C# File Handling

In this lab we will see the support from the C# to Files. Files handling is the need base programming feature we need to use to store the data which need not to store in Database server. Such data can be the exception message or log entry etc. If you are developing an application and you want to have offline store for the data to be entered into the application then File can be used.

System.IO is the namespace provided by Microsoft for Disk IO operations.

Exercise 1: Creating, Writing, Writing and Deleting file.

In this exercise we will see how to perform various IO operations on the file on the Disk. In the exercise following classes are used:

- FileStream
  - Used for managing files on disk as stream e.g. Creating, Opening etc.
- StreamWriter
  - User to write data in file.
- StreamReader
  - Used to Read data from the file.
- File
  - Used to perform operations on file like, delete, move, copy, append etc.

Task 1: Open VS2010 and create a C# console application, name it as 'CS_FileIO'. Use the following namespace in the Program.cs.

```
using System.IO;
```

Task 2: In the Program.cs add a new class, name it as 'FileOperation'. This class will contain operations for creating, writing, reading and deleting on file. The code of the class is as below:

```
public class FileOperation
  {
    /// <summary>
    /// This method creates a new File using FileStream class
    /// The StreamWriter class Detects the File created using FileCreate and Write the data into it
    /// </summary>
    /// <param name="FileName"></param>
    /// <param name="FileContents"></param>
    public void CreateAndWriteFile(string FileName,string FileContents)
    {
      FileStream Fs = new FileStream(FileName,FileMode.CreateNew);
      StreamWriter Sw = new StreamWriter(Fs);
      Sw.WriteLine(FileContents);
      Sw.Close();
      Sw.Dispose();
      Fs.Close();
```

```
            Fs.Dispose();
        }

        /// <summary>
        /// This method willopen the file using FileStream class.
        /// The contents of the file will be read using StreamWriter class
        /// </summary>
        /// <param name="FileName"></param>
        public void ReadFileData(string FileName)
        {
            FileStream Fs = new FileStream(FileName, FileMode.Open);
            StreamReader Sr = new StreamReader(Fs);
            string FileContents = Sr.ReadToEnd();
            Console.WriteLine("File Contents are:");
            Console.WriteLine("|=================================|");
            Console.WriteLine(FileContents);
            Console.WriteLine("|=================================|");
            Console.WriteLine("Ends Here");
            Fs.Close();
            Fs.Dispose();
        }

        /// <summary>
        /// This method will append the data in the existing file
        /// </summary>
        /// <param name="FileName"></param>
        /// <param name="FileContents"></param>
        public void AppendFile(string FileName, string FileContents)
        {
            File.AppendAllText(FileName, FileContents);
        }


        /// <summary>
        /// This method will delete the file of sepcific Name
        /// The 'File' class is used to perform Operation
        /// </summary>
        /// <param name="FileName"></param>
        public void DeleteFile(string FileName)
        {
            File.Delete(FileName);
            Console.WriteLine("The file " + FileName + "is successfully deleted");
        }
    }
```

The above class define methods as below:

- CreateAndWriteFile:
    - This method accept two parameters, FileName and FileContents. This method will create a file on the disk and write contents in it.
- ReadFileData:
    - This method accept filename to read data from it.

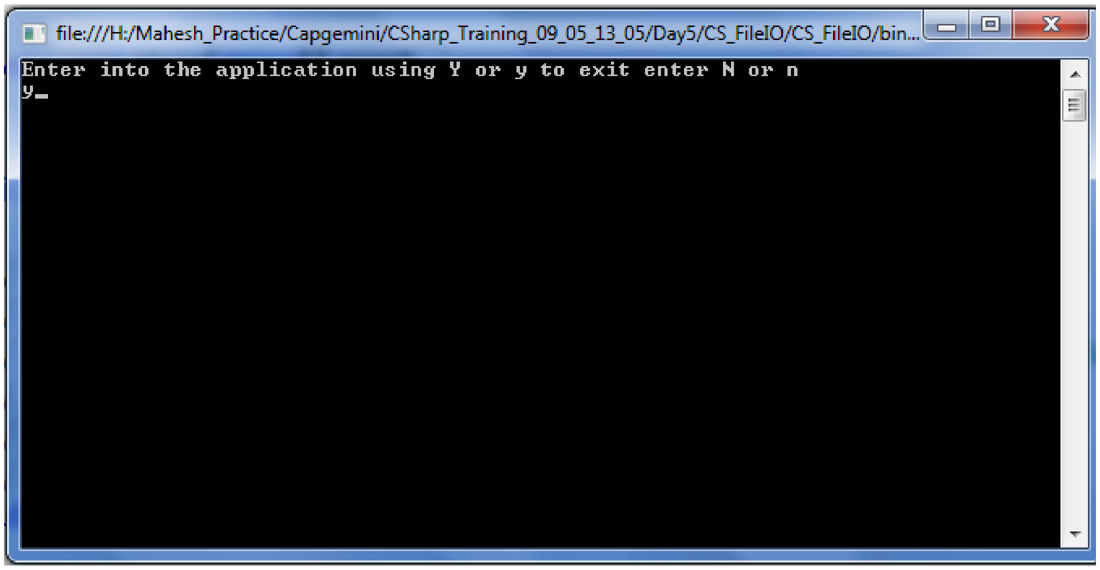
- AppendFile:
    - This method accepts FileName and FileContents as input parameter so that the file can be open for append and the contents can be appended in it.
- DeleteFile
    - This method will accept the file name to be deleted.

Task 3: In the main method of the program.cs write the following code. This code contains a menu driven programming for performing various file operations.

```
static void Main(string[] args)
  {
     Console.WriteLine("Enter into the application using Y or y to exit enter N or n");
     string Selection = Console.ReadLine();
     while ((Selection == "Y") || (Selection=="y"))
     {
        Console.WriteLine("Select Operation to be performed");

        Console.WriteLine("1: To Create and Write the file");
        Console.WriteLine("2: To Read Data from File");
        Console.WriteLine("3: Append into File");
        Console.WriteLine("4: To Delete File");

        int Choice = Convert.ToInt32(Console.ReadLine());
        FileOperation objFileOp = new FileOperation();
        switch (Choice)
        {
           case 1:
              Console.WriteLine("Enter the file path, file name to Create");
              string fileName = Console.ReadLine();
              Console.WriteLine("Enter Contents to be written in the file");
              string fileContents = Console.ReadLine();
              objFileOp.CreateAndWriteFile(@fileName, fileContents);
              break;
           case 2:
              Console.WriteLine("Enter the file path, file name to Read");
              string fileNameToRead = Console.ReadLine();
              objFileOp.ReadFileData(@fileNameToRead);
              break;
           case 3:
               Console.WriteLine("Enter the file path, file name to Append");
              string fileNameToAppend = Console.ReadLine();
               Console.WriteLine("Enter Contents to file to append data");
              string fileContentToAppend = Console.ReadLine();
              objFileOp.AppendFile(@fileNameToAppend, fileContentToAppend);
              break;
           case 4:
              Console.WriteLine("Enter the file path, file name to Delete");
              string fileNameToDelete = Console.ReadLine();
              objFileOp.DeleteFile(@fileNameToDelete);
              break;
        }
```

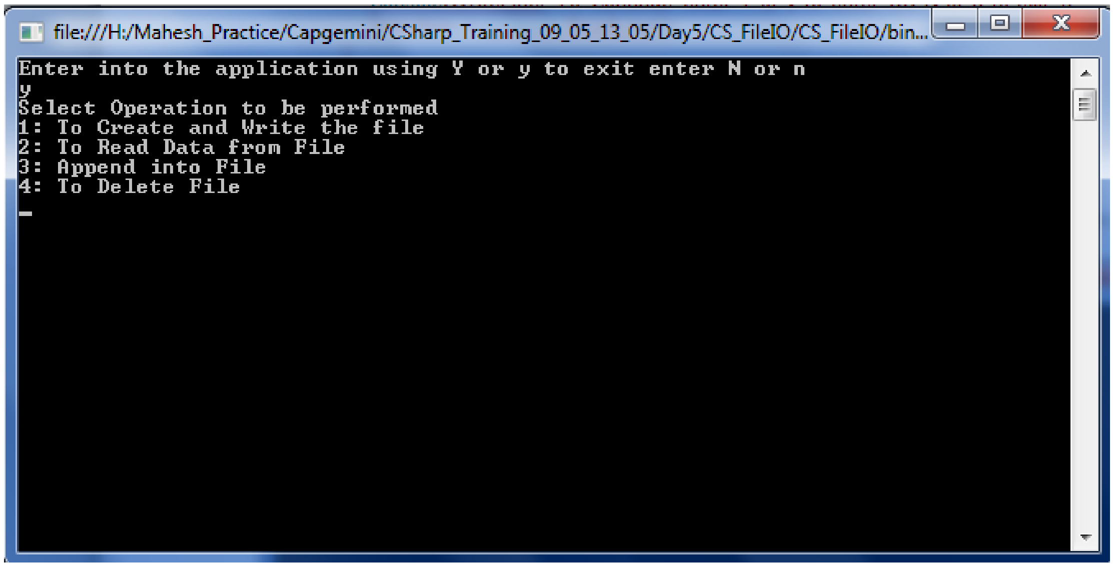
```
                Console.WriteLine("To Continue enter Y or y to enter OR N or n to exit");
                Selection = Console.ReadLine();
            }

            Console.ReadLine();
        }
```

Task 4: Run the application and test the functionality. The Console Window running is as below:


file:///H:/Mahesh_Practice/Capgemini/CSharp_Training_09_05_13_05/Day5/CS_FileIO/CS_FileIO/bin...

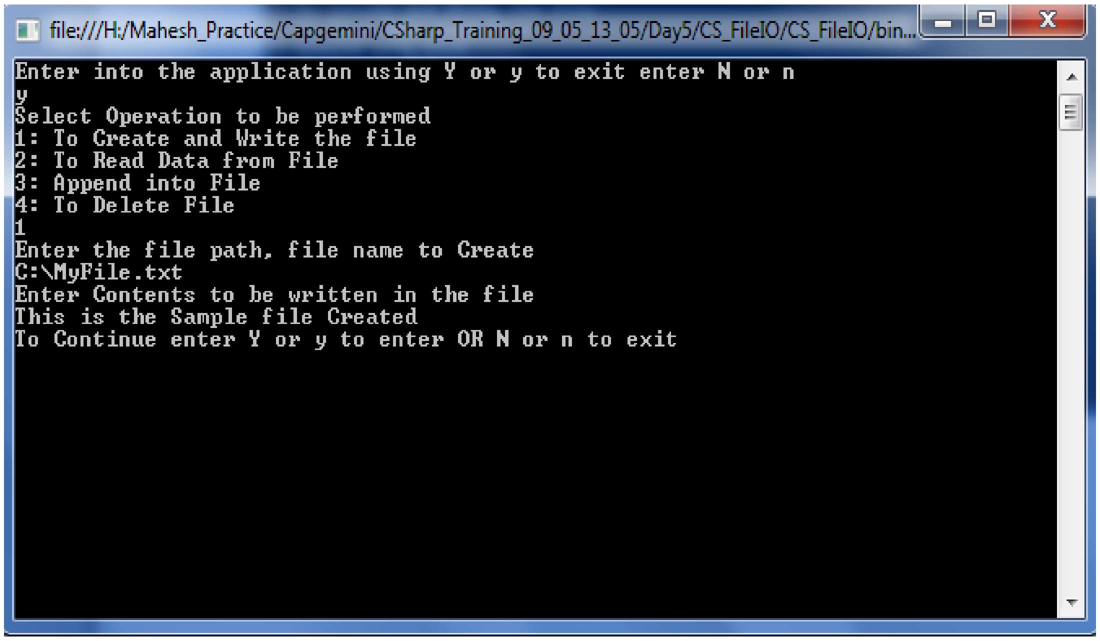
```
Enter into the application using Y or y to exit enter N or n
y_
```

After entering ‘y’ or ‘Y’ the window is as below:


file:///H:/Mahesh_Practice/Capgemini/CSharp_Training_09_05_13_05/Day5/CS_FileIO/CS_FileIO/bin...

```
Enter into the application using Y or y to exit enter N or n
y
Select Operation to be performed
1: To Create and Write the file
2: To Read Data from File
3: Append into File
4: To Delete File
_
```

Enter 1 to create a file with following parameters:


file:///H:/Mahesh_Practice/Capgemini/CSharp_Training_09_05_13_05/Day5/CS_FileIO/CS_FileIO/bin...

```
Enter into the application using Y or y to exit enter N or n
y
Select Operation to be performed
1: To Create and Write the file
2: To Read Data from File
3: Append into File
4: To Delete File
1
Enter the file path, file name to Create
C:\MyFile.txt
Enter Contents to be written in the file
This is the Sample file Created
To Continue enter Y or y to enter OR N or n to exit
```

Enter ‘y’ and then enter 2 to, it will ask you the file name of which contents are read after entering the file you created it will show its contents. Test all the functionality.

Serialization:

In this lab we are going to see the Serialization process.

Serialization is a process of saving the object data or the state across time and space. This data can be retrieved back and object can be recreated through a reverse process called as de-serialization.

Exercise 2: Serializing the objects using Binary, Soap and Xml Serialization.

Task 1: Open VS2010 and create a C# console application, name it as 'CS_Serialization'. Use the following namespace in the Program.cs.

```
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters;
```

For Binary Serialization

```
using System.Runtime.Serialization.Formatters.Binary;
```

For XML Serialization

```
using System.Xml.Serialization;
```

For SOAP Serialization

```
using System.Runtime.Serialization.Formatters.Soap;
```

Task 2: In the Program.cs add Employee class as follows:

```
[Serializable]
   public class Employee
   {
      public int EmpNo { get; set; }
      public string EmpName { get; set; }
   }
```

Look at the above class carefully, we have applied "Serializable" attribute to the class Employee. This attribute allows us to serialize the objects of the Employee class.

Task 3: Add BinarySerialization class in Program.cs

This class contains two methods as follows:

1: SerializeObject: This function is going to make use of FileStream Class to store data in a file. BinaryFormatter class: Binary Serialization writes content of an object in a binary form.

2: DeSerializeObject: This function applies reverse action. That means we have to de-serialize the objects which are already stored in the file. Call Deserialize method on object of BinaryFormatter class and make sure that you type cast it to Employee class. Note down the below steps.

```
public class BinarySerialization
   {
      public void SerializeObject(Employee objEmp, string FileName)
      {
         FileStream Fs = new FileStream(@FileName,FileMode.Create);
         BinaryFormatter bFormat = new BinaryFormatter();
         bFormat.Serialize(Fs, objEmp);
         Fs.Close();
         Fs.Dispose();
      }

      public Employee DeSerializeObject(string FileName)
      {
         Employee objEmp;
         FileStream Fs = new FileStream(@FileName, FileMode.Open);
         BinaryFormatter bFormat = new BinaryFormatter();
         objEmp= bFormat.Deserialize(Fs) as Employee;
         Fs.Close();
         Fs.Dispose();
         return objEmp;
      }

   }
```

Task 4: Add one more class XMLSerialization to Program.cs

This class also contains two methods as follows:

1: SerializeObject: This function is going to make use of FileStream Class to store data in a xml file.

XmlSerializer class: Xml Serialization writes content of an object in xml File.

2: DeSerializeObject: This function applies reverse action. That means we have to de-serialize the objects which are already stored in the xml file. Call the Deserialize method on object of XmlSerializer and make sure that you type cast it to Employee class.

```
public class XMLSerialization
  {
     public void SerializeObject(Employee objEmp, string FileName)
     {
        FileStream Fs = new FileStream(@FileName, FileMode.Create);
        XmlSerializer xmlSer = new XmlSerializer(typeof(Employee));
        xmlSer.Serialize(Fs, objEmp);
        Fs.Close();
        Fs.Dispose();
     }

     public Employee DeSerializeObject(string FileName)
     {
        Employee objEmp;
        FileStream Fs = new FileStream(@FileName, FileMode.Open);
        XmlSerializer xmlSer = new XmlSerializer(typeof(Employee));
        objEmp =xmlSer.Deserialize(Fs) as Employee;
        Fs.Close();
        Fs.Dispose();
        return objEmp;
     }
  }
```

Task 5: Add one more class SOAPSerialization to Program.cs
This class also contains two methods as follows:

1: SerializeObject: This function is going to make use of FileStream Class to store data in a file. SoapFormatter class: Objects are serialized into a SOAP message. It provides reliable way to serialize objects that will be transmitted across a network.

2: DeSerializeObject: This function applies reverse action. That means we have to de-serialize the objects which are already stored in the file. Call the Deserialize method on object of SoapFormatter class and make sure that you type cast it to Employee class.

```
public class SOAPSerialization
  {
     public void SerializeObject(Employee objEmp, string FileName)
     {
            FileStream Fs = new FileStream(@FileName, FileMode.Create);
            SoapFormatter sf = new SoapFormatter();
            sf.Serialize(Fs, objEmp);
            Fs.Close();
            Fs.Dispose();
        }

        public Employee DeSerializeObject(string FileName)
        {
            Employee objEmp;
            FileStream Fs = new FileStream(@FileName, FileMode.Open);
            SoapFormatter sf = new SoapFormatter();
            objEmp = sf.Deserialize(Fs) as Employee;
            Fs.Close();
            Fs.Dispose();
            return objEmp;
        }
    }
```

Now your classes are ready to use.
Now go to Main method of your Program class and call the respective methods.

Task 6: Create Objects of all classes.

```
            BinarySerialization binSerialization = new BinarySerialization();
            XMLSerialization xmlSerialization = new XMLSerialization();
            SOAPSerialization soapSerialization = new SOAPSerialization();
```

Task 7: Create object of Employee class

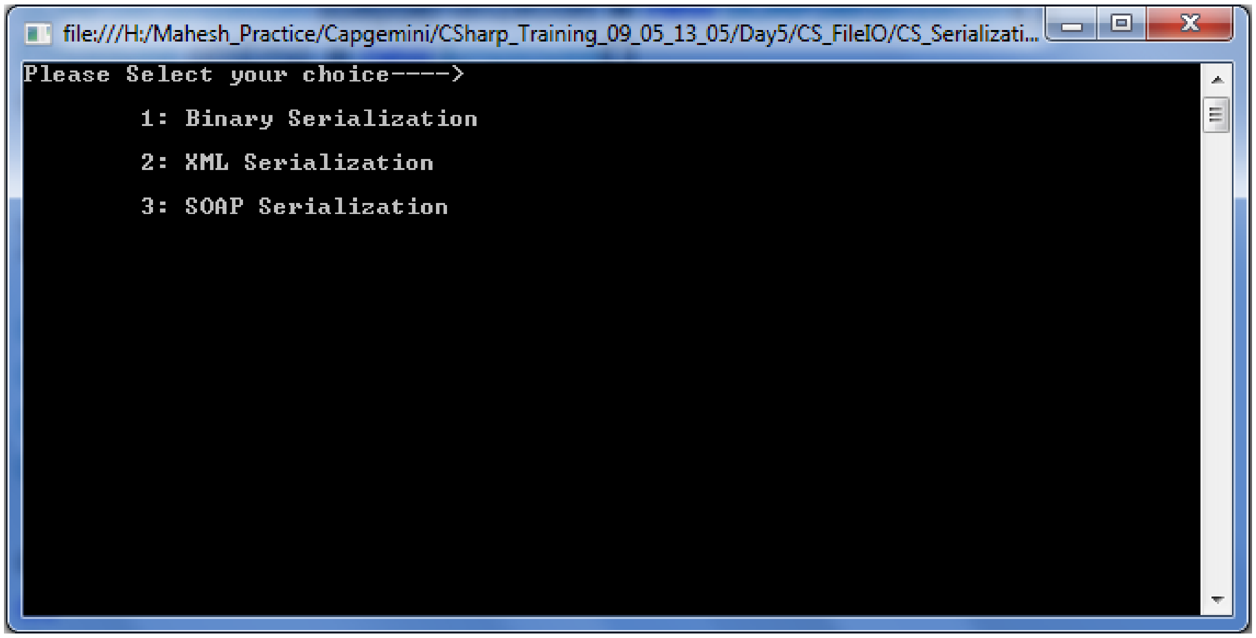
```
            Employee objEmp = new Employee()
            {
                EmpNo=101,
                EmpName="My Name"
            };
```

Task 8: Generate a code which will display a menu driven output to user in the following manner.

```
            Console.WriteLine("Please Select your choice---->");
            Console.WriteLine("\n\t1: Binary Serialization");
```

```
        Console.WriteLine("\n\t2: XML Serialization");
        Console.WriteLine("\n\t3: SOAP Serialization");
        choice = int.Parse(Console.ReadLine());


        do
        {
          switch (choice)
          {
            case 1:
                  binSerialization.SerializeObject(objEmp,@"c:\MyEmployee.dat");
                  binSerialization.DeSerializeObject(@"c:\MyEmployee.dat");

                  //Binary Serialization.
                  Console.WriteLine("Binary Deserialization Results are:");
                  Console.WriteLine();
               Console.WriteLine("EmpNo :" + objEmp.EmpNo + "\t EmpName :" + objEmp.EmpName);
                  Console.ReadLine();
                  break;

            case 2:
                  xmlSerialization.SerializeObject(objEmp, @"c:\MyEmployeeXml.xml");
                  xmlSerialization.DeSerializeObject(@"c:\MyEmployeeXml.xml");

                  //Xml Serialization.
                  Console.WriteLine("XML Deserialization Results are:");
                  Console.WriteLine();
               Console.WriteLine("EmpNo:" + objEmp.EmpNo + "\tEmpName:" + objEmp.EmpName);
                  Console.ReadLine();
                  break;

            case 3:
                  soapSerialization.SerializeObject(objEmp, @"c:\MyEmployeeSoap.dat");
                  soapSerialization.DeSerializeObject(@"c:\MyEmployeeSoap.dat");

                  //SOAP Serilization.
                  Console.WriteLine("SOAP Deserialization Results are:");
                  Console.WriteLine();
               Console.WriteLine("EmpNo:" + objEmp.EmpNo + "\tEmpName:" + objEmp.EmpName);
                  Console.ReadLine();
                  break;

          }
          Console.WriteLine("\n\t1: Binary Serialization");
          Console.WriteLine("\n\t2: XML Serialization");
          Console.WriteLine("\n\t3: SOAP Serialization");
          Console.WriteLine("\n\t4:Exit");
          choice = int.Parse(Console.ReadLine());
        } while (choice!= 4);
```

Bellow Screen will get displayed for the first time

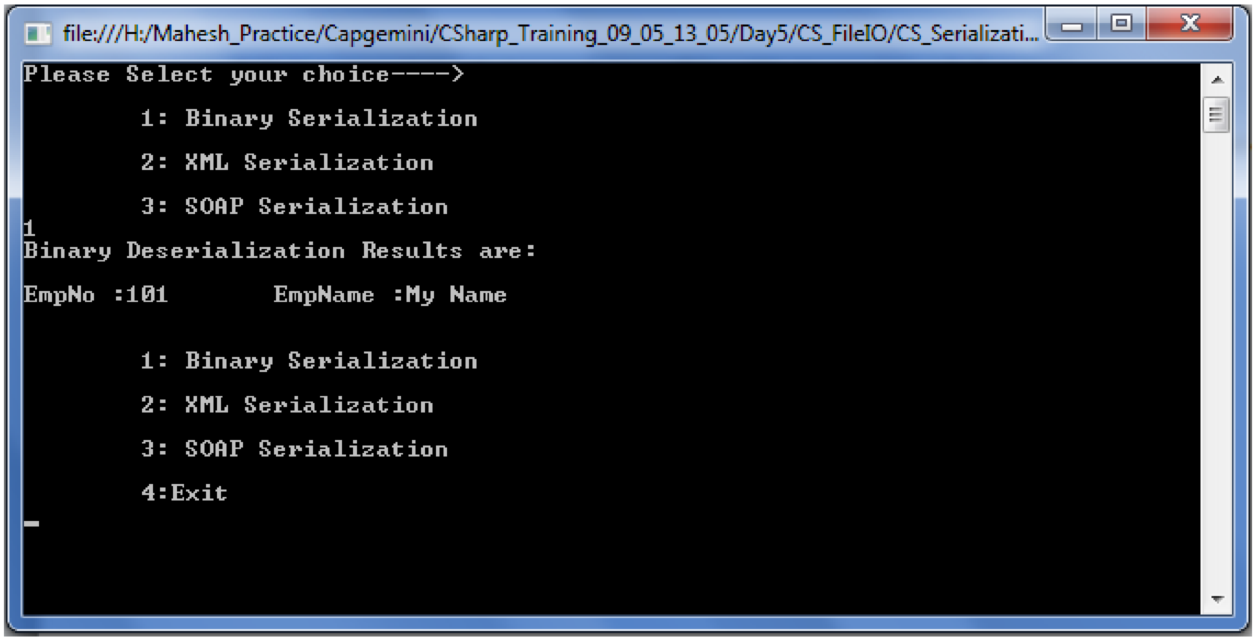


file:///H:/Mahesh_Practice/Capgemini/CSharp_Training_09_05_13_05/Day5/CS_FileIO/CS_Serializati...

```
Please Select your choice---->
        1: Binary Serialization
        2: XML Serialization
        3: SOAP Serialization
```

Select appropriate choice and you will be able to see intended output.

For the subsequent requests the following menu will be displayed including exit option.


file:///H:/Mahesh_Practice/Capgemini/CSharp_Training_09_05_13_05/Day5/CS_FileIO/CS_Serializati...

```
Please Select your choice---->

        1: Binary Serialization

        2: XML Serialization

        3: SOAP Serialization
1
Binary Deserialization Results are:

EmpNo :101       EmpName :My Name


        1: Binary Serialization

        2: XML Serialization

        3: SOAP Serialization

        4:Exit
_
```