

READING AND WRITING FILES

In Java, all files are byte-oriented, and **Java provides methods to read and write bytes from and to a file.**

Two of the most often-used stream classes are `FileInputStream` and `FileOutputStream`, which create byte streams linked to files.

File input stream

This stream is used for reading data from the files. Objects can be created using the keyword `new` and there are several types of constructors available.

The two constructors which can be used to create a `FileInputStream` object:

- i) Following constructor takes a file name as a string to create an input stream object to read the file:

`InputStream f = new FileInputStream("filename ");`

- ii) Following constructor takes a file object to create an input stream object to read the file. First we create a file object using `File()` method as follows:

`File f = new File("C:/java/hello");`

`InputStream f = new FileInputStream(f);`

Methods to read stream or to do other operations on the stream

Method	Description
<code>public void close() throws IOException {}</code>	<ul style="list-style-type: none"> • Closes the file output stream. • Releases any system resources associated with the file. • Throws an <code>IOException</code>.
<code>protected void finalize()throws IOException {}</code>	<ul style="list-style-type: none"> • Cleans up the connection to the file. • Ensures that the <code>close</code> method of this file output stream is called when there are no more references to this stream. • Throws an <code>IOException</code>.
<code>public int read(int r)throws IOException {}</code>	<ul style="list-style-type: none"> • Reads the specified byte of data from the <code>InputStream</code>. • Returns an <code>int</code>. • Returns the next byte of data and <code>-1</code> will be returned if it's the end of the file.
<code>public int read(byte[] r) throws IOException {}</code>	<ul style="list-style-type: none"> • Reads <code>r.length</code> bytes from the input stream into an array. • Returns the total number of bytes read. If it is the end of the file, <code>-1</code> will be returned.
<code>public int available() throws IOException {}</code>	<ul style="list-style-type: none"> • Gives the number of bytes that can be read from this file input stream. • Returns an <code>int</code>.

File output stream

FileOutputStream is used to create a file and write data into it.

The stream would create a file, if it doesn't already exist, before opening it for output.

The two constructors which can be used to create a FileOutputStream object:

- i) Following constructor takes a file name as a string to create an input stream object to write the file:

OutputStream f = new FileOutputStream("filename");

- ii) Following constructor takes a file object to create an output stream object to write the file. First, we create a file object using File() method as follows:

File f = new File("C:/java/hello");

OutputStream f = new FileOutputStream(f);

Methods to write to stream or to do other operations on the stream

Method	Description
public void close() throws IOException {}	<ul style="list-style-type: none"> • Closes the file output stream. • Releases any system resources associated with the file. • Throws an IOException.
protected void finalize()throws IOException {}	<ul style="list-style-type: none"> • Cleans up the connection to the file. • Ensures that the close method of this file output stream is called when there are no more references to this stream. • Throws an IOException.
public void write(int w)throws IOException {}	<ul style="list-style-type: none"> • Writes the specified byte to the output stream.
public void write(byte[] w)	<ul style="list-style-type: none"> • Writes w.length bytes from the mentioned byte array to the OutputStream.

Following code demonstrates the use of InputStream and OutputStream.

```
import java.io.*;
public class fileStreamTest
{
    public static void main(String args[])
    {
        try
        {
            byte bWrite [] = {11,21,3,40,5};
            OutputStream os = new FileOutputStream("test.txt");
            for(int x = 0; x < bWrite.length ; x++)
            {
```

```
        os.write( bWrite[x] ); // writes the bytes
    }
    os.close();
    InputStream is = new FileInputStream("test.txt");
    int size = is.available();
    for(int i = 0; i < size; i++)
    {
        System.out.print((char)is.read() + " ");
    }
    is.close();
}
catch (IOException e)
{
    System.out.print("Exception");
}
}
```

The above code creates a file named test.txt and writes given numbers in binary format. The same will be displayed as output on the stdout screen.

