming is a programming technique that makes use of objects.

objects - Objects are self-contained data structures that consist of properties, methods and events.

override - The override keyword replaces a base class member in a derived class.

polymorphism - Polymorphism refers to the ability of the derived classes to share common functionality with the base classes but still define their own unique behavior.

property - A property is a block of code that allows you to access a class data field in a safe and flexible way.

sealed class - A sealed class provides complete definition for a class but cannot be used as a base class.

static members - Static members belong to the class itself rather than individual objects.

this - The this keyword can be used to access members from within constructors, instance methods, and the accessors of instance properties.

Lesson 1

Understanding Object Oriented Programming

Knowledge Assessment

Fill in the Blank

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

1. A __class________ is a template for creating of an object.
2. A class that does not provide complete implementation must be declared with the ___`abstract`_______keyword.
3. The 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 if 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 the class.
8. You can use the _`static`_________ keyword to declare a member, which belongs to the class itself rather than to a specific object.
9. A __class library__ is a collection of functionality defined in terms of classes, interfaces and other types that can be reused to create applications, components, and controls.
10. If a class by the same name exists in two different namespaces, use a __using directive__ to resolve the ambiguous references.

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 rendering of additional image formats. You want the `Render` method of the derived class to be executed even if the 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 advance mathematical functionality. You do not want the functionality of this class to be inherited into a derived class. 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 query will likely be different. Not all the classes have an "is-a" relationship with each other. How should you support this functionality?

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

    b. Have all the classes inherit from an abstract base class and override 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 be used 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 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 chose?

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` 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 line of code should you use inside the `Process` method?

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. Which of the following C# features organizes code and creates globally unique types?

a. assembly

b. namespace

c. class

d. data type

Competency Assessment

Project 1-1: Creating Properties

You need to create a class named `Product` that represents a product. The class has a single property named `Name`. The users of the `Product` class should be able to get as well as 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. The user of the `Product` class should not be able to access any other data members of the `Product` class. How should you achieve this?

1. Create a new project based on the Class Library template. Name the project as ProductLibrary.

2. Rename the file Class1.cs to Product.cs

3. Replace the code in the Product.cs file with the following code:

```
using System;
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 Project to compile the code.

Project 1-2: Creating a Class Library

You are developing components of a complex accounting system. You need to develop two classes: `Payable` and `Receivable`. Both classes are part of the `CompanyA.Accounting` namespace. However, it should be possible to deploy each class separately. In other words, if you need only the code for the `Payable` class, you should not be required to package the code for the `Receivable` class as well. How should you achieve this?

1. Create a new c# project based on the Class Library template. Name the project as AccountsPayable.
2. Rename Class1.cs file to Payable.cs. Replace the code in the file with the following:

```
namespace CompanyA.Accounting
{
    public class Payable
    {
        // code for Payable class
    }
}
```

3. Add a new c# project based on the Class Library template. Name the project as AccountsReceivable.
4. Rename Class1.cs file to Receivable.cs. Replace the code in the file with the following:

```
namespace CompanyA.Accounting
{
    public class Receivable
    {
        // code for Receivable class
    }
}
```

5. Build the `AccountsPayable` project.
6. Build the `AccountsReceivable` project.

At this point you have two classes, `Payable` and `Receivable`, which are part of the CompanyA.Accounting namespace. Each file belongs to a different assembly; the `CompanyA.Accounting.Payable` class belongs to the `AccountsPayable` assembly and the `CompanyA.Accounting.Receivable` class belongs to the `AccountsReceivable` assembly. If your application needs to only work with the accounts payable, you can just add the reference to just the `AccountsPayable` assembly.

Proficiency Assessment

Project 1-3: 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 in order to meet this requirement?

1. Open the ProductLibrary project created in Project 1-1.
2. Create a new c# class named ProductEx.cs
3. Replace the code in the `ProductEx.cs` class with the following code:

```
using System;
```

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

4. Click **Build** > **Build Project** to compile the code.

Project 1-4: Working with Virtual Methods

You are writing code for an `Order` class. The `Order` class contains a method named `CalculateDiscount` that returns a decimal type. The users of the `Order` class should be able to extend the behavior of the `Order` class and when they do so, the users should be able to override the behavior of the `CalculateDiscount` to change how the discount is calculated. What code should you write for the `Order` class in order to meet this requirement?

1. Create a new project based on the Class Library template. Name the project as OrderProcessing.
2. Rename Class1.cs to Order.cs. Replace the code of the file with the following:

```
namespace OrderProcessing
{
   public class Order
   {
      public virtual decimal CalculateDiscount()
      {
         // code to calculate discount;
         return 0;
      }
   }
}
```

3. Click **Build** > **Build Project** to compile the code.

Lesson 2

Understanding Data Types and Collections

Learning Objectives

Students will learn about:

- Using different data types in the .NET Framework
- Arrays
- Generics
- Collections
- Generic Collections

ODN Skills

Understand and use different data types in the .NET Framework	2.5
Understand generics	2.6

Lesson Summary — Lecture Notes

Lesson 2 introduces students to data types and collections. A good way to start the discussion is to have students think about the different types of data that they deal with on a day-to-day basis. Then extend the discussion to the data as they relate to programming.

Start the discussion by introducing the intrinsic data types. Students should be able to choose the right data type based on the kind of data that needs to be stored. Students also need to know the C# data types as well as their equivalent .NET Framework type.

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 data type conversion and casting. Talk about the data type conversions that are permissible and the conversions that need an explicit cast operation. Next, you talk about the concept of boxing and unboxing. Emphasize that the C# programming language has a unified type system where value of any type can be treated as an object. Students should be aware that boxing and unboxing are computationally expensive processes.

The next section talks about arrays and collections. Emphasize that these data structures are used to store and manipulate a collection of items. The section covers both the single dimensional and multi-dimensional arrays. Emphasize that arrays have a homogeneous data type and a fixed size. Present `ArrayList` as a collection type that overcomes some of the arrays requirements but causes issues with type-safety and performance.

The next topic is generics. Present generics as a solution to the problem of performance and type safety associated with the collections such as `ArrayList`. Introduce students to the generic vocabulary such as constraints, verifiability and variance.

The rest of the lesson discusses generic collections. Spend time in discussing some common generic collections that are part of the .NET class libraries. Make sure students understand the use case of each generic collection. Also demonstrate how to perform basic operations such as adding an item and removing an item from a generic collection.

Key Terms

array - An array is a collection of items stored in a contiguous memory location. Each item of an array can be accessed by using the array's index, which is unique for each item.

boxing - Boxing is the process of converting a value type to the reference type.

collection classes – The collection classes provide data structures to store and manipulate a collection of items. An item in the collection can be anything such as a number, a string, a Customer object, and so on.

data type - The data types specify rules for converting values in one type to another type. The conversion can be performed implicitly by the compiler or explicitly by using the cast operator.

generic collection – generic collections provide a type-safe alternative to the non-generic collection classes. By using the generic collection, you can avoid the issues of type safety and the boxing and unboxing performance issue.

intrinsic data types - Intrinsic data types are the primitive data types for which the support is directly built into the programming language.

reference types – The reference types store a reference to a memory location. The referenced memory location stores the actual data value.

strongly typed language – In a strongly typed language, the data types for variables, constants, literal values, method return values, and parameters must be known at the compile time.

two-dimensional array – A two-dimensional array can be thought of as a table in which each cell is an array element and can be addressed by using the row number and the column number to which it belongs.

value types - The value types directly store a value of a specific data type in a memory location.

unboxing - Unboxing is the process of converting a reference type to the value type.

Lesson 2

Understanding Data Types and Collections

Knowledge Assessment

Fill in the Blank

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

1. __Intrinsic data types__ are the primitive data types for which the support is directly built into the programming language.
2. To access the first element of an array, use an index of __0__.
3. In the C# programming language, the data types for variables, constants, method return values, and parameters must be known at __compile time__.
4. The reference types store only a __reference__ to a memory location that contains the actual data.
5. __Boxing__ is the process of converting a value type to the reference type. __Unboxing__ is the process of converting an reference type to the value type.
6. An array is a collection of items, of the same data type, stored in a __contiguous__ memory location and addressed by using one or more __indices__.
7. The memory allocated in the __heap__ is automatically reclaimed by the garbage collector when the objects are not in use anymore.
8. __`List<T>`__ is the generic class that corresponds to `ArrayList`.
9. __Type safety__ ensures that you can only assign a value of the correct data type to a generic collection.
10. The generic collection classes are defined as part of the System.Collections.Generic namespace.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You need to store values ranging from 0 to 255. You need to also 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`

2. You write the following code snippet:

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

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

   a. 1

   b. 2

   c. 3

   d. 4

3. You need to create a variable that stores multiple unicode characters. Which data type should you use to define such a variable?

   a. `byte`

   b. `sbyte`

   c. `char`

   d. `string`

4. You need to create a structured data type named `Point`. `Point` should have two fields (`X` and `Y`), each of the `float` data type. Programs that use the `Point` data type should be able to directly reference the fields `X` and `Y`. You need to make sure that `Point` is a value data type so that the value of the variable is directly stored in its assigned memory location. How should you define such a data type?

   a.

```
public class Point
{
   private int X, Y;
}
```

   b.

```
public class Point
{
   public int X, Y;
}
```

   c.

```
public struct Point
{
   private int X, Y;
}
```

   d.

```
public struct Point
{
   public int X, Y;
}
```

5. Which of the following statements, when executed, will throw an exception of the type `InvalidCastException`?

   a.

```
int number1 = 12;
double number2 = number1;
```

b.

```
double number2 = 3.141;
int number1 = number1;
```

c.

```
int i = 10;
object o = i;
```

d.

```
object o = 10;
int i = (int) o;
```

6. You write the following code in your program:

```
double p = 3.141;
object o = p;
int i =  (int) o;
```

Not all data type conversion is permissible. When you run this code, which of the following exceptions should you expect to receive?

a. `System.InvalidCastException`

b. `System.InvalidOperationException`

c. `System.InvalidDataException`

d. `System.FormatException`

7. You are using a `HashTable` collection in your program. You want to modify your program to take advantage of type safety and increased performance offered by the generic types. Which of the following generic types should you use to replace `HashTable` with in your program?

a. `Collection<T>`

b. `Dictionary<TKey, TValue>`

c. `List<T>`

d. `SortedList<TKey, TValue>`

8. You are working with the `List<T>` collection in a program that you are developing. You write the following code:

```
List<int> numbers = new List<int>();
numbers.AddRange(new[] { 2, 6, 8 });
```

You need to insert the number 4 between the number 2 and 6. What code should you write to accomplish this?

a. `numbers.Insert(4, 1);`

b. `numbers.Insert(1, 4);`

c. `numbers.Insert(2, 4);`

d. `numbers.Insert(4, 2);`

9. You are developing a C# program. The program declares a variable that represents the first-in, first-out collection of values of the double data type. You need to make sure that you are not able to store any other data type in

the collection. What data type should you use to declare the variable in your program?

a. `Queue`

b. `Queue<double>`

c. `Stack`

d. `Stack<double>`

10. You are developing a C# program. The program declares a variable that represents a strongly typed collection of values of the double data type. The values of the collection can only be manipulated in the order of last-in, first-out. Which data type should you use to declare the variable in your program?

a. `Queue`

b. `Queue<double>`

c. `Stack`

d. `Stack<double>`

Competency Assessment

Project 2-1: Using Two-dimensional Arrays

You are writing a program that uses a two-dimensional array. The array has 4 rows and 5 columns. You need to print the largest element in each row of the array.

1. Create a new Visual C# Console Application project named TwoDimensionalArray.
2. Replace the code in the Program.cs file with the following code:

```
using System;

namespace TwoDimensionalArray
{
    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. Click **Debug** > **Start Debugging** (or press **F5**) to run the project. The following output is displayed on 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 2-2: Using a Linked List

You are writing a program that stores the list of product names in a linked list. The user will enter a product name and your program needs to check if 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 GenericLinkedList.

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

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

namespace GenericLinkedList
```

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

            Console.Write(
              "Enter product to find (type <end> to
   quit): ");
            string searchText = Console.ReadLine();

            while (searchText != "<end>")
            {
                if (list.Contains(searchText))
                {
                    Console.WriteLine(
                        "The search text was found\n");
                }
                else
                {
                    Console.WriteLine(
                     "The search text was NOT found\n");
                }

                Console.Write(
                  "Enter product to find (type <end> to
   quit): ");
                searchText = Console.ReadLine();
            }
        }
    }
}
```

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

Proficiency Assessment

Project 2-3: Using a Generic Queue Collection

You are writing a program that uses two queues. The data in each queue is already in the ascending order. You need to process the contents of both the queues in such a way that the output is printed on the screen in sorted order. You should be using to the generic version of the Queue class to benefit from type safety and faster performance. How should you write a program to address this scenario?

1. Create a new Visual C# Console Application project named GenericQueue.
2. Replace the code for the Program.cs file with the following code:

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

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

            Queue<int> second = new Queue<int>();
            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<int> first,
            Queue<int> 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 (first.Peek() >= second.Peek())
                    {
                        Console.WriteLine(
                            second.Dequeue());
                    }
                    else
                    {
                        Console.WriteLine(first.Dequeue());
                    }
                }
            }
        }
    }
```

1. 3. Click **Debug** > **Start Debugging** (or press **F5**) to run the project. You'll notice that the numbers are displayed in ascending order in the console window.

Project 2-4: Using a Generic Stack Collection

You are writing a program that uses two stacks. The data in each stack is already in the descending order. You need to process the contents of both the stacks in such a way that the output is printed on the screen in the ascending order. You should be using to the generic version of the `Stack` class to benefit from type safety and faster performance. How should you write a program to address this scenario?

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

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

namespace GenericStack
{
```

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

        Stack<int> second = new Stack<int>();
        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<int> first,
        Stack<int> 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 (first.Peek() >= second.Peek())
            {
                Console.WriteLine(
                    second.Pop());
            }
```

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

3. Click **Debug > Start Debugging** (or press **F5**) to run the project. You'll notice that the numbers are displayed in ascending order in the console window.

Lesson 3

Understanding Events and Exceptions

Learning Objectives

Students will learn about:

- Events and event handling
- Structured exception handling
- Working with application settings

ODN Skills

Understand events and event handling in the .NET Framework	1.2
Understand structured exception handling in the .NET Framework	1.3
Understand basic application settings	1.1

Lesson Summary — Lecture Notes

Lesson 3 covers three different .NET Framework programming topics. The first section of the lesson covers the topic of event handling. Event handling is a common programming pattern across the .NET Framework applications. Begin the lecture by describing common event handling scenarios. Follow this discussion by showing the students how to define an event. Next, show students how to raise and subscribe to an event. Make sure students understand the difference between a `delegate` and an `event`. It is also important to emphasize on the special `+=` syntax for attaching event handlers.
The next section covers the topic of exception handling. You'll first introduce the students to the `try-catch` construct followed by the `try-catch-finally` and `try-finally` constructs. Spend some time discussing common exception classes in the .NET Framework. Emphasize that application exceptions are represented by using an object of the `System.Exception` or one of its derived classes.
The final section discusses how to manage application settings for a Windows-based application or a Web-based application. Make sure students know the difference between the `ConfigurationManager` and the `WebConfigurationManager` classes and know when to use each of them. The application settings are stored in an XML file. So, you may also need to spend some time making sure that students understand the XML format.

Key Terms

app.config - App.config is an XML-based configuration file that is used to change an application's behavior at runtime.

application settings - Application settings allow the applications to store custom application-specific data. The settings data is stored as XML in a disk file. Application settings allow the programs to change certain settings at runtime without the need to modify the program’s source code.

delegate - A delegate is special type that can hold a reference to a method. Delegates are commonly used to provide event-handling functionality.

event - An event enables an object to notify other objects or classes when something of interest occurs.

exception - An exception is an error condition that occurs during the execution of a .NET program.

web.config - Web.config is the configuration file for any ASP.NET Web site or Web service. The web.config file is used to store the settings that you might need to modify after an application has been built.

Lesson 3

Understanding Events and Exceptions

Knowledge Assessment

Fill in the Blank

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

1. The __`finally`__ block is often used in association with the `try` block to write cleanup code.
2. To handle exceptions, place the code that throws exceptions inside a __`try`__ block and place the code that handles the exceptions inside __`catch`__ blocks.
3. When working with events, the class that sends the notification is called as a __publisher__ of the event and the class that receives the notification is called the __subscriber__ of the event.
4. When passing event-related data from event publisher to event subscribers, use a class that is derived from the __`EventArgs`__ class.
5. In the .NET Framework, an exception is represented by using an object of the __`System.Exception`__ class or one of its derived classes.
6. When handling exceptions, the `catch` block should be written in order of the more __specific__ exceptions to the more __general__ exceptions.
7. The applications are stored on the disk in __XML__ format.
8. A __delegate__ is a type that references a method.
9. A delegate can be bound to any method whose signature matches that of the __ delegate declaration __.
10. The __Web.config__ file allows the Web applications to change certain settings at runtime without the need to modify the application's source code.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You need to write code that closes a connection to a database. You need to make sure that 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

2. You are writing code to handle events in your program. You defined a delegate named `RectangleHandler` like this:

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

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

```
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 with the signature of the `RectangleHandler` method. Any code that you write should not affect any existing event handling code. Which of the following code examples should you write?

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

b. `handler = DisplayArea;`

c. `handler += DisplayArea;`

d. `handler -= DisplayArea;`

3. 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. Which of the following statements should you use to attach the event handler with 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);
```

4. What will be the output of the following code?

```
int num = 5;
int den = 0;
try
{
    Console.WriteLine("Performing Division");
    int res = num/den;
    Console.WriteLine("After Division");
}
```

```
catch(SystemException se)
{
    Console.WriteLine("In Catch Block");
}
Console.WriteLine("After Catch Block");
```

a.

```
Performing Division
After Catch Block
```

b.

```
Performing Division
After Division
In Catch Block
After Catch Block
```

c.

```
Performing Division
In Catch Block
After Catch Block
```

d.

```
Performing Division
In Catch Block
After Division
After Catch Block
```

5. You are developing a Windows Forms application using the .NET Framework. Your application needs to query the name and address of the user. You have placed two TextBox controls named `txtName` and `txtAddress`. When the user presses a key on TextBox controls, or when the user changes the text in the TextBox controls, you want to run code to ensure that the user does not enters a numeric value.

   You are looking at responding to the `TextChanged` and `KeyPress` events. The event handler for `TextChanged` receives an argument of the `EventArgs` type while the `KeyPress` receives an argument of the `KeyPressEventArgs` type

   You want to write minimum code. Which of the following best describes how to structure your code to fulfill this requirement?

   a. Write four separate event handlers, one each for the `TextChanged` event of `txtName`, the `KeyPress` event of `txtName`, the `TextChanged` event of `txtAddress`, and the `KeyPress` event of `txtAddress`.

   b. Write two event handlers. The first handles both `TextChanged` events and the second handles both `KeyPress` events.

   c. Write two event handlers. The first handles the `TextChanged` and `KeyPress` events for `txtName` and the second handles the `TextChanged` and `KeyPress` events for `txtAddress`.

   d. Write a single event handler to handle the `TextChanged` and `KeyPress` events of both controls.

6. You develop a Windows form application using C#. The name of the application's executable file is SalesAnalysis.exe. For easy configuration of the application, you decide to deploy the application with an application configuration file. At runtime, the Windows form application must be able to read the settings stored in the configuration file. Which of the following actions should you take? (Select all that apply)

   a. Name the configuration file as SalesAnalysis.config

   b. Name the configuration file as SalesAnalysis.exe.config

   c. Deploy the configuration file to the same directory as the executable file.

   d. Deploy the configuration file to a child folder named config.

7. You have developed an ASP.NET application. The application needs to retrieve orders from a warehouse. For the sake of flexibility, you store the warehouse name in the `<appSettings>` element in the web.config, application's configuration file. After you deploy the application to the production Web server, the administrator will be required to modify the configuration file to change the warehouse. You want the new settings to be applied as soon as possible. What action should you take to ensure that the application reads the new connection string from the configuration file?

   a. Nothing. As soon as the administrator modifies and saves the configuration file, the application will automatically use the new warehouse name.

   b. Recompile the application

   c. Restart the ASP.NET process

   d. Close the application and restart the Web server

8. You are developing a Web application that processes orders. You need to store certain application parameters that can be configured without recompiling the application. Which of the following should you do? (Select all that apply)

   a. Store the configuration in app.config file.

   b. Store the configuration in web.config file.

   c. Use the `ConfigurationManager` class to access the settings from the configuration file.

   d. Use the `WebConfigurationManager` class to access the settings from the configuration file.

9. What will be the output of the following code?

```
int num = 5;
int den = 0;
try
{
    Console.WriteLine("First");
    int res = num/den;
    Console.WriteLine("Second");
}
```

```
catch(DivideByZeroException ex)
{
    Console.WriteLine("Third");
}
finally
{
   Console.WriteLine("Fourth");
}
Console.WriteLine("Fifth");
```

a.

```
First
Second
Third
Fourth
Fifth
```

b.

```
First
Third
Fourth
Fifth
```

c.

```
First
Second
Third
Fourth
```

d.

```
First
Third
Fifth
```

10. You are developing a console application that reads data from the file. Your application needs to handle all exceptions, including when the file is not found. You write the following code (line numbers are for reference only):

```
01: StreamReader sr = null;
02: try
03: {
04:     sr = File.OpenText(@"c:\data.txt");
05:     Console.WriteLine(sr.ReadToEnd());
06: }
07: catch (FileNotFoundException fnfe)
08: {
```

```
09:     Console.WriteLine(fnfe.Message);
10: }
11: catch (Exception ex)
12: {
13:     Console.WriteLine(ex.Message);
14: }
```

You need to write code to close the `StreamReader` object. You need to make sure that `StreamReader` object is always closed, whether an exception was thrown or not. What should you do?

a. Insert the following code after line 05:

```
sr.Close();
```

b. Insert the following code after line 09:

```
sr.Close();
```

c. Insert the following code after line 13:

```
sr.Close();
```

d. Insert the following code after line 14:

```
finally
{
    sr.Close();
}
```

Competency Assessment

Project 3-1: Accessing Application Settings for Windows Forms Application

You are developing a Windows Forms application. You should provide an application-level setting that allows the users to configure the background color of the application. The changes should not require modification to the source code. How should you achieve this?

1. Create a new Windows Forms project and name the file as BackColor.
2. Click **Project** > **Add New Item** and then select the Application Configuration File template. Name the file as app.config.
3. Update the app.config file to match the following XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
         <add key="BackColor" value="Blue"/>
  </appSettings>
</configuration>
```

4. Open Form1.cs and then click **View** > **Code** to switch to the code view.
5. Modify the constructor of `Form1` as follows:

```
public Form1()
{
    InitializeComponent();
    ConfigureColor();
}
```

6. Add a new method to the `Form1` class as follows:

```
private void ConfigureColor()
{
    try
    {
        string colorName = ConfigurationManager.
            AppSettings["BackColor"];
        this.BackColor = Color.FromName(colorName);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

7. Click **Project** > **Add Reference**. The Add Reference dialog box displays.
8. On the .NET tab, select **System.Configuration** and then click **OK**.
9. Add the following `using` directive to the code:

```
 using System.Configuration;
```

10. Set the project as the Startup project.
11. Build and run the project. You'll notice that the background color of the form is blue as specified in the configuration file.
12. Open the BackColor.exe.config file from the project's output folder. Change the color setting from Blue to Green.
13. Run the application again without rebuilding. You'll notice that the background color of the form is changed to green as per the configuration.

Project 3-2: Handling the MouseDown Event

You are developing a game that allows users to hit target areas on a Windows Form with 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 and name the project as MouseDownEvent.
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.
3. Modify the `MouseDown` event handler created by Visual Studio as follows:

```
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.

4. 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 3-3: Handle 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 might be thrown from the code. You need to create proper structured exception handling code to address this requirement.

1. Create a new console application project and name the project as ExceptionTest.

2. Replace the code in the Program.cs file as follows:

```
using System;

namespace ExceptionTest
{
    class Program
    {
        static void Main(string[] args)
        {
            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 3-4: Creating and Handling Events

You are writing code for creating and handling events in your program. The `SampleClass` class 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.

1. Create a new console application project and name the project as SampleEvents.
2. Add a new class file SampleClass.cs and replace its code with the following code:

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

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

3. Replace the code in the Program.cs file with the following:

```
using System;
namespace SampleEvents
{
    class Program
    {
        static private void TestHandler()
        {
            Console.WriteLine(
    "TestHandler is called when the event fires.");
        }

        static void Main(string[] args)
        {
            ISampleEvents se = new SampleClass();

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

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

Lesson 4

Understanding Code Compilation and Deployment

Learning Objectives

Students will learn about:

- Fundamentals of Microsoft Intermediate Language (MSIL) and Common Intermediate Language (CLI)
- Strong naming
- Private and shared assemblies
- Assembly metadata
- Assembly version control

ODN Skills

Understand the fundamentals of Microsoft Intermediate Language (MSIL) and Common Language Infrastructure (CLI)	3.1
Understand the use of strong naming	3.2
Understand assemblies and metadata	3.4
Understand version control	3.3

Lesson Summary — Lecture Notes

Begin the lesson by explaining that the code written in a high-level programming language must be translated before it can be understood by the machine. You should highlight the fact that the .NET Framework is a multi-language environment where the code is compiled in a two-step process. The first is a language compiler and the second is the Common Language Runtime’s just-in-time compilation process.

Next, spend some time discussing the common language infrastructure terminology and introduce the students to common type system (CTS), common language specification (CLS), metadata, virtual execution system (VES) and the common language runtime (CLR).

The next topic deals with the concept of assemblies and metadata. Make sure students understand the structure of an assembly and importance of metadata. Demonstrate how they can create an assembly and peek inside it by using tools such as the Intermediate Language Disassembler (ildasm.exe).

Following-up the discussion on assemblies, introduce the deployment model for assemblies. Introduce to students that the assemblies can be deployed as private assemblies or shared assemblies. Discuss a scenario where you might need a shared assembly rather than a private assembly. Also, discuss the requirements that an assembly must satisfy in order to be used as a shared assembly.

Introduce students to the concept of a strong name and to the different components that makes up a strong name. Demonstrate how to use the strong name tool (sn.exe) to assign a strong name to an assembly. Next, explain the need of delay signing and show the process of delay signing an assembly.

Next, explain the concept of the global assembly cache (GAC). Demonstrate how to install and uninstall an assembly to the GAC by using the global assembly cache tool (gacutil.exe).

In the last section of the lesson, you discuss how the GAC allows multiple versions of a shared assembly to co-exist. Demonstrate how to install two separate version of an assembly to the GAC and then use a specific version of the assembly from a .NET application. Finally, demonstrate how to configure a publisher policy that sets a publisher policy. A publisher policy directs all applications that use an assembly to the new version of the assembly.

Key Terms

assembly - An assembly is a library of compiled code and is the fundamental unit of deployment, version control, and security configuration in the .NET Framework. Depending on how assemblies are deployed, there are two types of assemblies: private and shared.

assembly manifest - An assembly manifest contains information such as the version and identity of the assembly.

Common Intermediate Language (CIL) - Common Intermediate Language (CIL) is an intermediate language that sits between the high-level programming language and the machine code. CIL can’t be directly understood by the processor but it can be efficiently converted to processor-specific code.

Common Language Infrastructure (CLI) - Common Language Infrastructure (CLI) is an international standard approved by ISO and Ecma International. The CLI standard provides the specifications for the executable code and the runtime environment in which the code runs.

Common Language Runtime (CLR) - Common Language Runtime (CLR) is a part of the .NET Framework that is responsible for managing the execution of the programs.

Common Language Specification (CLS) - Common Language Specification (CLS) defines the rules for programming language features. These are the rules that a language must comply with in order to interoperate with other CLS-compliant programming languages.

Common Type System (CTS) - Common Type System (CTS) defines the rules for declaring, using, and managing data types and their operations. CTS helps with language integration and allows for objects written in one language to be used by objects written in another language as if they were written in the same language.

compilation - A code written in high-level programming language must be translated to the machine language before it can be executed. This translation is also known as compilation.

delay signing - Delay signing is the process in which only the public key is extracted from the key pair and stored in a separate file. Then only the public key is used for development and testing. When the code is ready for packaging, the assemblies are signed with private key.

Global Assembly Cache (GAC) - The Global Assembly Cache (GAC) is the central repository for storing shared assemblies on a computer. The GAC simplifies deployment by allowing multiple versions of an assembly to co-exist.

Just-in-Time (JIT) compilation - The CLR uses a process called Just-in-Time (JIT) compilation to convert the CIL code into the code specifically targeted to a particular processor architecture.

language interoperability - Language interoperability is a feature of .NET Framework that enables the code written in a programming language to interact with code written in a different programming language.

metadata - The metadata is a structured way to represent information about a program structure that the CLI uses to locate and load classes at runtime.

Microsoft Intermediate Language (MSIL) - Microsoft Intermediate Language (MSIL) is an intermediate language that sits between the high-level programming language and the machine code.

private assembly - A private assembly is local to the application that uses it and is deployed within the application's directory structure. Private assemblies are designed to be used by only a single application.

private key - The cryptography key pair consists of two related pieces of binary data: a public key and a private key. The private key is the key used for signatures and is assumed to be known only to the assembly's publisher.

public key - The cryptography key pair consists of two related pieces of binary data: a public key and a private key. The public key is the key that represents the identity of a software publisher and can be shared.

publisher policy - Publisher policy is an XML-based assembly created by the publisher of an assembly and released with an upgrade to set the policy on the assembly version that should be used.

shared assembly - An assembly that can be referenced by more than one application is called as a shared assembly. The shared assemblies are all installed at a common, well-known location on the file system known as the Global Assembly Cache (GAC).

side-by-side execution - The execution model where different applications can use different versions of an assembly is referred to as side-by-side execution.

strong name - Strong name specifies an assembly's unique identity. Strong name is used to make sure that the applications can precisely refer to an assembly, thereby avoiding any conflicts in name, version, or publisher.

Virtual Execution System (VES) - VES specifies how the runtime loads and executes CLI-compatible programs. VES provides the services needed to execute code, using the metadata to connect separately generated code at runtime. When the code is executed, the platform-specific VES compiles the CIL to the machine language according to the specific hardware and operating system.

Lesson 4

Understanding Code Compilation and Deployment

Knowledge Assessment

Fill in the Blank

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

1. The __common language infrastructure (CLI)__ standard provides the specifications for the executable code and the runtime environment in which the code runs.
2. The __metadata__ is a structured way to represent information about a program structure that the common language infrastructure uses to locate and load classes at the runtime.
3. The __common language specification (CLS)__ specifies a set of rules that enables the code written in a programming language to interoperate with the code written in other programming languages.
4. A(n) __assembly__ is a library of compiled code and is the fundamental unit of deployment, version control, and security configuration in the .NET Framework.
5. The __Intermediate Language Disassembler (ildasm.exe)__ allows you to view the contents of an assembly (such as the manifest, metadata, and the common intermediate language code).
6. Shared assemblies are signed with a __strong name__ and are deployed to the global assembly cache.
7. The __global assembly cache__ is the central repository for storing shared assemblies on a computer.
8. The __global assembly cache tool (gacutil.exe)__ program can be used to add assemblies to the global assembly cache during development and testing.
9. The __strong name tool (sn.exe)__ program can be used to generate keys needed to create a strong name for an assembly.
10. The __assembly manifest__ of an assembly contains information such as version and identity of the assembly.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You work as a software developer for a large infrastructure company. You are writing a component that will be shared across several applications throughout the company. You want to place the assembly named CommonComponents.dll in the global assembly cache. You have already

stored the company's public key in the assembly manifest for CommonComponents.dll. Which of the following commands should you run to accomplish your task?

a.

```
sn.exe –Vr CommonComponents.dll
```

b.

```
sn.exe –Vu CommonComponents.dll
```

c.

```
gacutil.exe /i CommonComponents.dll
```

d.

```
gacutil.exe /u CommonComponents.dll
```

2. You have written a component that will be shared among multiple applications. You want to prepare the component so that it can be added to the global assembly cache. You have assigned a version number and culture. Next, you want to add the public key information to the assembly manifest for the assembly. Which of the following tools should you use?
   a. sn.exe
   b. gacutil.exe
   c. installutil.exe
   d. al.exe
3. You have written a component that will be shared among multiple applications. You may have multiple versions of this component and each application that might use this component might refer to a specific version of the component. You need to make sure that applications can correctly use the component and you want to minimize deployment efforts. Where should you deploy this component?
   a. Store the component in the application folder for each application.
   b. Add the component to the global assembly cache.
   c. Add the component to the Windows System directory.
   d. Store the component anywhere that you like and specify the location by using the `<codebase>` element in the application configuration file.
4. Which of the following portions of the common language infrastructure defines the rules that a programming language must comply with in order to interoperate with other programming languages?
   a. Common Type System (CTS)
   b. Common Language Specification (CLS)
   c. metadata
   d. Virtual Execution System (VES)
5. Which of the following portions of the common language infrastructure defines the rules for declaring, using, and managing data types and their operations?
   a. Common Type System (CTS)
   b. Common Language Specification (CLS)
   c. metadata
   d. Virtual Execution System (VES)

6. You are working with a set of .NET Framework components that your application will use. You have access to the component assemblies but you don't have access to the component's source code. You need to get information about the classes and methods provided in the assemblies. You decide to look into the assembly contents to look for this information. Which of the following sections of an assembly will give you the needed information?

   a. Manifest

   b. metadata

   c. CIL code

   d. resources

7. You have developed a new business application that helps users in entering sales orders. Some of the application's assemblies need to be added to the global assembly cache. The software application will be installed by the end-user. The installation process should be simple and similar to other applications on the user's computer. Which of the following solutions should you recommend?

   a. Use the Windows Installer technology to install the application.

   b. Use xcopy.exe for copying files to the Program folder and use gacutil.exe to copy the files to the global assembly cache.

   c. Use installutil.exe for copying files to the Program folder and use gacutil.exe to copy the files to the global assembly cache.

   d. Create a console application by using C# to copy the files to their correct locations.

8. You are responsible for maintaining the installation of a Windows application that processes orders. The application uses an assembly named `CommonComponents`. This assembly is signed with a strong name. You have installed version 1.0 of `CommonComponents` to the root directory of your Windows application. You later receive a version 2.0 of `CommonComponents` that you install to the global assembly cache, as well as the root directory of the application. You configure the application's configuration file to redirect calls to version 1.0 of `CommonComponents` with version 2.0. Finally, you receive version 3.0 of the `CommonComponents` assembly, which you install to the global assembly cache. At this time, you do not reconfigure the application configuration file. Now when you run the Windows application, which version of the `CommonComponents` assembly is loaded?

   a. Version 1.0 from the application's root directory

   b. Version 2.0 from the application's root directory

   c. Version 2.0 from the global assembly cache

   d. Version 3.0 from the global assembly cache

9. You ship the version 1.0 of the `CommonComponents` shared assembly. After you ship, you discover a critical defect that must be resolved. You fix the defect and ship a new version (1.1) of the `CommonComponents` assembly. You want to completely override all references to version 1.0 with version 1.1 for all the applications that use the `CommonComponents` assembly on a computer. The solution you suggest must minimize the deployment and configuration efforts. What should you do?

   a. Modify the application configuration file for each application that uses the `CommonComponents` assembly with the following:

```
<?xml version="1.0" ?>
<configuration>
  <runtime>
    <assemblyBinding
      xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity
          name=" CommonComponents"
          publicKeyToken="e3ec12fd024dccae" />
        <bindingRedirect
          oldVersion="1.0.0.0"
          newVersion="1.1.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

b. Modify the application configuration file for each application that uses the `CommonComponents` assembly with the following:

```
<?xml version="1.0" ?>
<configuration>
  <runtime>
    <assemblyBinding
      xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity
          name=" CommonComponents"
          publicKeyToken="e3ec12fd024dccae" />
        <bindingRedirect
          oldVersion="1.1.0.0"
          newVersion="1.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

c. Create a publisher policy file for the `CommonComponents` assembly with the following XML. Convert the XML file to a publisher policy DLL.

```
<?xml version="1.0" ?>
<configuration>
  <runtime>
    <assemblyBinding
      xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity
          name=" CommonComponents"
          publicKeyToken="e3ec12fd024dccae" />
        <bindingRedirect
          oldVersion="1.0.0.0"
          newVersion="1.1.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

d. Create a publisher policy file for the `CommonComponents` assembly with the following XML. Convert the XML file to a publisher policy DLL.

```
<?xml version="1.0" ?>
```

```
<configuration>
  <runtime>
    <assemblyBinding
      xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity
          name=" CommonComponents"
          publicKeyToken="e3ec12fd024dccae" />
        <bindingRedirect
          oldVersion="1.1.0.0"
          newVersion="1.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

10. You create a new shared code library, CommonComponent.dll. You also install the corresponding assembly, `CommonComponents`, to the global assembly cache. You now need to uninstall the assembly from the global assembly cache. Which of the following commands should you use?

    a.

    `gacutil /i MathUtil.dll`

    b.

    `gacutil /l MathUtil`

    c.

    `gacutil /u MathUtil.dll`

    d.

    `gacutil /u MathUtil`

Competency Assessment

Project 4-1: Looking at an Assembly Manifest

Your team is developing code that uses assemblies developed by other companies. You have access to a code library in GraphTools.dll but you do not have access to the source code. You need to find the version of the assembly to make sure that you use the correct version. How should you verify the version of the assembly?

1. Open the Intermediate Language Disassembler (ildasm.exe) tool.
2. Click **File** > **Open** and then browse to the GraphTools.dll.
3. The version of the assembly is displayed at the bottom panel; if the version is 1.1, it will display text similar to the following:
   ```
   .assembly GraphTools
   {
     .ver 1:1:0:0
   }
   ```
4. Alternatively, open the Manifest node by double-clicking on it. In the file, look for the block of code that reads `.assembly GraphTools`:

```
              .assembly GraphTools
             {
               // extra information here...
               .ver 1:1:0:0
      }
```

Project 4-2: Uninstalling an Assembly from the Global Assembly Cache

Your team is developing code that is compiled into the vcsComponents.dll file. The name of this culture-neutral assembly is `VcsComponents` and its public key token is a3db12fd012cbdbe. You install version 1.0 and version 1.1 of this assembly to the global assembly cache. You now need to uninstall only the version 1.0 from the assembly. What should you do?

1. Open the Visual Studio Command Prompt as an administrator.
2. Run the following command:

```
gacutil.exe /u VcsComponents, Version=1.0.0.0,
 Culture=neutral,PublicKeyToken= a3db12fd012cbdbe
```

3. Verify that the assembly is removed from the global assembly cache by running the following command:

```
gacutil.exe /l VcsComponents
```

Notice that only the version 1.1 of `VcsComponents` is listed.

Proficiency Assessment

Project 4-3: Using Multiple Languages

You are writing code for an application that uses functionality from other reusable assemblies. Your application's code is written in Visual Basic. The application uses an assembly, `MathUtil`, which was compiled by using a C# compiler. How would you write code that calls the code written in a different .NET Framework programming language? (for this project, you can use the `MathUtil` C# library project that you created earlier in this lesson.)

1. Create a new Visual Basic console application project named VBApplication.
2. Click **Project** > **Add Reference**. In the Add Reference dialog box, select the `MathUtil` project and click **OK**.
3. Replace the code in the Module1.vb with the following VB code:

```
Imports MathUtil
Module Module1
    Sub Main()
        Dim r As New RandomNumber() With { _
         .MinValue = 0, _
         .MaxValue = 500 _
        }
        Console.WriteLine(r.[Get]())
    End Sub
End Module
```

4. Build the project.
5. Click **Project > Set as Startup Project** to set the project as the startup project.
6. Run the project. Notice that the program calls a method, which was originally written in C#, to generate a random number and displays the number on the Console.

Project 4-4: Delay Sign an Assembly

Your team is developing code that will be deployed as shared assembly. Multiple programmers are involved in developing the code. The code for the shared assembly is compiled into a file named vcsComponents.dll. This code needs to be signed with a strong name so that it can be deployed to the global assembly cache for development and testing. You need to limit access to the private key. The key-pair file keyfile.snk contains both public and private keys. What steps should you take to add the vcsComponents.dll file to the global assembly cache?

1. Extract the public key to a different file, say publicKey.dll, by using the following command :
   ```
   sn.exe –p keyfile.snk publickey.snk
   ```
2. Change the Visual studio project properties to sign the vcsComponents.dll by using the publickey.snk file.
3. Turn off the security verification for vcsComponent.dll by using the following command:
   ```
   sn.exe –Vr vcsComponents.dll
   ```
4. Deploy the assembly to the global assembly cache by using the following command:
   ```
   gacutil /i vcsComponents.dll
   ```

Lesson 5

Understanding Input/output (I/O) Classes

Learning Objectives

Students will learn about:
- The console class
- Command-line arguments
- File operations
- Reading and writing text files
- Reading and writing binary files
- Reading and writing XML files
- XML Schema

ODN Skills

Understand console I/O	4.2
Understand .NET file classes	4.1
Understand XML classes in the .NET Framework	4.3

Lesson Summary — Lecture Notes

Lesson 5 explores the various input and out classes in the .NET Framework. The first section of the lesson covers the Console applications. These are the applications that do not have a graphical user interface. Console-based applications are best suited for tasks that require minimal or no user interface. The `Console` class in the `System` namespace provides methods to interact with the console window.

The next section of the lesson covers the .NET Framework classes that manipulate disk files. The `File` and `FileInfo` classes in the `System.IO` namespace provide methods for working with the disk files. The lesson also covers how to create text files as well as binary files. A text file is a disk file that stores only character-based data. The `StreamReader` and `StreamWriter` classes provide methods to respectively read data from and write data to text files. A binary file is a disk file that can store any type of data. Examples of binary data include mathematical data, image data, video data, audio data, or a combination of them all. The `BinaryReader` and `BinaryWriter` classes provide methods to respectively read data from and write data to binary files.

The final section in the lesson shows students how to work with the XML data using the classes in the `System.Xml` namespace. XML is a text-based format for representing structured data. The lesson discusses how to use the `XmlReader` and `XmlWriter` classes to respectively read data from and write data to XML files. The final topic of the lesson covers the topic of XML schema and how to validate an XML document against an XML schema.

Key Terms

attribute - An attribute is a piece of data that further describes an XML element.

backing store - A backing store represents the source or destination of the stream. A backing store might be anything, such as a disk file, memory, network connection, and so on.

binary file - A binary file is a disk file that can store any type of data. Examples of binary data include mathematical data, image data, video data, audio data, or a combination of them all.

command-line arguments - The command-line arguments are the values passed to the Main method from the operating system.

command piping - Command piping is the process of chaining commands by passing the output of a command as an input to another command. With command piping, you can create more powerful commands by combining simple commands.

console applications - A console application is a program designed to be used via a text-based interface. Console-based applications are best suited for tasks that require minimal or no user interface.

element - An opening tag and closing tag together with their content is called as an XML element.

stream - A stream represents a flow of raw data.

text files - Text files are the data files that contain only character-based data. These files are often organized as lines of text separated by end-of-line characters.

XML - XML (Extensible Markup Language) is a text-based format for representing structured data. XML is widely used in applications that exchange information between organizations, Web site publishing, object serialization, remote procedure calls, and so on.

XML schema - XML schema describes the structure of an XML document. An XML document is considered valid only when it conforms to its XML schema.

Lesson 5

Understanding Input/Output (I/O) Classes

Knowledge Assessment

Fill in the Blank

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

1. You find classes for working with streams and backing stores in the __System.IO___ namespace.
2. The ___XML___ format is a hierarchical data representation format.
3. The ___ReadKey___ method of the `Console` class reads the next character or function key pressed by the user.
4. Files in the ___binary___ format can allow you to store text, images, and video data.
5. Some objects open unmanaged operating resources, such as file handles. When you are done working with these objects, be sure to call the ___Dispose___ method to release the unmanaged resources back to the operating system.
6. ___Console applications__ do not have a graphical user interface; they use a text-mode console window to interact with users.
7. ___XML schema___ describes the structure of an XML document.
8. The ___command-line arguments___ are the values passed to the Main method from the operating system.
9. The .NET Framework’s classes to work with XML data are stored organized as part of the ___System.Xml___ namespace.
10. ___Character encoding___ describes the rules by which each character is represented inside a text file.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Your application needs to store the product image out to a disk file. You'd like to minimize the size of the disk file. Which object should you use to write the file?
   a. `FileStream`
   b. `StreamWriter`
   c. `BinaryWriter`
   d. `XmlWriter`

2. You are developing a Console Application by using C#. When the user presses a key, your program needs to respond immediately. Which of the following methods of the `Console` class should you use?

   a. `Read`

   b. `ReadKey`

   c. `ReadLine`

   d. `OpenStandardInput`

3. You are developing a C# application that creates text files. You write the following code:

```
using (StreamWriter sw =
     new StreamWriter("file.txt"))
{
     sw.Write("Sample Text");
}
```

   When the text is written to the file, what encoding scheme is used?

   a. UTF8Encoding

   b. UTF7Encoding

   c. ASCIIEncoding

   d. UnicodeEncoding

4. You are developing an application that manipulates XML data. You are reviewing the rules that make XML data well-formed and valid. Which of the following statements are true? (Choose all that apply)

   a. Any XML that conforms to the syntactical rules for XML is considered well-formed.

   b. Any XML that conforms to the syntactical rules for XML is considered valid.

   c. When an XML file conforms to a predefined schema, the file is considered to be well-formed.

   d. When an XML file conforms to a predefined schema, the file is considered to be valid.

   e. XML schema must be present in the same file as the XML data.

5. You are writing a C# program that writes data to a binary file. You write the following code:

```
static void WriteBinaryFile(string fileName)
{
    Int32 int32Data = 5;
    string stringData = "Sample String";
    Single singleData = 3.141f;
    bool boolData = true;

    using (BinaryWriter writer =
          new BinaryWriter(
          File.Open(fileName, FileMode.Create)))
    {
        writer.Write(int32Data);
        writer.Write(stringData);
        writer.Write(singleData);
        writer.Write(boolData);
    }
}
```

You now need to read data from this file. Which of the following code segments should you choose?

a.

```
static void ReadBinaryFile(string fileName)
{
    if (File.Exists(fileName))
    {
        using (BinaryReader reader =
            new BinaryReader(
            File.Open(fileName, FileMode.Open)))
        {
            Console.WriteLine(reader.ReadInt32());
            Console.WriteLine(reader.ReadString());
            Console.WriteLine(reader.ReadSingle());
            Console.WriteLine(reader.ReadBoolean());
        }
    }
}
```

b.

```
static void ReadBinaryFile(string fileName)
{
    if (File.Exists(fileName))
    {
        using (BinaryReader reader =
            new BinaryReader(
            File.Open(fileName, FileMode.Open)))
        {
            Console.WriteLine(reader.ReadBoolean());
            Console.WriteLine(reader.ReadSingle());
            Console.WriteLine(reader.ReadString());
            Console.WriteLine(reader.ReadInt32());
        }
    }
}
```

c.

```
static void ReadBinaryFile(string fileName)
{
    if (File.Exists(fileName))
    {
        using (BinaryReader reader =
            new BinaryReader(
            File.Open(fileName, FileMode.Open)))
        {
            Console.WriteLine(reader.ReadString());
            Console.WriteLine(reader.ReadString());
            Console.WriteLine(reader.ReadString());
            Console.WriteLine(reader.ReadString());
        }
    }
}
```

d.

```
static void ReadBinaryFile(string fileName)
{
    if (File.Exists(fileName))
    {
        using (BinaryReader reader =
            new BinaryReader(
            File.Open(fileName, FileMode.Open)))
        {
            Console.WriteLine(reader.ReadBytes());
            Console.WriteLine(reader.ReadBytes ());
            Console.WriteLine(reader.ReadBytes ());
            Console.WriteLine(reader.ReadBytes ());
        }
    }
}
```

6. Your Windows application needs to perform frequent copy and move operations on a particular file. You want your application to be as optimized as possible. Which of the following classes should you use to perform these operations?
   a. `File`
   b. `FileInfo`
   c. `FileStream`
   d. `Stream`
7. You are developing a C# application that creates text files. You need to use an encoding scheme that supports storing characters from international languages in the text file. Which encoding scheme should you use?
   a. UTF8Encoding
   b. UTF7Encoding
   c. ASCIIEncoding
   d. UnicodeEncoding
8. You are developing a C# application that reads data from disk files. The disk files stores text and video in a pre-defined and structured format. Which of the following .NET Framework class should you use to read such data from disk files?
   a. StreamReader
   b. BinaryReader
   c. XmlReader
   d. StringReader
9. You are developing a C# application that manipulates disk files. The program needs to copy, delete and move the disk files. Which of the following .NET Framework class should you use to manipulate disk files?
   a. File
   b. Directory
   c. FileStream

d. MemoryStream

10. You are developing a C# application that process data in XML files. You are analyzing an XML document with the following data:

```
<?xml version="1.0" encoding="utf-8"?>
<!--Customer List-->
<Customers>
  <Customer Id="ALFKI">
    <CompanyName>Alfreds Futterkiste</CompanyName>
    <Phone>030-0074321</Phone>
  </Customer>
  <Customer Id="EASTC">
    <CompanyName>Eastern Connection</CompanyName>
    <Phone>(171) 555-0297</Phone>
  </Customers>
</Customer>
```

Which of the following observations are true for this XML document (select all that apply)?

a. The XML document is well formed

b. The XML document is not well formed

c. The XML document is valid

d. The XML document is not valid

Competency Assessment

Project 5-1: Working with Encoding

You are developing a program that manipulates text files. You need to develop a console application that writes a text file by using the Unicode encoding scheme. You also need to read the text from the file that was created. How would you write such a program?

1. Create a new project based on the Console Application template. Name the project as Encoding.
2. In the Program.cs file, modify the code inside the `Program` class as follows:

```
static void Main(string[] args)
{
    string fileName = "fileList.txt";
    WriteTextFile(fileName);
    ReadTextFile(fileName);
}

static void WriteTextFile(string fileName)
{
    DirectoryInfo dInfo = new DirectoryInfo(
        Environment.GetFolderPath(
        Environment.SpecialFolder.MyDocuments));
    StringBuilder sb = new StringBuilder();

    foreach (FileInfo fileInfo in dInfo.GetFiles())
    {
        sb.AppendLine(fileInfo.Name);
    }

    using (StreamWriter sw =
        new StreamWriter(filename, false,
            new UnicodeEncoding()))
    {
        sw.Write(sb.ToString());
    }
}

static void ReadTextFile(string fileName)
{
    try
    {
        using (StreamReader sr =
          new StreamReader(filename,
              new UnicodeEncoding()))
        {
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                Console.WriteLine(line);
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: {0}", ex.Message);
    }
}
```

3. Add the following `using` directive to your code:

```
using System.IO;
```

4. Click **Project** > **Set as StartUp Project** to set the project as the startup project.
5. Build and run the program (press **Ctrl+F5**) and then review the output in the console window.

Project 5-2: Working with Typed XML Data

You are developing a program that reads and writes XML data. Your program should be able to read and write typed values rather than their string representation. How would you write such a program?

1. Create a new console application project named TypedXmlData.
2. Add the following code to the `Main` method of the `Program` class:

```
static void Main(string[] args)
{
    string fileName = "Items.xml";
    WriteXmlFile(fileName);
    ReadXmlFile(fileName);
}
```

```
static void WriteXmlFile(string fileName)
{
    XmlWriterSettings settings =
   new XmlWriterSettings();
      settings.Indent = true;
      settings.IndentChars = "  ";
      settings.NewLineOnAttributes = true;

    using (XmlWriter writer =
        XmlWriter.Create(fileName, settings))
    {
        writer.WriteStartElement("Items");
        writer.WriteStartElement("Item");
        writer.WriteStartElement("Id");
        writer.WriteValue(10);
        writer.WriteEndElement();
        writer.WriteElementString(
                "Name", "Rye Bread");
        writer.WriteStartElement("Price");
        writer.WriteValue((double)5.50);
        writer.WriteEndElement();
        writer.WriteEndElement();
        writer.WriteEndElement();
        writer.Flush();
    }
}

static void ReadXmlFile(string fileName)
{
    using (XmlReader reader =
                XmlReader.Create(fileName))
    {
        while (reader.Read())
        {
            if (reader.IsStartElement())
            {
                switch (reader.Name)
                {
                    case "Id":
                      Console.WriteLine(
                      "Id: {0}", reader.
                      ReadElementContentAsInt());
                      break;
                    case "Name":
                      Console.WriteLine(
                      "Name: {0}", reader.
                      ReadElementContentAsString());
                      break;
                    case "Price":
                      Console.WriteLine(
                      "Price: {0}", reader.
                      ReadElementContentAsDouble());
                      break;
                }
            }
        }
    }
}
```

3. Add the following `using` directive to the program:

```
using System.Xml;
```

4. Build and run the program (press **Ctrl+F5**) and then review the output in the console window.

Proficiency Assessment

Project 5-3: Working with Console Input

You are developing a program that manipulates text input from the console. You need to develop a console application that accepts text from users and converts that text to uppercase letters. The application should not convert input that is not characters, but should still output this input. How would you write such a program?

1. Create a new console application project named 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 characters on the command line and press **Enter**. You will see that letters are converted to upper case.
4. Press **Ctrl+Z** (end of input) to end the program.

Project 5-4: Append to a Text File

You are developing a program that manipulates text files. Your program needs to open the LogFile.txt file (or create the file if it does not already exist) and then append a message with the current time to the file. You also need to write the contents of the file to the console window for display. How would you write such a program?

1. Create a new console application project named TextAppend.
2. Add the following code to the `Main` method:

```
static void Main(string[] args)
{
```

```
using (StreamWriter w =
    File.AppendText("LogFile.txt"))
{
    w.WriteLine(
        String.Format("Logged on: {0}\n",
        DateTime.Now));
}
using (StreamReader r
    = File.OpenText("LogFile.txt"))
{
    string line;
    while ((line = r.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}}
```

3. Build and run the program (press **Ctrl+F5**). The current time that was written to the text file is displayed on the screen.
4. Run the program again. You will see that a new time entry is created in the file but the previous time entry is still preserved.

Lesson 6

Understanding Security

Learning Objectives

Students will learn about:

- Authentication
- Authorization
- Cryptography
- Code access security

ODN Skills

Understand the System.Security namespace	5.1
Understand authentication and authorization.	5.2

Lesson Summary — Lecture Notes

In Lesson 6, you will introduce students to the security features of the .NET Framework. The .NET Framework provides several classes that help developers write secure code and control access to protected resources. Students should know that these classes are available as part of the System.Security namespace.

You'll begin the lesson by differentiating authentication from authorization. Give examples to demonstrate for students how they can programmatically retrieve a user's identity and a user's role. At this point also introduce the concept of role-based security in business applications.

The next section of the lesson discusses .NET Framework's implementation of standard cryptography algorithms. Students should be aware of the different encryption algorithms available to them and should be able to choose the best encryption algorithm for a given scenario. Make sure that the students also understand the difference between the secret-key encryption and public-key encryption.

The final section of the lesson is focused on the concept of code-access security. First, explain how the .NET Framework categorizes code as transparent, security-critical, and safe-critical. Next, introduce the concept of permissions. Permissions refer to the actions that a code is allowed to or not allowed to perform. Finally, discuss how to use access control to set who can access specific resources or control access to resources provided by the application.

Key Terms

authentication - Authentication refers to the process of obtaining credentials from a user and verifying his or her identity.

authorization - Authorization is the process of determining whether an authenticated identity is allowed to perform a requested action.

code access security - Code access security is a security mechanism provided by the .NET Framework to manage what a code running on a computer system is allowed to do.

cryptography - Cryptography encrypts the data so that it cannot be by viewed by unauthorized users and to detect whether the data has been modified. Cryptography also helps establish the identity of the sender so that you can trust that the message is actually coming from the claimed sender.

permissions - Permissions refer to the actions that a code is allowed to or not allowed to perform.

permission sets - Permission sets are a predefined collection of permissions that are applied together.

public-key encryption - Public-key encryption protects data by using a set of two cryptographically paired keys. One key is called a private key and it is known only to its owner. The second key is called a public key, which belongs to the owner, but as the name suggests, it can be made public to anyone.

secret-key encryption - In the secret-key encryption technique, both the sender and receiver of the message share a secret encryption key. The sender encrypts the message before sending it across and the receiver uses the same key to decrypt the message.

Transparency Level 2 – Way of grouping the code in the three categories: Transparent, Security-Critical, and Safe-Critical. Transparency separates code that can do privileged things (also called critical code), such as calling native code, and code that cannot (also called transparent code).

Lesson 6

Understanding Security

Knowledge Assessment

Fill in the Blank

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

1. __Authentication__ refers to the process of obtaining credentials from a user and verifying his or her identity.
2. __Authorization__ is the process of determining whether an authenticated identity is allowed to perform a requested action.
3. The __public-key__ encryption technique uses a set of two distinct keys that are cryptographically paired.
4. The code classified as __security-transparent__ cannot do anything that is security-sensitive. This code cannot call any unsafe code or any code that is classified as security-critical.
5. Role-based security revolves around two interfaces: __`IIdentity`__ and __`IPrinicipal`__.
6. The __`IsInRole`__ method of the `WindowsPrincipal` object can be used to determine whether the current user is in a specific windows group.
7. The secret-key encryption is also known as __symmetric__ encryption because the same key is used for both encryption and decryption.
8. The __`RSACryptoServiceProvider`__ class implements a public-key cryptography algorithm based on the RSA algorithm and allows both encryption and signing.
9. __Permissions__ refer to the actions that code is allowed to or not allowed to perform.
10. The __System.Security.Permissions__ namespace defines the classes that specify the permissions you can apply to an assembly declaratively or programmatically.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Your application requires the user to be in the Domain Admins group in order to activate certain functions. Which .NET security feature should you use to ensure that the user is in this group?
   a. Code access security
   [✓ b. Role-based security]
   c. Encryption
   d. Type safety

2. You are developing a program that implements role-based security. You need to check a user for membership in the Windows groups. Which of the following methods should you use to accomplish your requirements?

   a. `WindowsPrincipal.IsInRole` method

   b. `WindowsIdentity.GetCurrent` method

   c. `WindowsIdentity.Impersonate` method

   d. `PrincipalPermission.IsSubsetOf` method

3. You are developing a program that implements role-based security. You need to determine whether the user belongs to a custom Windows user group that has a specified name. Which form of the `WindowsPrincipal.IsInRole` method should you use?

   a. `IsInRole(Int32)`

   `b. IsInRole(SecurityIdentifier)`

   `c. IsInRole(string)`

   `d. IsInRole(WindowsBuiltInRole)`

4. You are developing a C# application that needs to use secret-key encryption. You need to provide the highest-level of encryption that is compliant with the US Federal Information Processing Standards (FIPS). Which cryptography class should you use?

   a. `RijndaelManaged`

   b. `AesManaged`

   c. `TripleDESCryptoServiceProvider`

   d. `DSACryptoServiceProvider`

5. You are developing a C# application that needs to use public-key encryption and digital signing. You need to use an algorithm that provide highest-level of encryption and allow both encryption and signing. Which cryptography class should you use?

   a. `AesManaged`

   b. `RSACryptoServiceProvider`

   c. `DSACryptoServiceProvider`

   d. `TripleDESCryptoServiceProvider`

6. You are developing a C# application that uses the `RSACryptoServiceProvider` class. You need to provide the public key information but protect the private key. What should you do to accomplish this requirement?

   a. Call the `ExportParameters` method with a parameter value of `false`.

   b. Call the `ExportParameters` method with a parameter value of `true`.

   c. Call the `Encrypt` method with a parameter value of `true`.

   d. Call the `Encrypt` method with a parameter value of `false`.

7. You are developing a C# application that uses a variable named asm of the type Assembly. You check that the value of the expression, `asm.IsFullyTrusted` is false. Which of the following statements are true for the assembly? (Choose all that apply)

a. The assembly is fully trusted

b. The assembly is partially trusted

c. All the types in the assembly are running as `SecurityTransparent` code

d. All the types in the assembly are running as `SecuritySafeCritical` code

e. All the types in the assembly are running as `SecurityCritical` code

8. You downloaded a .NET assembly from an Internet Website to your local computer. You run the code directly on the local computer. Which of the following statements are true for the assembly? (Choose all that apply)

a. The assembly is fully trusted

b. The assembly is partially trusted

c. All the types in the assembly are running as `SecurityTransparent` code

d. All the types in the assembly are running as `SecuritySafeCritical` code

e. All the types in the assembly are running as `SecurityCritical` code

9. You are developing an application that doesn't need to access protected resource. You want to ensure that your code is never used to accidentally or maliciously accesses protected resources. What should you do to accomplish this?

a. Apply the following attribute to the assembly

```
[assembly: SecurityTransparent()]
```

b. Apply the following attribute to the assembly

```
[assembly: SecurityCritical()]
```

c. Apply the following attribute to the assembly

```
[assembly: SecuritySafeCritical()]
```

d. Apply the following attribute to the assembly

```
[assembly: SecurityRules(SecurityRuleSet.Level2)]
```

10. You need to create a partially trusted host for an assembly in your application. What should you do to accomplish this?

a. Apply the following attribute to the assembly

```
[assembly: SecurityRules(SecurityRuleSet.Level1)]
```

b. Apply the following attribute to the assembly

```
[assembly: SecurityRules(SecurityRuleSet.Level2)]
```

c. Apply the following attribute to the assembly

```
[assembly: AllowPartiallyTrustedCallers()]
```

d. Use the `AppDomain.CreateDomain` method

Competency Assessment

Project 6-1: Using the RijndaelManaged Class for Secret-Key Encryption

You are developing a program that manipulates text files. You need to encrypt and decrypt your files by using the `RijndaelManaged` class. How would you write such a program?

1. Create a new project based on the Console Application template. Name the project as RijndaelAlgorithm.
2. Modify the code in the default Program file as follows:

```
using System;
using System.IO;
using System.Security.Cryptography;

namespace RijndaelAlgorithm
{
    class Program
    {
        static void Main(string[] args)
        {
            string plainText =
                "Lorem ipsum dolor sit amet...";
            using (var rm = new RijndaelManaged())
            {
                Console.WriteLine(
                    "Before Encryption: {0}",
                     plainText);
                byte[] encryptedBytes =
                    Encrypt(plainText,
                    rm.Key, rm.IV);
                string decryptedText =
                    Decrypt(encryptedBytes,
                    rm.Key, rm.IV);
                Console.WriteLine(
                    "After Decryption : {0}",
                    decryptedText);
            }
        }

        private static byte[] Encrypt(
            string plainText, byte[] key,
            byte[] iv)
        {
            byte[] encryptedBytes;

            using (var rm = new RijndaelManaged())
            {
                rm.Key = key;
                rm.IV = iv;

                ICryptoTransform encryptor =
                    rm.CreateEncryptor(
                    rm.Key, rm.IV);
                using (MemoryStream ms =
                    new MemoryStream())
                {
                    using (CryptoStream cs =
                        new CryptoStream(
                        ms, encryptor,
                        CryptoStreamMode.Write))
                    {
                        using (StreamWriter sw =
                            new StreamWriter(cs))
                        {
                            sw.Write(plainText);
                        }
                        encryptedBytes =
                              ms.ToArray();
                    }
                }
            }
            return encryptedBytes;
        }

        private static string Decrypt(
            byte[] encryptedBytes, byte[] key,
            byte[] iv)
        {
            string decryptedText = null;
            using (var rm = new RijndaelManaged())
            {
                rm.Key = key;
                rm.IV = iv;

                ICryptoTransform decryptor =
                    rm.CreateDecryptor(rm.Key,
                    rm.IV);
                using (MemoryStream ms =
                 new MemoryStream(encryptedBytes))
                {
                    using (CryptoStream cs =
                        new CryptoStream(
                        ms, decryptor,
                        CryptoStreamMode.Read))
                    {
                        using (StreamReader sr =
                            new StreamReader(cs))
                        {
                            decryptedText =
                             sr.ReadToEnd();
                        }
                    }
                }
            }
            return decryptedText;
        }
    }
}
```

3. Build and run the program. Notice that the console window displays the message before it is encrypted and after it is decrypted.

Project 6-2: Getting User Identity and Group Membership

You are developing a program that needs to find out the identity of the logged-in Windows user and the name of the Windows Groups that the user is a member of. How would you write a program to retrieve this information?

1. Create a new project based on the Console Application template. Name the project as GetWindowsIdentity.
2. Modify the code in the default Program file as follows:

```
using System;
using System.Security.Principal;

namespace GetWindowsIdentity
{
    class Program
    {
        static void Main(string[] args)
        {
            AppDomain.CurrentDomain
                .SetPrincipalPolicy(
                PrincipalPolicy.WindowsPrincipal);

            WindowsIdentity identity =
               WindowsIdentity.GetCurrent();
            Console.WriteLine(
               "Windows Principal: {0}",
               identity.Name);

            foreach (var item in identity.Groups)
            {
                Console.WriteLine(
                item.Translate(
                typeof(NTAccount)));
            }
        }
    }
}
```

3. Build and run the program. Notice that the console window displays the name of the logged-in Windows user and the name of the groups that the user is a member of.

Proficiency Assessment

Project 6-3: Working with Access Control List

You are developing a program that manipulates files. You need to find the access control list for a given file such as the access control type and file system rights for the Windows users or group accounts. How would you write such a program?

1. Create a new project based on the Console Application template. Name the project as AccessControl.
2. Modify the code in the default Program file as follows:

```
using System;
using System.IO;
using System.Security.AccessControl;
using System.Security.Principal;

namespace AccessControl
{
    class Program
    {
        static void Main(string[] args)
        {
            string fileName = "TestFile.txt";
            using (StreamWriter writer =
                File.CreateText(fileName))
            {
                writer.WriteLine(
                "Lorem ipsum dolor sit amet...");
            }

            var stream = File.Open(
                   fileName, FileMode.Open);
            var fileSecurity =
                   stream.GetAccessControl();

            foreach (var item in
                fileSecurity.GetAccessRules(
                true, true, typeof(NTAccount)))
            {
                var rule = item as
                  FileSystemAccessRule;
                Console.WriteLine(
                  "Identity Reference : {0}",
                  rule.IdentityReference.Value);
                Console.WriteLine(
                  "Access Control Type: {0}",
                  rule.AccessControlType);
                Console.WriteLine(
                  "File System Rights : {0}\n",
                  rule.FileSystemRights);
            }
        }
    }
}
```

3. Build and run the program. The console window displays the message as shown in Figure 6-5.

Insert Figure06-05.png here.

Figure 6-5
Displaying the Access Control settings for a file

Project 6-4: Verifying Role Membership

You are developing a program that needs to find out if the logged-in Windows user is a member of the Windows built-in Administrator role. How would you write a program to display this information?

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

2. In the `Program` class, add the following `using` directive:

```
using System.Security.Principal;
using System.Threading;
```

3. In the Program.cs file, modify the code inside the `Program` class as follows:

```
static void Main(string[] args)
{
    AppDomain.CurrentDomain.SetPrincipalPolicy(
        PrincipalPolicy.WindowsPrincipal);
    WindowsPrincipal principal =
        (WindowsPrincipal) Thread.CurrentPrincipal;
    bool isAdmin = principal.IsInRole(
        WindowsBuiltInRole.Administrator);

    Console.WriteLine(
        "You are {0}in the Administrators role",
        isAdmin ? "" : "not ");
}
```

4. Set the project as the startup project.
5. Build and run the program (press Ctrl+F5) and then review the output in the console window.