Agenda

- System Namespace
- Configuration
- Globalization
- String Handling
- IO Operations
- System.Reflection
- System.Text

System Namespace

- The System namespace contains fundamental classes and base classes that define commonly used:
  - Value and reference data types.
  - Events and event handlers.
  - Interfaces.
  - Attributes.
  - Processing exceptions.
- System namespace contains:
  - Classes that provide services supporting data type conversion.
  - Method parameter manipulation.
  - Mathematics.
  - Remote and local program invocation.
  - Application environment management.
  - Supervision of managed and unmanaged applications.

System.Configuration

- The **System.Configuration** namespace provides classes and interfaces that allow you to programmatically access .NET Framework configuration settings and handle errors in configuration files (.config files).
- The classes in System.Configuration:
  - Provide a method for reading values of a particular type from the .config file.
  - Provide access to configuration settings in a specified configuration section.
  - Provide a section handler definition for configuration sections read and handled by systems other than System.Configuration.

System.Configuration (cont.)

- System.Configuration hierarchy:

```
System.Object
  System.Configuration.AppSettingsReader
  System.Configuration.ConfigurationSettings
  System.Configuration.DictionarySectionHandler ---- System.Configuration.IConfigurationSectionHandler
  System.Configuration.IgnoreSectionHandler ---- System.Configuration.IConfigurationSectionHandler
  System.Configuration.NameValueFileSectionHandler ---- System.Configuration.IConfigurationSectionHandler
  System.Configuration.NameValueSectionHandler ---- System.Configuration.IConfigurationSectionHandler
  System.Configuration.SingleTagSectionHandler ---- System.Configuration.IConfigurationSectionHandler
  System.Exception
    System.SystemException
      System.Configuration.ConfigurationException
  System.Xml.XmlNode
    System.Xml.XmlDocument
      System.Configuration.ConfigXmlDocument
```

- Note that System.Object is the class from which all others are derived.

System.Globalization

- The **System.Globalization** namespace contains classes that define culture-related information.
- It handles the following culture information:
  - Language
  - Country / Region
  - Calendars in use
  - Format pattern for dates
  - Currency
  - Numbers
  - Sort order for strings

System.Globalization (cont.)

- System.Globalization contains the following major classes:

Class	Description
Calendar	Represents time in divisions, such as weeks, months, and years.
CultureInfo	Implements a set of methods for culture-sensitive string comparisons.
DateTimeFormatInfo	Defines how DateTime values are formatted and displayed, depending on the culture.
DaylightTime	Defines the period of daylight-saving time.
GregorianCalendar	Represents the Gregorian calendar.
HebrewCalendar	Represents the Hebrew calendar.
JapaneseCalendar	Represents the Japanese calendar.
NumberFormatInfo	Defines how numeric values are formatted and displayed, depending on the culture.
RegionInfo	Contains information about the country/region.
SortKey	Represents the result of mapping a string to its sort key.
TextInfo	Defines properties and behaviors, such as casing, that are specific to a writing system.

String Handling – System.String

- A string is a sequential collection of Unicode characters, typically used to represent text, while a String is a sequential collection of System.Char objects that represents a string.
- A String is called immutable because its value cannot be modified once it has been created. Methods that appear to modify a String actually return a new String containing the modification.
- Casing rules determine how to change a Unicode character between one case and another.
- Formatting rules determine how to convert a value to its string representation, while parsing rules determine how to convert a string representation to a value.
- Sort rules determine the alphabetic order of Unicode characters and how two strings are compared to each other.
- Any string, including the empty string ("""), is greater than a null reference. Two null references are equal to each other.

String Handling – System.String (cont.)

- System.String has many useful methods:
  - Compare, CompareOrdinal, CompareTo, Equals, EndsWith, and StartsWith:
    - Used for comparisons.
  - IndexOf, IndexOfAny, LastIndexOf, and LastIndexOfAny:
    - Used to obtain the index of a substring or Unicode character in a string.
  - Copy and CopyTo:
    - Use to copy a string or substring to another string or an array of Char.
  - Substring and Split:
    - Used to create one or more new strings from portions of an original string.
  - Concat and Join:
    - Used to create a new string from one or more substrings.
  - Insert, Replace, Remove, PadLeft, PadRight, Trim, TrimEnd, and TrimStart:
    - Used to modify all or part of a string.
  - ToLower and ToUpper:
    - Used to change the case of Unicode characters in a string.
  - Format:
    - Used to replace one or more placeholders in a string with the string representation of one or more values.

String Handling

```
using System;

namespace sample
{
   public class String
   {
      static void Main()
      {
         string date = DateTime.Today.ToShortDateString();

         // Use the + and += operators for one-time concatenations.
         string str = "Hello ! Today date is " + date + ".";
         Console.WriteLine(str);

         str += " How are you today?";
         Console.WriteLine(str);

         // Remove a substring from the middle of the string.
         string subString = "date";
         int i = str.IndexOf(subString);
         if (i >= 0)
         {
            str = str.Remove(i, subString.Length);
         }
         Console.WriteLine(str);

         //To get the substring between two strings
         int firstIndex = str.IndexOf("!") + "!".Length;
         int lastIndex = str.LastIndexOf(".");
         str = str.Substring(firstIndex, lastIndex - firstIndex);
         Console.WriteLine("Substring between ! and . is '{0}'", str);

      }
   }
}
```

Concatenating strings

Removing substring

To get substring

```
Hello ! Today date is 4/22/2008.
Hello ! Today date is 4/22/2008. How are you today?
Hello ! Today  is 4/22/2008. How are you today?
Substring between ! and . is ' Today  is 4/22/2008'
```

IO Operations – System.IO

- System.IO namespace contains types that allow synchronous and asynchronous reading and writing on data streams and files.
- A file is an ordered and named collection of a particular sequence of bytes having persistent storage.
  - With regard to files, think in terms of directory paths, disk storage, and file and directory names.
  - In contrast, streams provide a way to write and read bytes to and from a backing store that can be one of several storage mediums.
  - There are several kinds of streams other than file streams namely, network, memory, and tape streams.

IO Operations – System.IO (cont.)

- Stream class integrates asynchronous support. Its default implementations define synchronous reads and writes in terms of their corresponding asynchronous methods, and vice versa.
- Streams involve these fundamental operations:
  - Streams can be read from - Reading is the transfer of data from a stream into a data structure, such as an array of bytes.
  - Streams can be written to - Writing is the transfer of data from a data source into a stream.
  - Streams can support seeking - Seeking is the querying and modifying of the current position within a stream.

System.IO namespace

- The following classes are used for File I/O:
  - FileObject – Provides a representation of a text file. Use the FileObject class to perform most typical text file operations such as reading, writing, appending, copying, deleting, moving, or renaming.
  - Directory – Provides static methods for creating, moving, and enumerating through directories and subdirectories.
  - DirectoryInfo – Provides instance methods for creating, moving, and enumerating through directories and subdirectories.
  - File – Provides static methods for the creation, copying, deletion, moving, and opening of files; and aids in the creation of a FileStream.
  - FileInfo – Provides instance methods for the creation, copying, deletion, moving, and opening of files; and aids in the creation of a FileStream.
  - FileStream – Supports random access to files through its Seek method. FileStream opens files synchronously by default, but supports asynchronous operation as well.
  - FileSystemInfo – Abstract base class for FileInfo and DirectoryInfo.
  - Path – Provides methods and properties for processing directory strings in a cross-platform manner.

System.IO

- The following classes are used for reading from and writing to streams:
  - BinaryReader and BinaryWriter – Use to read and write encoded strings and primitive data types from and to streams.
  - StreamReader – Reads characters from streams, using encoding to convert characters to and from bytes.
  - StreamWriter – Writes characters to streams, using encoding to convert characters to bytes.
  - StringReader – Reads characters from strings. StringReader allows you to treat strings with the same API, so your output can be either a stream in any encoding or a string.
  - StringWrite – Writes characters to strings.
  - TextReader – Abstract base class for StreamReader and StringReader.
  - TextWriter – Abstract base class for StreamWriter and StringWriter.

System.IO (cont.)

```
using System;
using System.IO;

namespace sample
{
   public class SampleFile
   {
      public static void Main()
      {
         // Create a text file C:\ATS\SampleFile.txt
         using (FileStream fs = new FileStream(@"C:\ATS\SampleFile.txt ", FileMode.OpenOrCreate, FileAccess.ReadWrite))
         {
            StreamWriter m_streamWriter = new StreamWriter(fs);
            // Write to the file using StreamWriter class
            m_streamWriter.BaseStream.Seek(0, SeekOrigin.End);
            m_streamWriter.Write(" File Write Operation Starts : ");
            m_streamWriter.WriteLine("{0} {1}", DateTime.Now.ToLongTimeString(), DateTime.Now.ToLongDateString());
            m_streamWriter.WriteLine(" Welcome to the sample file.\n");
            m_streamWriter.WriteLine(" This is next line in the text file.\n ");
            m_streamWriter.Flush();
            // Read from the file using StreamReader class
            StreamReader m_streamReader = new StreamReader(fs);
            string str = m_streamReader.ReadLine();

         }
      }
   }
}
```

Used FileStream to create or open a new file and used "using" block to open and close file.

Used StreamWriter to write text to text file.

Used StreamReader to read text from text file.

System.Reflection

- The following classes and interfaces are used to provide a managed view of loaded types, methods, and fields; with the ability to dynamically create and invoke types:
  - Assembly - A reusable, versionable, and self-describing building block of a CLR application.
  - AssemblyName - Specifies the assembly's unique identity in full.
  - AssemblyCultureAttribute - Specifies the culture supported by assembly.
  - MemberInfo - Specifies the attributes of a member and provides access to member’s metadata.
  - ICustomAttributeProvider - Provides custom attributes for reflection objects that support them.
  - IReflect - Allows objects to return MemberInfo objects that represent an object.

System.Reflection (cont.)

```
using System;
using System.Reflection;
namespace sample
{
class MainClass
   {
      public static void Main()
      {
         System.Type type = typeof(SampleReflection);

         MethodInfo[] methods = type.GetMethods();
         PropertyInfo[] properties = type.GetProperties();
         FieldInfo[] fields = type.GetFields();
         Console.WriteLine("SampleReflection class has following Methods");
         foreach (MethodInfo method in methods)
         {
            Console.WriteLine(method.Name);
         }
         Console.WriteLine("\nSampleReflection class has following Properties");
         foreach (PropertyInfo property in properties)
         {
            Console.WriteLine(property.Name);
         }
         Console.WriteLine("\nSampleReflection class has following fields");
         foreach (MemberInfo field in fields)
         {
            Console.WriteLine(field.Name);
         }
         Console.ReadLine();

      }
   }
  public class SampleReflection
   {
      public int age;
      public string name;
      public int Age
      {
         get
         {
            return age;
         }
         set
         {
            age = value;
         }
      }
      public string Name
         {
            get
            {
               return name;
            }
            set
            {
               name = value;
            }
         }
         public string GetName()
         {
            return name;
         }
         public int GetAge()
         {
            return age;
         }
      }
   }
```

```
SampleReflection class has following Methods
get_Age
set_Age
get_Name
set_Name
GetName
GetAge
GetType
ToString
Equals
GetHashCode

SampleReflection class has following Properties
Age
Name

SampleReflection class has following fields
age
name
```

System.Text

- System.Text namespace contains:
  - Classes representing ASCII, Unicode, UTF-7, and UTF-8 character encodings.
  - Abstract base classes for converting blocks of characters to and from blocks of bytes.
  - A helper class (StringBuilder) that manipulates and formats string objects without creating intermediate instances of string. This class is particularly useful when a lot of string manipulations are done (as StringBuilder does not create a new string each time an insert / append / delete operation is performed on the string).
- System.Text hierarchy:

```
System.Object
  System.Text.Decoder
  System.Text.Encoder
  System.Text.Encoding
    System.Text.ASCIIEncoding
    System.Text.UnicodeEncoding
    System.Text.UTF7Encoding
    System.Text.UTF8Encoding
  System.Text.StringBuilder
```

System.Text (cont.)

```
using System;
using System.Text;

namespace sample
{
   public class SampleFile
   {
      public static void Main()
      {
         StringBuilder builder = new StringBuilder("Hello Welcome to ATS School!", 30);
         int cap = builder.EnsureCapacity(55);
         builder.Append(" This is a sample test.");
         Console.WriteLine(builder.ToString());
         builder.Insert(28, " String Builder");
         Console.WriteLine(builder.ToString());
         builder.Remove(6, 21);
         Console.WriteLine(builder.ToString());
         builder.Replace('!', '?');
         Console.WriteLine(builder.ToString());
         Console.WriteLine("Length of string is:" + builder.Length.ToString());
         Console.WriteLine("Capacity of string is:" + builder.Capacity.ToString());
      }
   }
}
```

Used StringBuilder class to append, add, replace and remove characters.

```
Hello Welcome to ATS School! This is a sample test.
Hello Welcome to ATS School! String Builder This is a sample test.
Hello ! String Builder This is a sample test.
Hello ? String Builder This is a sample test.
Length of string is:45
Capacity of string is:120
```

System.Math (1 of 4)

- System.Math class provides constants and static methods for trigonometric, logarithmic, and other common mathematical functions.
- Math class cannot be extended (as it is declared sealed).
- Its methods are declared static and can be invoked using its class name.
- System.Math hierarchy:

```
System.Object
  System.Math
```

System.Math (2 of 4)

- System.Math Method list:

Method(s)	Description
Acos, Asin, Atan, Atan2, Cos, Cosh, Sin, Sinh, Tan, Tanh	Trigonometric functions.
Floor, Ceiling, Round	Functions to round the number.
Log, Log10	Logarithmic functions.
Max, Min	Use to find the maximum and minimum of the numbers.
DivRem, IEEERemainder	Use to find the results of division of numbers.
Abs	Returns the absolute value of a specified number.
BigMul	Produces the full product of two 32-bit numbers.
Exp	Returns **e** raised to the specified power.
Pow	Returns a specified number raised to the specified power.
Sign	Returns a value indicating the sign of a number.
Sqrt	Returns the square root of a specified number.

System.Math (3 of 4)

```
using System;
class MathSample
{
    private double length;
    private double breadth;
    private double height;
    private double radius;

    public MathSample(double num1, double num2 , double num3, double num4)
    {
        length = Math.Abs(num1);
        breadth = Math.Abs(num2);
        height = Math.Abs(num3);
        radius = Math.Abs(num4);
    }

    private double GetRectangleArea()
    {
        return (length * breadth * height);
    }

    public double GetCircleArea()
    {
        return (3.14 * Math.Pow(radius, 2.0));
    }
```

Returns the absolute Value of number.

Returns a specified number raised to the specified power.

System.Math (4 of 4)

```
public double GetSine()
   {
      return Math.Round(Math.Sin(height),2);
   }

public double GetSquareRoot()
   {
      return  Math.Round(Math.Sqrt(radius),2);
   }

public static void Main()
   {
      MathSample sample = new MathSample(10.0, 20.0, 45, 40.0);
      Console.WriteLine("The length = 10, breadth = 20, height = 45 and radius = 40");
      double rectangleArea = sample.GetRectangleArea();
      Console.WriteLine("Rectangle Area is: " + rectangleArea.ToString());
      double circleArea = sample.GetCircleArea();
      Console.WriteLine("Circle's Area is: " + circleArea.ToString());
      double sine = sample.GetSine();
      Console.WriteLine("Sine of 45 is: " + sine.ToString());
      double squareRoot = sample.GetSquareRoot();
      Console.WriteLine("Square Root of 40 is: " + squareRoot.ToString());

   }
}
```

Returns the sine of the specified value.

Returns the square root of a specified number.

```
The length = 10, breadth = 20, height = 45 and radius = 40
Rectangle Area is: 9000
Circle's Area is: 5024
Sine of 45 is: 0.85
Square Root of 40 is: 6.32
```