File Handling and Serialization

Objectives

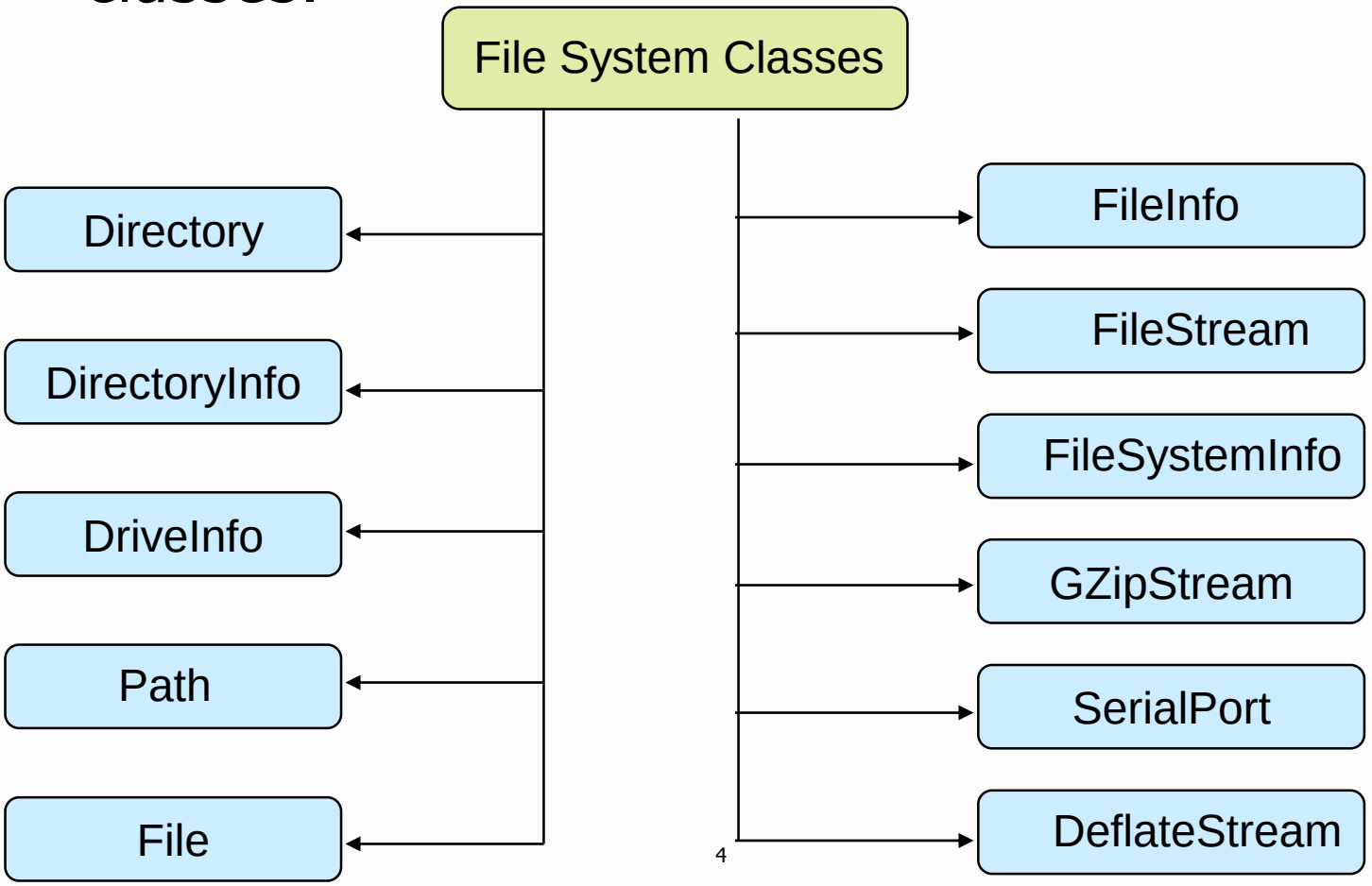
- After completing this session, you will be able to:
  - Access files paths by using the `Path` class
  - Access files,directories,drives by using classes available in System.IO
  - Define Streaming
  - Define Serialization and Deserialization
  - Serialize and Deserialize object by using BinaryFormatter and SoapFormatter classes.
  - Serialize and Deserialize objects by using XmlSerializer class.

Input and Output Operations in C#

- I/O in C# is stream based.
- Stream is flow of data from a source to a receiver through a channel.
- Two types of Streams
  - Byte Streams.
  - Character Streams.
- Three predefined streams are
  - `Console.Out`
  - `Console.In`
  - `Console.Error`

`System.IO` Namespace

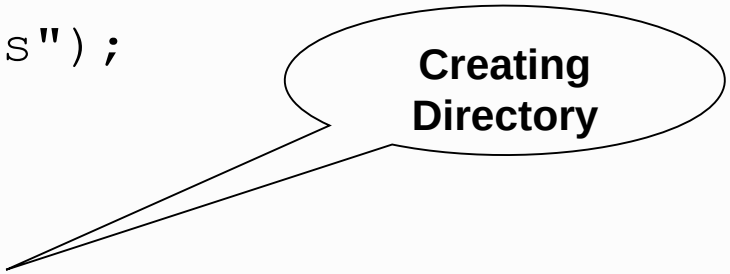
- File IO using System.IO:
  System.IO namespace defines all the stream classes.



`Directory` Class

- Static class and helps to manage a single directory.

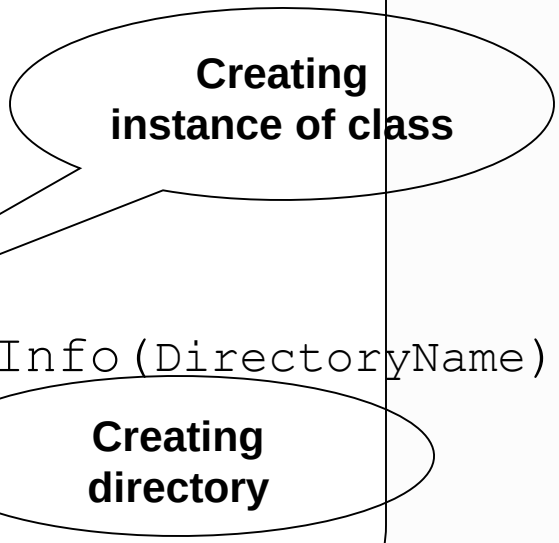
```
string DirectoryName = @"C:\MyDirectory";
  if (Directory.Exists(DirectoryName))
    {
      MessageBox.Show("Exists");
    }
  else
    {
      Directory.CreateDirectory (DirectoryName);
      MessageBox.Show("Created");
    }
```



`DirectoryInfo` Class

- Extends the `FileSystemInfo` class.
- Performing operations such as copying, moving, renaming, creating, and deleting directories.

```
string DirectoryName = @"C:\MyDirectory";
  if (Directory.Exists(DirectoryName))
    {
      MessageBox.Show("Exists");
    }
  else
    {
      DirectoryInfo  d = new DirectoryInfo(DirectoryName);
      d.Create ();
      MessageBox.Show("Created");
}
```



`DriveInfo` and `Path` Class

- `DriveInfo` class
  - Models a drive and gives its information.

```
string s=@"C:\";
DriveInfo d = new DriveInfo(s);
MessageBox.Show(d.AvailableFreeSpace.ToString()
                ,d.Name );
```

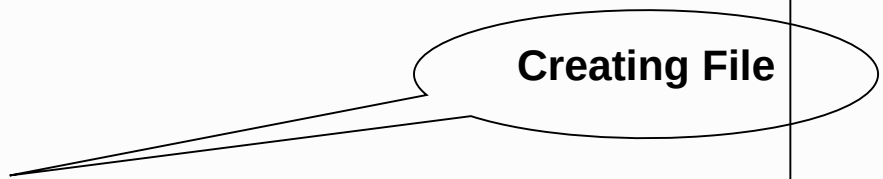
- `Path` class
  - Used to manage file and directory paths.

```
string s = @"C:\temp\MyData.text\machine.config”;
MessageBox.Show (Path.GetFileName(s));
MessageBox.Show(Path.GetTempPath());
```

`File` Class

- Static class and helps to manage a single file

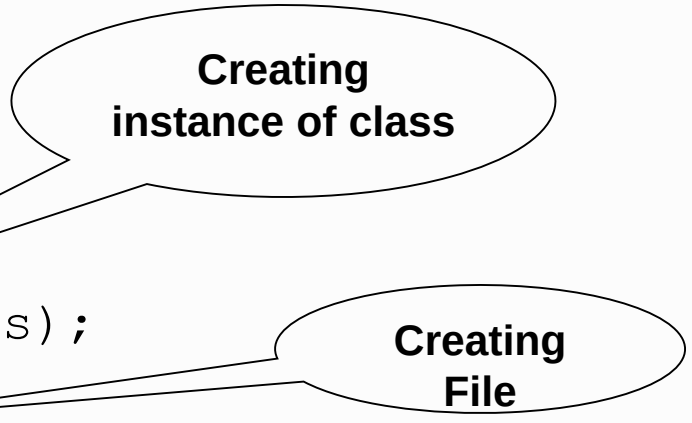
```
string FileName = @"C:\MyFile.dat";
    if (File.Exists(FileName))
          {
              MessageBox.Show("Exists");
          }
    else
          {
              File.Create(FileName);
              MessageBox.Show("Created");
          }
```



`FileInfo` Class

- Defines instance methods to perform file operations.

```
string FileName = @"C:\MyFile.dat";
if (File.Exists(FileName))
{
      MessageBox.Show("Exists");
}
else
{
      FileInfo f = new FileInfo(s);
      f.Create();
      MessageBox.Show("Created");
}
```

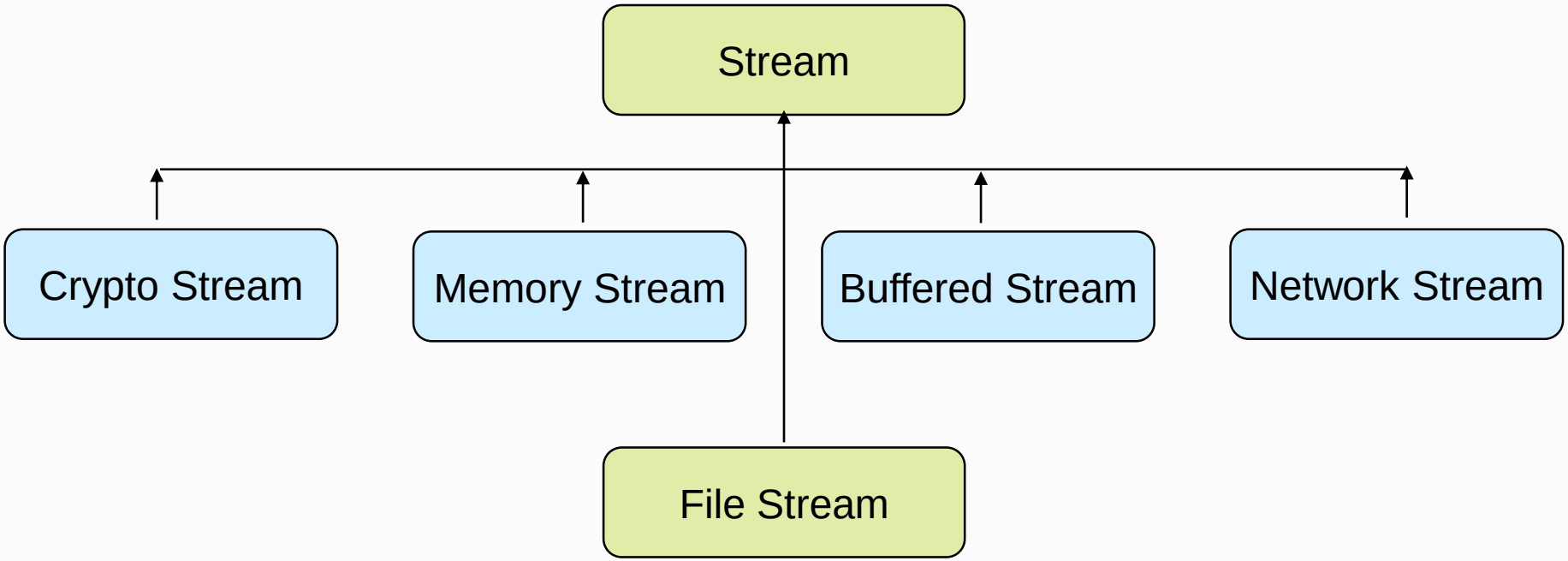


Other File System Classes

- `DeflateStream`
- `GZipStream`
- `SerialPort`
- `FileSystemInfo`

`Stream` class

- Stream is an abstract base class for all other stream classes
- Common I/O Stream Classes



`FileStream` class

- Used to perform operations like read, write, open, and close operations on files on a file system
  - For better performance, `FileStream` class buffers input and output.

BinaryReader & BinaryWriter Class

- BinaryReader class is used to read data in binary files.
- BinaryWriter class is used to write data to binary files.

```
FileStream fs = new FileStream(@"C:\FileIO.txt",
                FileMode.Create,FileAccess.Write);
  BinaryWriter bw = new BinaryWriter(fs);
  bw.Write("Microsoft");
  bw.Close();
  fs.Close();
```

Writing into File

```
FileStream fs = new FileStream(@" C:\FileIO.txt",
                 FileMode.Open,FileAccess.Read );
  BinaryReader br = new BinaryReader(fs);
  MessageBox.Show( br.ReadString().ToString());
  br.Close();
  fs.Close();
```

Reading from File

The Character Stream Wrapper Classes

- `TextReader`
  - Abstract class to read sequential series of charaters
  - Inherited by
    - `StreamReader`
    - `StringReader`
- `TextWriter`
  - Abstract class to write sequential series of charaters
  - Inherited by
    - `StreamWriter`
    - `StringWriter`

StreamReader & StreamWriter Class

- StreamReader
  - Used to read data from stream.

```
StreamReader sr = new StreamReader(@"c:\ FileIO.txt ");
 MessageBox.Show(sr.ReadToEnd().ToString());
sr.Close();
```

Reading from File

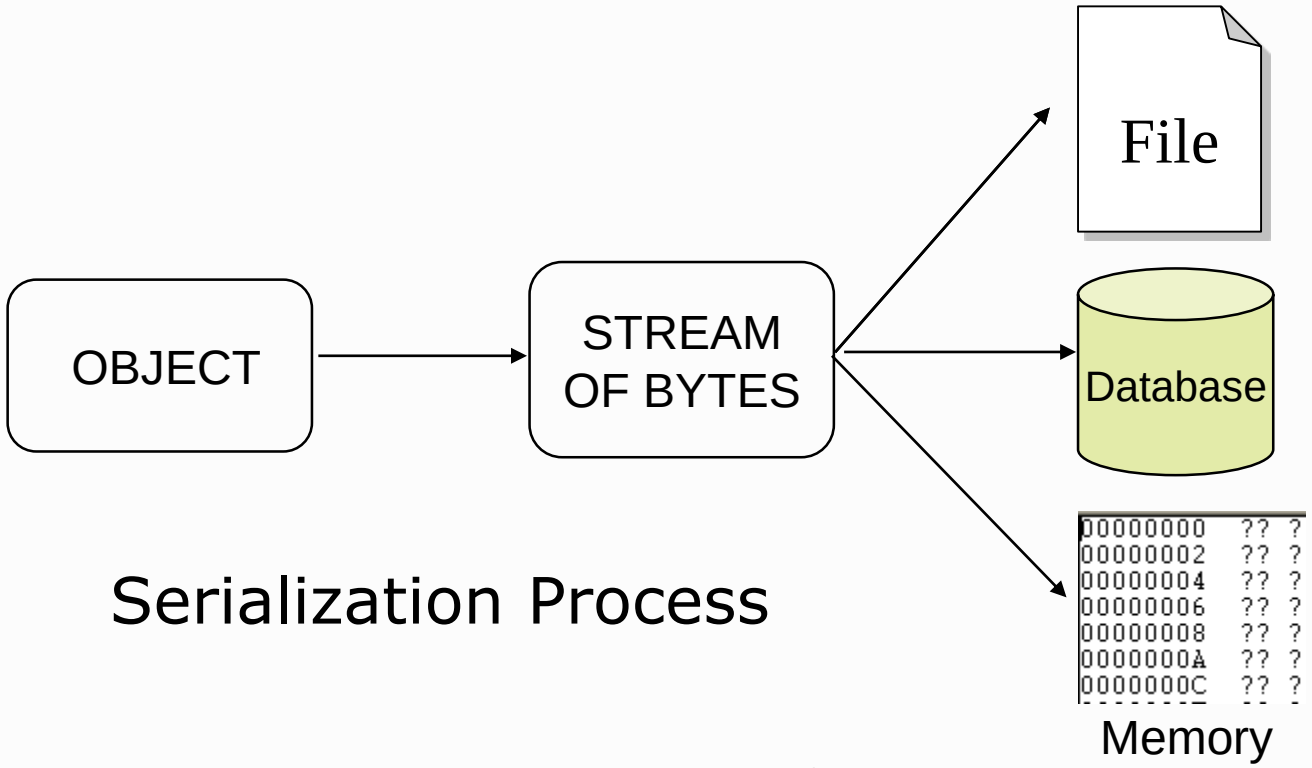
- StreamWriter
  - Used to write data to a stream.

```
StreamWriter sw = new StreamWriter(@"c:\FileIO.txt ");
          sw.Write("Microsoft");
          sw.Close();
```

Writing into File

Serialization

- Persistence is mapped by using serialization.
- Serialization is a process of converting data into portable format.



Serialization Process

Types of Serialization

- Binary Serialization
  - `System.Runtime.Serialization.Formatters.Binary`
- XML Serialization
  - `System.Xml.Serialization`
- SOAP Serialization
  - `System.Runtime.Serialization.Formatters.Soap`

[NonSerialized] attribute

- Indicates that a field of a serializable class should not be serialized
- To reduce the size of the serialized object add the [NonSerialized] attribute to the member
  - For example

```
[NonSerialized] public string Name;
```

Quick Recap…

- Files,directories,drives can be accessed by using classes defined in `System.IO`.
- `Stream` is the abstract base class of all streams
- BinaryReader & BinaryWriter and StreamReader & StreamWriter are used to read and write data in file.
- Serialization is the process of serializing and deserializing objects so that they can be stored and then later re-created.