JAVA BASICS

4KCJ001 - Lecture 8

By
Sudha Agarwal
sudha.agarwal@4kitsolutions.com

http://www.4kitsolutions.com/

INDEX

- ❖ File I/O

http://www.4kitsolutions.com/

FILE HANDLING IN JAVA

❖Java I/O (Input and Output) is used to process the input and produce the output.

❖Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

❖We can perform file handling in java by Java I/O API.

STREAM

❖A stream is a sequence of data. In Java a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

❖In java, 3 streams are created for us automatically. All these streams are attached with console.

- ❖1) **System.out**: standard output stream
- ❖2) **System.in**: standard input stream
- ❖3) **System.err**: standard error stream

OUTPUTSTREAM/INPUTSTREAM

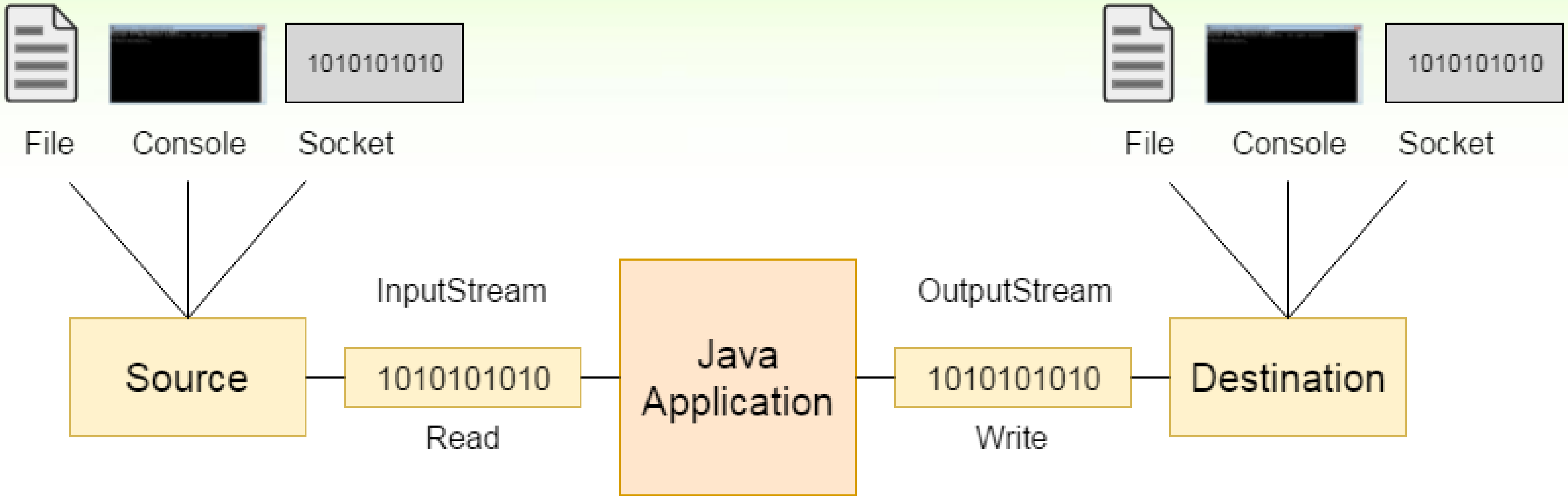
❖OutputStream

❖Java application uses an output stream to write data to a destination, it may be a file, an array, peripheral device or socket.

❖InputStream

❖Java application uses an input stream to read data from a source, it may be a file, an array, peripheral device or socket.

OUTPUTSTREAM/INPUTSTREAM



http://www.4kitsolutions.com/

OutputStream class

❖OutputStream class is an abstract class. It is the super class of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

❖Method

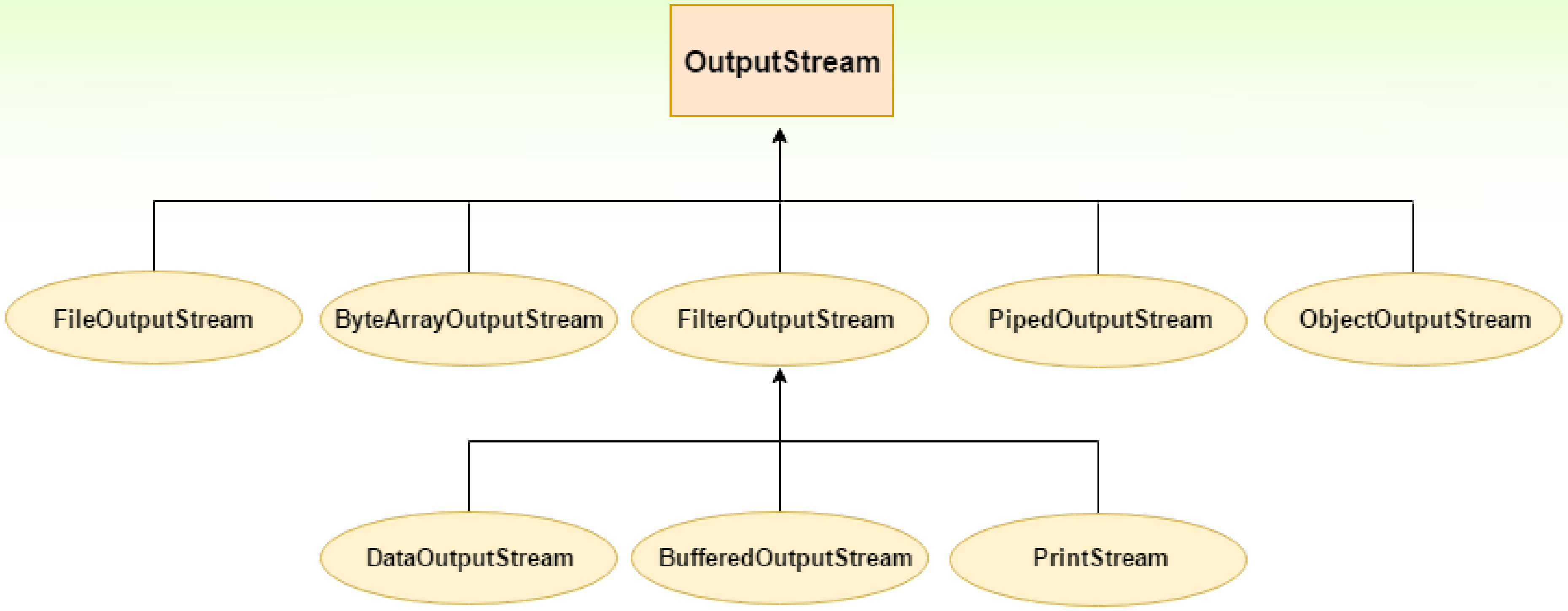
❖**1) public void write(int)throws IOException** - is used to write a byte to the current output stream.

❖**2) public void write(byte[])throws IOException** - is used to write an array of byte to the current output stream.

❖**3) public void flush()throws IOException** - flushes the current output stream.

❖**4) public void close()throws IOException** - is used to close the current output stream.

OUTPUTSTREAM HIERARCHY



FileOutputStream Class

❖Java FileOutputStream is an output stream used for writing data to a file.

❖If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

FILEOUTPUTSTREAM CLASS METHODS

❖**protected void finalize()** - It is used to clean up the connection with the file output stream.

❖**void write(byte[] ary)** - It is used to write ary.length bytes from the byte array to the file output stream.

❖**void write(byte[] ary, int off, int len)** - It is used to write len bytes from the byte array starting at offset off to the file output stream.

❖**void write(int b)** - It is used to write the specified byte to the file output stream.

❖**FileChannel getChannel()** - It is used to return the file channel object associated with the file output stream.

❖**FileDescriptor getFD() -** It is used to return the file descriptor associated with the stream.

❖**void close()** - It is used to closes the file output stream.

FileInputStream Class

❖Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc.

❖You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

FILEINPUTSTREAM CLASS METHODS

- **int available()** It is used to return the estimated number of bytes that can be read from the input stream.
- **int read()** It is used to read the byte of data from the input stream.
- **int read(byte[] b)** It is used to read up to b.length bytes of data from the input stream.
- **int read(byte[] b, int off, int len)** It is used to read up to len bytes of data from the input stream.
- **protected void finalize()** It is used to ensure that the close method is call when there is no more reference to the file input stream.
- **void close()** It is used to closes the stream.

BufferedOutputStream Class

❖Java BufferedOutputStream class is used for buffering an output stream. It internally uses buffer to store data. It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

❖For adding the buffer in an OutputStream, use the BufferedOutputStream class. Let's see the syntax for adding the buffer in an OutputStream:

❖OutputStream os= new BufferedOutputStream(new FileOutputStream("D:\\IO Package\\testout.txt"));

BUFFEREDOUTPUTSTREAM

❖Constructor:

❖**BufferedOutputStream(OutputStream os)** It creates the new buffered output stream which is used for writing the data to the specified output stream.

❖**BufferedOutputStream(OutputStream os, int size)** It creates the new buffered output stream which is used for writing the data to the specified output stream with a specified buffer size.

BUFFEREDOUTPUTSTREAM METHODS

❖Method:

❖**void write(int b)** It writes the specified byte to the buffered output stream.

❖**void write(byte[] b, int off, int len)** It write the bytes from the specified byte-input stream into a specified byte array, starting with the given offset

❖**void flush()** It flushes the buffered output stream.

BUFFEREDINPUTSTREAM CLASS

❖Java BufferedInputStream class is used to read information from stream. It internally uses buffer mechanism to make the performance fast.

❖The important points about BufferedInputStream are:

❖When the bytes from the stream are skipped or read, the internal buffer automatically refilled from the contained input stream, many bytes at a time.

❖When a BufferedInputStream is created, an internal buffer array is created.

BufferedInputStream Class

❖Constructor

❖**BufferedInputStream(InputStream IS)** It creates the BufferedInputStream and saves it argument, the input stream IS, for later use.

❖**BufferedInputStream(InputStream IS, int size)** It creates the BufferedInputStream with a specified buffer size and saves it argument, the input stream IS, for later use.

BufferedInputStream Methods

- Method
- **int available()** It returns an estimate number of bytes that can be read from the input stream without blocking by the next invocation method for the input stream.
- **int read()** It read the next byte of data from the input stream.
- **int read(byte[] b, int off, int ln)** It read the bytes from the specified byte-input stream into a specified byte array, starting with the given offset.
- **void close()** It closes the input stream and releases any of the system resources associated with the stream.
- **void reset()** It repositions the stream at a position the mark method was last called on this input stream.

READER CLASS

❖The Java Reader class (java.io.Reader) is the base class for all Reader subclasses in the Java IO API.

❖A Reader is like an InputStream except that it is character based rather than byte based.

❖In other words, a Java Reader is intended for reading text, whereas an InputStream is intended for reading raw bytes.

❖Some of the implementation class are BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader

❖The **read()** method of a Reader returns an int which contains the char value of the next character read. If the read() method returns -1, there is no more data to read in the Reader, and it can be closed.

FILEREADER CLASS

❖The Java.io.FileReader class is a convenience class for reading character files.

❖FileReader is meant for reading streams of characters. For reading streams of raw bytes, use FileInputStream.

❖Constructors of FileReader class:

❖**FileReader(String file)** - It gets filename in string. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.

❖**FileReader(File file)** - It gets filename in file instance. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.

FileReader Class - Methods

❖**int read()** - It is used to return a character in ASCII form. It returns -1 at the end of file.

❖**void close()** - It is used to close the FileReader class.

WRITER CLASS

❖The Java Writer class (java.io.Writer) is the base class for all Writer subclasses in the Java IO API. A Writer is like an OutputStream except that it is character based rather than byte based.

❖Subclasses of Writer include OutputStreamWriter, CharArrayWriter, FileWriter

FileWriter Class

❖Java FileWriter class is used to write character-oriented data to a file. It is character-oriented class which is used for file handling in java.

❖Unlike FileOutputStream class, you don't need to convert string into byte array because it provides method to write string directly.

❖Constructors of FileWriter class:

❖**FileWriter(String file)** - Creates a new file. It gets file name in string.

❖**FileWriter(File file)** - Creates a new file. It gets file name in File object.

FileWriter class - Methods

❖**void write(String text)** It is used to write the string into FileWriter.

❖**void write(char c)** It is used to write the char into FileWriter.

❖**void write(char[] c)** It is used to write char array into FileWriter.

❖**void flush()** It is used to flushes the data of FileWriter.

❖**void close()** It is used to close the FileWriter.

BufferedReader Class

❖ava BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast.

❖It inherits Reader class.

❖BufferedReader class constructors:

❖**BufferedReader(Reader rd)** It is used to create a buffered character input stream that uses the default size for an input buffer.

❖**BufferedReader(Reader rd, int size)** It is used to create a buffered character input stream that uses the specified size for an input buffer.

❖default buffer size is 8192 characters capacity

BUFFEREDREADER CLASS METHODS

- **int read()** It is used for reading a single character.
- **int read(char[] cbuf, int off, int len)** It is used for reading characters into a portion of an array.
- **boolean markSupported()** It is used to test the input stream support for the mark and reset method.
- **String readLine()** It is used for reading a line of text.
- **boolean ready()** It is used to test whether the input stream is ready to be read.
- **long skip(long n)** It is used for skipping the characters.
- **void reset()** It repositions the stream at a position the mark method was last called on this input stream.
- **void mark(int readAheadLimit)** It is used for marking the present position in a stream.
- **void close()** It closes the input stream and releases any of the system resources associated with the stream.

BufferedWriter Class

❖The Java.io.BufferedWriter class writes text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings.

❖Class constructors:

❖**BufferedWriter(Writer wrt)** It is used to create a buffered character output stream that uses the default size for an output buffer.

❖**BufferedWriter(Writer wrt, int size)** It is used to create a buffered character output stream that uses the specified size for an output buffer.

BufferedWriter Class Methods

❖**void newLine()** It is used to add a new line by writing a line separator.

❖**void write(int c)** It is used to write a single character.

❖**void write(char[] cbuf, int off, int len)** It is used to write a portion of an array of characters.

❖**void write(String s, int off, int len)** It is used to write a portion of a string.

❖**void flush()** It is used to flushes the input stream.

❖**void close()** It is used to closes the input stream

SCANNER CLASS

❖Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double etc. and strings. It is the easiest way to read input in a Java program'

❖To create an object of Scanner class, we usually pass the predefined object System.in, which represents the standard input stream. We may pass an object of class File if we want to read input from a file.

❖To read numerical values of a certain data type XYZ, the function to use is nextXYZ(). For example, to read a value of type short, we can use nextShort()

❖To read strings, we use nextLine().

❖To read a single character, we use next().charAt(0). next() function returns the next token/word in the input as a string and charAt(0) funtion returns the first character in that string.

SCANNER CLASS METHODS

- **public String next()** it returns the next token from the scanner.
- **public String nextLine()** it moves the scanner position to the next line and returns the value as a string.
- **public byte nextByte()** it scans the next token as a byte.
- **public short nextShort()** it scans the next token as a short value.
- **public int nextInt()** it scans the next token as an int value.
- **public long nextLong()** it scans the next token as a long value.
- **public float nextFloat()** it scans the next token as a float value.
- **public double nextDouble()** it scans the next token as a double value.

Scanner class /BufferedReader

❖The Scanner has a little buffer (1KB char buffer) as opposed to the BufferedReader (8KB byte buffer), but it’s more than enough.

❖BufferedReader is a bit faster as compared to scanner because scanner does parsing of input data and BufferedReader simply reads sequence of characters.

SCANNER CLASS /BUFFEREDREADER

❖java.util.Scanner class is a simple text scanner which can parse primitive types and strings. It internally uses regular expressions to read different types.

❖Java.io.BufferedReader class reads text from a character-input stream, buffering characters so as to provide for the efficient reading of sequence of characters

❖BufferedReader is synchronous while Scanner is not. BufferedReader should be used if we are working with multiple threads.

❖BufferedReader has significantly larger buffer memory than Scanner.

SCANNER CLASS /BUFFEREDREADER

❖The Scanner has a little buffer (1KB char buffer) as opposed to the BufferedReader (8KB byte buffer), but it’s more than enough.

❖BufferedReader is a bit faster as compared to scanner because scanner does parsing of input data and BufferedReader simply reads sequence of characters.