

## ***java.io.InputStream* class methods**

*int available ( )* throws java.io.IOException  
returns the number of bytes currently available for reading.

*void close ( )* throws java.io.IOException  
closes the input stream.

*int read ( )* throws java.io.IOException  
returns the next byte of data from the input stream or -1 when the end of stream is encountered.

*int read (byte[ ] byteArray)* throws java.io.IOException  
reads a sequence of bytes and places them in the specified byte array and returns the number of bytes read or -1 which indicates the end of stream.

*long skip(long numBytes )* throws java.io.IOException  
skips over the specified number of bytes specified by numBytes. Returns the number of bytes that was successfully skipped over.

```
import java.io.*;
public class sampleInputStream {
    public static void main(String args[ ]) {
        try {
            FileInputStream inFile = new
                FileInputStream("sample.txt");
            int inData = inFile.read ( );
            while (inData != -1) {
                System.out.write(inData);
                inData = inFile.read( );
            }//end of while
            inFile.close( );
        }//end of try
        catch (IOException e) {
            System.out.println("I/O exception -" + e);
        }
    }//end of main
} //end of class definition
```

## ***java.io.OutputStream* class methods**

*void close ( )* throws java.io.IOException  
closes the output stream.

*void flush ( )* throws java.io.IOException  
flushes all buffered data to the output stream.

*void write (int byte)* throws java.io.IOException  
writes the specified byte.

*void write (byte [ ] byteArray)* throws java.io.IOException  
writes the contents of the byteArray to the output stream.

```
import java.io.*;
public class sampleOutputStream {
    public static void main(String args[ ] ) {
        try {
            FileInputStream inFile =
                new FileInputStream( "Sample.txt" );
            FileOutputStream outFile =
                new FileOutputStream ( "SamOut.txt" );
            int inData = inFile.read( );

            while ( inData != -1) {
                outFile.write (inData);
                inData=inFile.read();
            }// end of while

            // Close both streams
            inFile.close( );
            outFile.close( );
            System.out.println("I/Ostreams closed");
        }// end of try
        catch (IOException e) {
            System.err.println ("Error: " + e);
        }
    }// end of main
} // end of class definition
```

## ***BufferedInputStream* class constructor**

*BufferedInputStream (InputStream inData)*

Creates a buffered stream that will read from the specified `InputStream` object.

## ***DataInputStream* class constructor**

*DataInputStream (InputStream inData)*

Creates a data input stream that will read from the specified `InputStream` object.

## ***DataInputStream* class methods**

*byte readByte ( )* throws `java.io.EOFException`, `java.io.IOException`  
reads a byte value from the input stream.

*char readChar ( )* throws `java.io.EOFException`, `java.io.IOException`  
reads a character value from the input stream.

*double readDouble ( )* throws `java.io.EOFException`,  
`java.io.IOException`  
reads 8 bytes from the input stream and returns a double.

*float readFloat ( )* throws `java.io.EOFException`, `java.io.IOException`  
reads 4 bytes from the input stream and returns a float.

*int readInt ( )* throws `java.io.EOFException`, `java.io.IOException`  
reads 4 bytes from the input stream and returns an int.

*String readLine ( )* throws `java.io.IOException`  
reads an entire line from the input stream and returns a string.

*long readLong ( )* throws `java.io.EOFException`, `java.io.IOException`  
reads 8 bytes from the input stream and returns a long.

*short readShort ( )* throws `java.io.EOFException`, `java.io.IOException`  
reads 2 bytes from the input stream and returns a short.

## ***BufferedOutputStream* class constructor**

*BufferedOutputStream (OutputStream outData)*

Creates a buffered stream for writing on a specified OutputStream object. Buffer size is 512 bytes.

## ***DataOutputStream* class constructor**

*DataOutputStream (OutputStream outData)*

Creates a data output stream that will write to a specified OutputStream object.

## ***DataOutputStream* class methods**

*int size ( )*

returns the number of bytes written to the output stream.

*void writeByte (int byte )* throws java.io.IOException

writes the specified byte to the output stream.

*void writeBytes (String inString )* throws java.io.IOException

writes the specified string to the output stream.

*void writeChar (char inChar )* throws java.io.IOException

writes the specified char to the output stream.

*void writeDouble (double dVal )* throws java.io.IOException

writes the specified double value to the output stream.

*void writeFloat (float fVal )* throws java.io.IOException

writes the specified float value to the output stream.

*void writeInt (int iVal )* throws java.io.IOException

writes the specified int value to the output stream.

*void writeLong (Long lVal )* throws java.io.IOException

writes the specified long value to the output stream.

*void writeShort (short sVal )* throws java.io.IOException

## **Java Readers and Writers**

**- better suited for Unicode characters**

### ***java.io.Reader* class methods**

*void close ( )* throws *java.io.IOException*  
closes the reader.

*int read ( )* throws *java.io.IOException*  
returns a character, blocks if character is not yet available, -1 if  
end of reader stream is encountered.

*int read (char[ ] charArray)* throws *java.io.IOException*  
reads a sequence of chars and places them in the specified  
char array and returns the number of chars read or -1 which indicates  
the end of reader stream.

*long skip(long numChars )* throws *java.io.IOException*  
skips over the specified number of chars. Returns the number  
of chars that was successfully skipped over.

### **Low-level Readers**

*CharArrayReader (char [ ] charArray )*

Creates a character array reader that will operate on the *charArray*.

*FileReader (File file )*, *FileReader(String filename)*

Creates a reader that will access the contents of the specified file  
object or filename.

*InputStreamReader (InputStream input )*

Connects an input stream to the reader.

*BufferedReader (Reader readerObj )*

Reads data from the specified reader into a buffer.

## ***java.io.Writer* class methods**

*void close ( )* throws java.io.IOException  
closes the writer stream.

*void flush ( )* throws java.io.IOException  
flushes all buffered data to the output stream.

*void write (char chars)* throws java.io.IOException  
writes the specified char.

*void write (char [ ] charArray)* throws java.io.IOException  
writes the contents of the charArray to the output stream.

*void write (String inString)* throws java.io.IOException  
writes the specified string.

## ***Low-level Writers***

*CharArrayWriter ( )*

Creates a character array writer that can be converted to a character array.

*FileWriter (File file )*, *FileWriter(String filename)*,  
*FileWriter (String fileName, Boolean appendFlag)*

Creates a writer that is connected the specified file object or filename.

*OutputStreamWriter (OutputStream output)*

Creates a writer that will translate between characters and bytes, using the default character encoding.

*BufferedWriter (Writer writerObj )*

Creates a buffered writer connected to a specified writer.

*PrintWriter (Writer writerObj )*

Creates a print writer connected to the specified writer.

```
import java.io.*;

public class SampleReaderWriter {
    public static void main(String args[ ]) {
        char [] buf = new char[64];
        int chRead = 0;
        try {
            FileReader inFile = new
                FileReader("sample.txt");
            FileWriter outFile = new
                FileWriter("samOut.txt");
            while((chRead=inFile.read(buf))>-1) {
                outFile.write(buf,0,charsRead);
            }//end of while
            inFile.close( );
            outFile.close();
        }//end of try
        catch (IOException e) {
            System.out.println("I/O exception -" + e);
        }
    }//end of main
} //end of class definition
```