C# - READING FROM AND WRITING INTO BINARY FILES

The **BinaryReader** and **BinaryWriter** classes are used for reading from and writing to a binary file.

The BinaryReader Class

The **BinaryReader** class is used to read binary data from a file. A **BinaryReader** object is created by passing a **FileStream** object to its constructor.

The following table describes commonly used **methods** of the **BinaryReader** class.

Sr.No.	Methods
1	**public override void Close** It closes the BinaryReader object and the underlying stream.
2	**public virtual int Read** Reads the characters from the underlying stream and advances the current position of the stream.
3	**public virtual bool ReadBoolean** Reads a Boolean value from the current stream and advances the current position of the stream by one byte.
4	**public virtual byte ReadByte** Reads the next byte from the current stream and advances the current position of the stream by one byte.
5	**public virtual byte[] ReadBytes*intcount*** Reads the specified number of bytes from the current stream into a byte array and advances the current position by that number of bytes.
6	**public virtual char ReadChar** Reads the next character from the current stream and advances the current position of the stream in accordance with the Encoding used and the specific character being read from the stream.
7	**public virtual char[] ReadChars*intcount*** Reads the specified number of characters from the current stream, returns the data in a character array, and advances the current position in accordance with the Encoding used and the specific character being read from the stream.
8	**public virtual double ReadDouble** Reads an 8-byte floating point value from the current stream and advances the current position of the stream by eight bytes.

9	**public virtual int ReadInt32** Reads a 4-byte signed integer from the current stream and advances the current position of the stream by four bytes.
10	**public virtual string ReadString** Reads a string from the current stream. The string is prefixed with the length, encoded as an integer seven bits at a time.

The BinaryWriter Class

The **BinaryWriter** class is used to write binary data to a stream. A BinaryWriter object is created by passing a FileStream object to its constructor.

The following table describes commonly used methods of the BinaryWriter class.

Sr.No.	Functions
1	**public override void Close** It closes the BinaryWriter object and the underlying stream.
2	**public virtual void Flush** Clears all buffers for the current writer and causes any buffered data to be written to the underlying device.
3	**public virtual long Seek*intoffset, SeekOriginorigin*** Sets the position within the current stream.
4	**public virtual void Write*boolvalue*** Writes a one-byte Boolean value to the current stream, with 0 representing false and 1 representing true.
5	**public virtual void Write*bytevalue*** Writes an unsigned byte to the current stream and advances the stream position by one byte.
6	**public virtual void Write*byte[]buffer*** Writes a byte array to the underlying stream.
7	**public virtual void Write*charch*** Writes a Unicode character to the current stream and advances the current position of the stream in accordance with the Encoding used and the specific characters being written to the stream.
8	**public virtual void Write*char[]chars*** Writes a character array to the current stream and advances the current position of the stream in accordance with the Encoding used and the specific characters being written to the stream.

9	**public virtual void Write*doublevalue*** Writes an eight-byte floating-point value to the current stream and advances the stream position by eight bytes.
10	**public virtual void Write*intvalue*** Writes a four-byte signed integer to the current stream and advances the stream position by four bytes.
11	**public virtual void Write*stringvalue*** Writes a length-prefixed string to this stream in the current encoding of the BinaryWriter, and advances the current position of the stream in accordance with the encoding used and the specific characters being written to the stream.

For a complete list of methods, please visit Microsoft C# documentation.

Example

The following example demonstrates reading and writing binary data:

```
using System;
using System.IO;

namespace BinaryFileApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         BinaryWriter bw;
         BinaryReader br;
         int i = 25;
         double d = 3.14157;
         bool b = true;
         string s = "I am happy";

         //create the file
         try
         {
            bw = new BinaryWriter(new FileStream("mydata", FileMode.Create));
         }
         catch (IOException e)
         {
            Console.WriteLine(e.Message + "\n Cannot create file.");
            return;
         }

         //writing into the file
         try
         {
            bw.Write(i);
            bw.Write(d);
            bw.Write(b);
            bw.Write(s);
         }

         catch (IOException e)
         {
            Console.WriteLine(e.Message + "\n Cannot write to file.");
            return;
         }
```

```
            bw.Close();

            //reading from the file
            try
            {
                br = new BinaryReader(new FileStream("mydata", FileMode.Open));
            }
            catch (IOException e)
            {
                Console.WriteLine(e.Message + "\n Cannot open file.");
                return;
            }
            try
            {
                i = br.ReadInt32();
                Console.WriteLine("Integer data: {0}", i);
                d = br.ReadDouble();
                Console.WriteLine("Double data: {0}", d);
                b = br.ReadBoolean();
                Console.WriteLine("Boolean data: {0}", b);
                s = br.ReadString();
                Console.WriteLine("String data: {0}", s);
            }
            catch (IOException e)
            {
                Console.WriteLine(e.Message + "\n Cannot read from file.");
                return;
            }
            br.Close();

            Console.ReadKey();
        }
    }
}
```

When the above code is compiled and executed, it produces the following result:

```
Integer data: 25
Double data: 3.14157
Boolean data: True
String data: I am happy
```