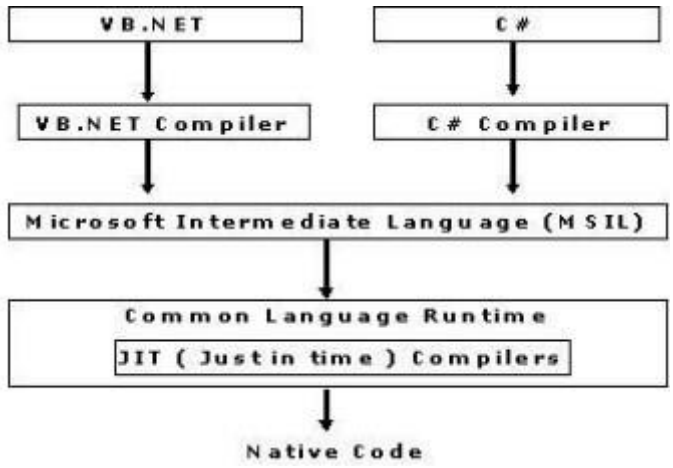
Sub:	Programming Using C#						
Date:	20/05/23	Duration:	90 min's	Max Marks:	50	Sem:	IV

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

PART I

1 **Briefly discuss the role of Common Language Runtime (CLR) in .NET Framework with a neat diagram.**

Ans : .NET Framework provides runtime environment called Common Language Runtime (CLR). CLR is run time environment in which programs written in C# and other .NET language are executed. The code which runs under the CLR is called as Managed Code. CLR also supports services that the application uses to access various resources such as collections, arrays and operating system folders. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management. It also supports cross-language interoperability. The runtime automatically releases the objects when they are no longer in use. Programmatically, when our program needs memory, CLR allocates the memory for scope and de-allocates the memory if the scope is completed. Language Compilers (e.g. C#, VB.Net, J#) will convert the Code/Program to Microsoft Intermediate Language (MSIL) intern this will be converted to Native Code by CLR. CLR provides different functionality for the applications.



- CLR Compiles application into the runtime, compile the IL code into native code, execute the code
- Run-time environment
- Memory management
- Run-time services
- Allocation of Memory
- De-Allocation of Memory (garbage Enforces Security
- Type safety collation)
- Code Security based on Trust (granted permission to execute code.
- Code Debugging support
- Thread support
- Exception Management.

	Managed code:- This code is directly executed by the CLR. The applications that are created using managed code automatically have CLR services such as type checking, security and automatic garbage collection. The CLR compiles the applications to intermediate language (IL) which is CPU independent. The IL along with the metadata that describes the attributes, classes and methods of the code reside in an assembly.
2	**What is .NET Framework? Explain the benefits of .Net framework.** .NET Framework .NET Framework is the original implementation of .NET. It supports running apps, desktop applications, websites etc on Windows. It was designed and maintained by Microsoft. Using .NET Framework we can write/build the following types of applications: Console applications Windows applications Web applications Web services Basics of .NET Programs developed with .NET needs a execution engine that handles running applications. It is called Common Language Runtime (CLR). It provides services like thread management, garbage collection, exception handling etc. .NET allows using types defined by one .NET language to be used by another under the Common Language Infrastructure (CLI) specification . Any language that conforms to the CLI specification of .NET can run in the .NET run-time. Following are the few .NET languages. • C# • C++ (CLI version) • J# (CLI version of Java) • A# (CLI version of ADA) • L# (CLI version of LISP) • IronRuby (CLI version of RUBY) **Benefits of .NET Framework Consistent Programming Model** :- Provides a consistent object oriented programming model across different languages to create programs for performing different tasks such as connecting to and retrieving data from databases , and reading and writing into files. Cross- Platform support:- Specifies that any windows platform that supports CLR can execute .NET application that is a .NET application enables interoperability between multiple windows operating systems. Language Interoperability:- Enables code written in different languages to interact with each other. This allows reusability of code and improves the efficiency of development process. Automatic Management of Resources :- While developing application we need not manually free up the application resources such as files, memory, network and database connections. The framework provides 3 a feature called CLR that automatically tracks the resource usage and helps you in performing the task of manual resource management. Ease of development:- The framework installs applications or components that do not affect the existing applications. In most cases, to install an application we need to copy the application along with its components on the target computer. In .NET applications are deployed in the form of assemblies. Registry entries are not required to store information about components and applications. In addition assemblies store information about different versions of a single component used by an application. This resolves the version problem .
	PART II
3	**Explain ASP.Net and ADO.Net** ASP.NET is a web development mode, which is used to deliver interactive and data-driven web application over the internet. It also consists of a large number of controls, such as text boxes, buttons,

	and labels for assembling, configuring, and manipulating code to create Hyper Text Markup Language (HTML). Better Performance: Specifies that when you request a web page for the first time after compiling •Advantages of ASP.NET: Better Performance : ASP.NET code, the CLR compiles the code and stores the cached copy of the result. Now, for any subsequent calls to the same page, the cached copy of the result is retrieved instead of going back to the server. Improved Security: Refers to the different methods of authentication included in ASP.NET:•server. Forms: Allowsthe ASP.NET application to use its own custom businesslogic for authentication. Windows: Checks the identity of a user against the Windows user accounts that are stored on the Web Server. If the credentials of a user match with that of a Windows user account, thenthe user is authenticated Greater Scalability: Specifies that the session states in ASP.NET are maintained in a separate process on• a different machine or database. This enables cross-server sessions to occur, solving the problem of web Cookie-less Sessions: Specifies that ASP.NET stores the session even when the cookies in a Web browser•forms when more web servers need to be added as the traffic grows. are disabled. In such a case, the session ID is passed as a part of the Uniforms Resource Locator (URL **ADO .NET**: ADO.NET is a technology used for working with data and databases of all types. It provides access to data sources, such as Microsoft SQL Server, data sources exposed through OLE DB, and Extensible Markup Language (XML). **Disconnected Data Architecture:** Implies that applications connect to the database only when data needs to be retrieved or modified. After the database operation has been performed, the connection to the database is closed. To perform any database operation again, the connection with the database will have to be re-established. **Cached Data in Datasets**: Follows a disconnected architecture for accessing or modifying data. The data is accessed and later stored in the datasets. A dataset is a cached set of database records, which is independent of data source. Even when you are disconnected from the database on which you are working, you can make modifications in the database. **Scalability:** Reduces the traffic on the database and saves the resources to make the database more efficient. ADO.NET help in attaining scalability by performing all database operations on the dataset instead of on the database. **Transfer of Data in XML Format:** Transfers data from a database into a dataset and from the dataset to another components using XML, which is the standard format used for transferring data in ADO.NET **Interaction with the Database through Data Commands**: All operations on database are performed, such as retrieving, modifying, or updating of data using data commands. A data command is either a Structured Query Language (SQL) statement or a stored procedure
4	**Make a short note on LINQ with examples.** The acronym LINQ is for Language Integrated Query. Microsoft's query language is fully

integrated and offers easy data access from in-memory objects, databases, XML documents and many more. It is through a set of extensions, LINQ ably integrate queries in C# and Visual Basic. Developers across the world have always encountered problems in querying data because of the lack of a defined path and need to master a multiple of technologies like SQL, Web Services, XQuery, etc. Introduced in Visual Studio 2008 and designed by Anders Hejlsberg, LINQ (Language Integrated Query) allows writing queries even without the knowledge of query languages like SQL, XML etc. LINQ queries can be written for diverse data types.

```
using System;
using System.Linq;

class Program
{
  public static void Main(string[] args)
  {
    string[] words = {"hello", "wonderful", "LINQ", "beautiful", "world"};
                //Get only short words
                var shortWords = from word in words where word.Length <= 5 select word;
                //Print each word out
                foreach (var word in shortWords)
                {
                        Console.WriteLine(word);
                }
                Console.ReadLine();
  }
}
```

Syntax of LINQ

There are two syntaxes of LINQ. These are the following ones.

- **Lamda (Method) Syntax**
  Example

  ```
  In C#:    var longWords = words.Where( w => w.length > 10);

  In VB.NET  Dim longWords = words.Where(Function(w) w.length > 10)
  ```

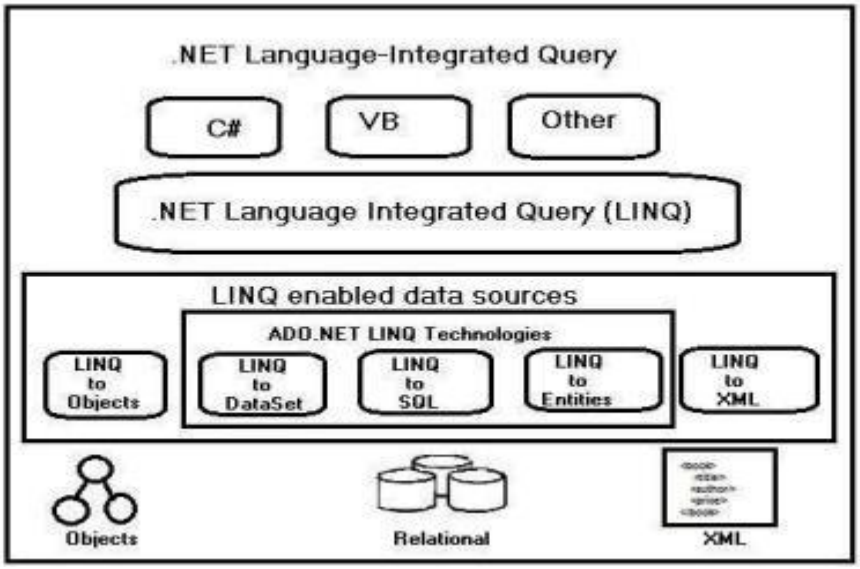
- **Query (Comprehension) Syntax**
  Example

  ```
  In C#:    var longwords = from w in words where w.length > 10;
  ```

Types of LINQ

LINQ to Entities
• LINQ to SQL (DLINQ)
• LINQ to DataSet •
LINQ to XML(XLINQ)
• LINQ to Objects

•The types of LINQ are mentioned below in brief. Apart from the above, there is also a LINQ type named PLINQ which is Microsoft's parallel LINQ. LINQ Architecture in .NET LINQ has a 3-layered architecture in which the uppermost layer consists of the language extensions and the bottom layer consists of data sources that are typically objects implementing IEnumerable or IQueryable generic interfaces. The architecture is shown below in Figure



LINQ offers a host of advantages and among them the foremost is its powerful expressiveness which enables developers Viewing relationship between two tables is easy with LINQ due to its hierarchical feature and this enables

- LINQ makes easy debugging due to its integration in the C# language.
- Writing codes is quite faster in LINQ and thus development time also gets reduced significantly.
- LINQ offers IntelliSense which means writing more accurate queries easily.
- LINQ offers syntax highlighting that proves helpful to find out mistakes during design time.
  to express declaratively.

- LINQ allows usage of a single LINQ syntax while querying many diverse data sources and this is mainly because composing queries joining multiple tables in less time.

- LINQ offers the facility of joining several data sources in a single query as well as breaking complex problems
- LINQ is extensible that means it is possible to use knowledge of LINQ to querying new data source types. of its unitive foundation.
- LINQ offers easy transformation for conversion of one data type to another like transforming SQL data to XMLinto a set of short queries easy to debug. data

PART III

5 **Describe with example :**

- **Array of objects**

  Array of objects in C# is just an array of object data as its value. By using array of objects, you can access the members (field and methods) of the class with each object.

Syntax:

```
class_name array_name[] = new class_name[SIZE];
```

Example:

```
using System;

namespace ExampleArrayOfObjects {
  // Class definition
  class Student {
    //private data members
    private int rollno;
    private string name;
    private int age;

    //method to set student details
    public void SetInfo(string name, int rollno, int age) {
      this.rollno = rollno;
      this.age = age;
      this.name = name;
    }

    //method to print student details
    public void printInfo() {
      Console.WriteLine("Student Record: ");
      Console.WriteLine("\tName     : " + name);
      Console.WriteLine("\tRollNo   : " + rollno);
      Console.WriteLine("\tAge      : " + age);
    }

  }

  class Program {
    static void Main() {
      //creating array of objects
      Student[] S = new Student[2];

      //Initialising objects by defaults/inbuilt
      //constructors
      S[0] = new Student();
      S[1] = new Student();

      //Setting the values and printing first object
      S[0].SetInfo("Herry", 101, 25);
      S[0].printInfo();

      //Setting the values and printing second object
      S[1].SetInfo("Potter", 102, 27);
      S[1].printInfo();

    }
  }
}
```

- **Partial class**

A partial class is a special feature of C#. It provides a special ability to implement the functionality of a single class into multiple files and all these files are combined into a single class file when the application is compiled. A partial class is created by using a ***partial*** keyword. This keyword is also useful to split the functionality of methods, interfaces, or structure into multiple files.

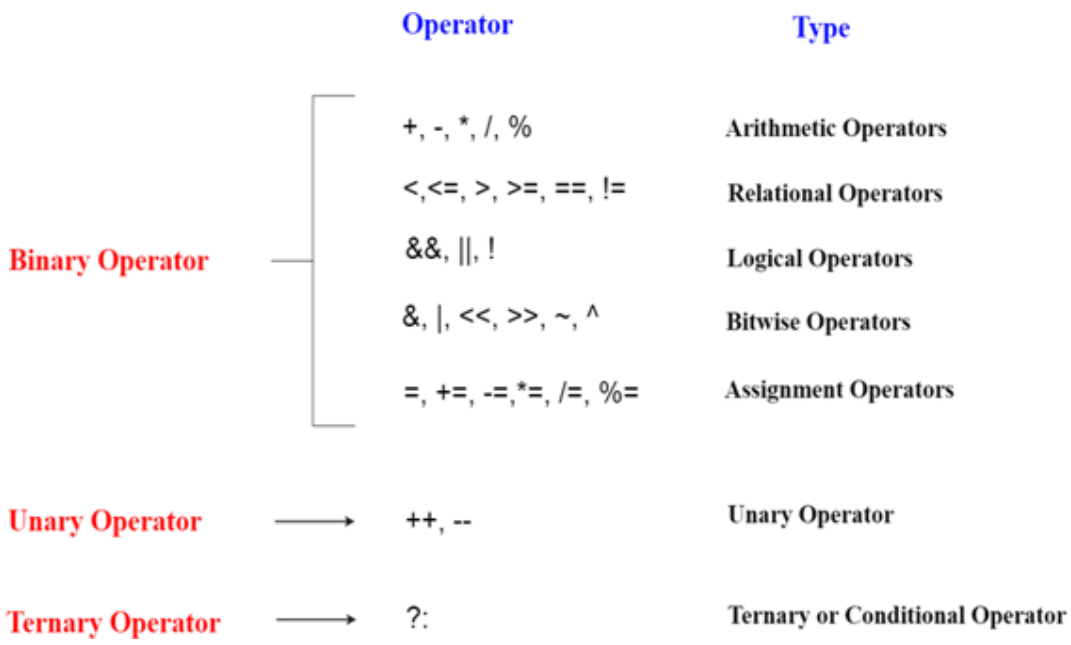
Syntax:

```
public partial Clas_name
{
        // code
}
```

```
public partial class Geeks {

    private string Author_name;

    private int Total_articles;



    public Geeks(string a, int t)

    {

        this.Authour_name = a;

        this.Total_articles = t;

    }

}
```

```
public partial class Geeks {
    public void Display()
    {
        Console.WriteLine("Author's name is : " + Author_name);
        Console.WriteLine("Total number articles is : " + Total_articles);
    }
}
public class Geeks {
    private string Author_name;
    private int Total_articles;

    public Geeks(string a, int t)
    {
        this.Authour_name = a;
        this.Total_articles = t;
    }

    public void Display()
    {
        Console.WriteLine("Author's name is : " + Author_name);
        Console.WriteLine("Total number articles is : " + Total_articles);
    }
}
```

6	**Explain different types of Operators in C #.** An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc. There are following types of operators to perform different types of operations in C# language. • Arithmetic Operators • Relational Operators • Logical Operators • Bitwise Operators • Assignment Operators • Unary Operators • Ternary Operators • Misc Operators



PART IV

7 **With the help of a C # program explain nested classes in C #.**

A nested class is a class which is declared in another enclosing class. A nested class is a member and as such has the same access rights as any other member. The members of an enclosing class have no special access to members of a nested class; the usual access rules shall be obeyed.

```
#include<iostream>

using namespace std;

/* start of Enclosing class declaration */
class Enclosing {
private:
    int x;

/* start of Nested class declaration */
class Nested {
    int y;
    void NestedFun(Enclosing *e) {
            cout<<e->x; // works fine: nested class can access
                                // private members of Enclosing class
    }
}; // declaration Nested class ends here
}; // declaration Enclosing class ends here

int main()
{

}

#include<iostream>

using namespace std;
```

```
/* start of Enclosing class declaration */
class Enclosing {

int x;

/* start of Nested class declaration */
class Nested {
     int y;
}; // declaration Nested class ends here

void EnclosingFun(Nested *n) {
               cout<<n->y; // Compiler Error: y is private in Nested
}
}; // declaration Enclosing class ends here

int main()
{

}
```

8	**Explain :**
	i. Boxing and Unboxing
	
	Boxing is the process of converting a value type to the object type or any interface type implemented by this value type. Boxing is implicit.
	
	Example: Boxing
	
	Copy
	
	int i = 10;
	
	object o = i; //performs boxing
	
	In the above example, the integer variable i is assigned to object o. Since object type is a reference type and base class of all the classes in C#, an int can be assigned to an object type. This process of converting int to object is called boxing.
	
	Let's look at a more practical example.
	
	Example: Boxing

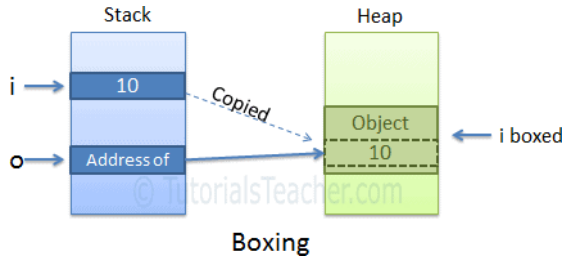
Copy

```
ArrayList list = new ArrayList();

list.Add(10); // boxing

list.Add("Bill");
```

Above, ArrayList is a class in C#, and so it is a reference type. We add an int value 10 in it. So, .NET will perform the boxing process here to assign value type to reference type.

As you know, all the reference types stored on heap where it contains the address of the value and value type is just an actual value stored on the stack. Now, as shown in the first example, int i is assigned to object o. Object o must be an address and not a value itself. So, the CLR boxes the value type by creating a new System.Object on the heap and wraps the value of i in it and then assigns an address of that object to o. So, because the CLR creates a box on the heap that stores the value, the whole process is called 'Boxing'.

The following figure illustrates the boxing process.



Unboxing is the reverse of boxing. It is the process of converting a reference type to value type. Unboxing extract the value from the reference type and assign it to a value type.

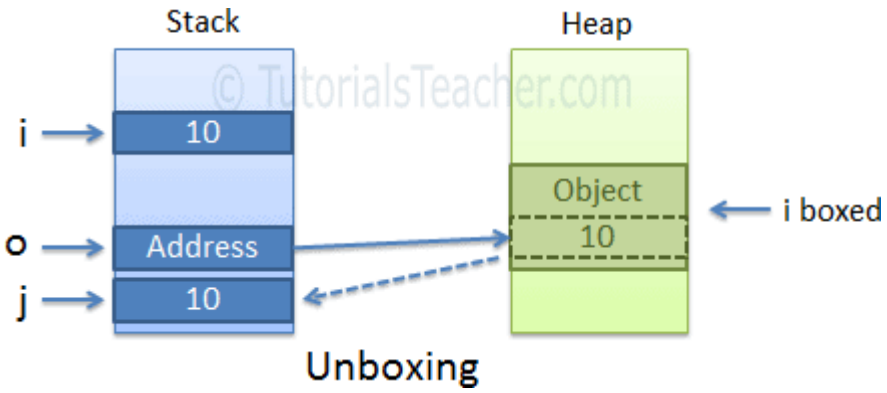
Unboxing is explicit. It means we have to cast explicitly.

Example: Unboxing

Copy

```
object o = 10;

int i = (int)o; //performs unboxing
```

The following figure illustrates the unboxing process.



A boxing conversion makes a copy of the value. So, changing the value of one variable will not impact others.

int i = 10;

object o = i; // boxing

o = 20;

Console.WriteLine(i); // output: 10

The casting of a boxed value is not permitted. The following will throw an exception.

Example: Invalid Conversion

Copy

int i = 10;

object o = i; // boxing

double d = (double)o; // runtime exception

First do unboxing and then do casting, as shown below.

Example: Valid Conversion

Copy

int i = 10;

object o = i; // boxing

double d = (double)(int)o; // valid

ii. Identifiers and keywords

An Identifiers is a sequence of character used to identify a variable, constant, or any user-defined programming element. Rules: Starts with a letter or an underscore and ends with a character. Can have letters, digits and underscores Must not be a reserved word Must be a complete word without any blank spaces. Example: sum, SUM, _sum Keywords are the reserved words whose meanings are predefined to the C# compiler. Rules: You cannot use keywords as variable, methods and properties If you want to use the keywords asidentifiers, prefix the keyword with @ character.

Table 2.1: Lists of Reserved keywords available in C#

Reserved Keywords								
abstract	as	base	bool	break	byte	case	this	object
Catch	char	checked	class	const	continue	decimal	is	sizeof
Default	delegate	do	double	else	enum	event	out	readonly
Extern	false	finally	fixed	float	for	if	param	using
foreach	goto	int	public	protected	private	short	namespace	unchecked

iii. Foreach

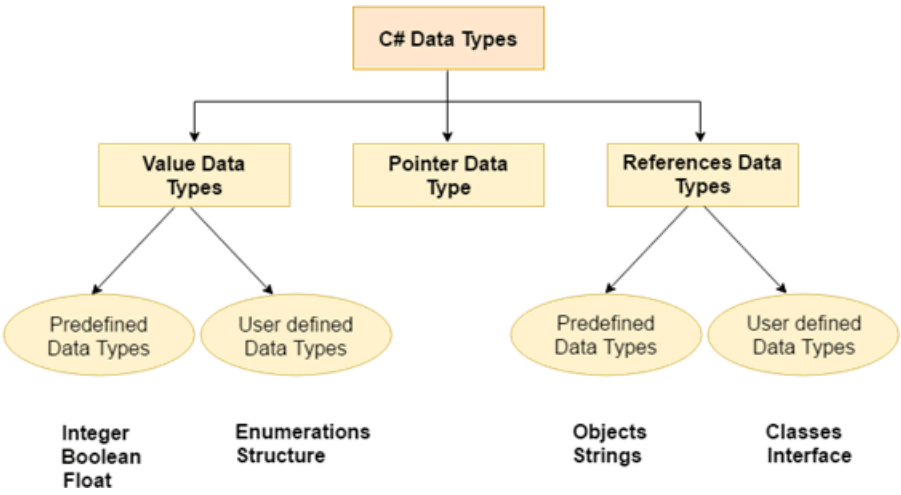
- It is necessary to enclose the statements of foreach loop in curly braces {}.
- Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.
- In the loop body, you can use the loop variable you created rather than using an indexed array

element.

```
using System;

class GFG {

    // Main Method
    static public void Main()
    {

        Console.WriteLine("Print array:");

        // creating an array
        int[] a_array = new int[] { 1, 2, 3, 4, 5, 6, 7 };

        // foreach loop begin
        // it will run till the
        // last element of the array
        foreach(int items in a_array)
        {
            Console.WriteLine(items);
        }
    }
}
```

PARTV

9 **Explain different data types in C #.**



Value types: Value types directly contain their data, and instances of value types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.

Reference types Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self-describing types, pointer types, or interface types. The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types, and

delegates.

10 **What are namespaces? List and explain the purpose of any five namespaces**

Within System we can find numerous useful types dealing with built in data, mathematical computations, random number generation, environment variables, and garbage collection, as well as a number of commonly used exceptions and attributes. So System is a root namespace. The following are some of the common namespaces provided by the .NET Framework class library

System.Drawing System.Drawing.Drawing2D System.Drawing.Printing	Here, you find numerous types wrapping graphical primitives such as bitmaps, fonts, and icons, as well as printing capabilities.
System.IO System.IO.Compression System.IO.Ports	Include file I/O, buffering, and so forth. As of .NET 2.0, the IO namespaces now include support compression and port manipulation.
System.Net	Contains types related to network programming (requests/responses, sockets, end points, and so on).
System.Reflection System.Reflection.Emit	Define types that support runtime type discovery as well as dynamic creation of types.
System.Runtime.InteropServices	Provides facilities to allow .NET types to interact with "unmanaged code" (e.g., C-based DLLs and COM servers) and vice versa.
System.Runtime.Remoting	Defines types used to build solutions that incorporate the .NET remoting layer.
System.Security	Security is an integrated aspect of the .NET universe. In the security-centric namespaces you find numerous types dealing with permissions, cryptography, and so on.
System.Threading	This namespace defines types used to build multithreaded applications.
System.Web System.Web.Security	A number of namespaces are specifically geared toward the development of .NET web applications, including ASP.NET and XML web services.
System.Windows.Forms	Contains types that facilitate the construction of traditional desktop GUI applications.
System.Xml	The XML-centric namespaces contain numerous types used to interact with XML data.