Character Stream Vs Byte Stream in Java

A **stream** is a way of sequentially accessing a file. In Streams you can process the data one at a time as bulk operations are unavailable with them. But, streams supports a huge range of source and destinations including disk file, arrays, other devices, other programs etc. In Java, a **byte** is not the same thing as a **char** . Therefore a byte stream is different from a character stream. So, Java defines two types of streams: **Byte Streams** and **Character Streams** .

Byte Streams

A **byte** stream access the file byte by byte. Java programs use byte streams to perform input and output of **8-bit** bytes. It is suitable for any kind of file, however not quite appropriate for text files. For example, if the file is using a **unicode encoding** and a character is represented with two bytes, the byte stream will treat these separately and you will need to do the conversion yourself. Byte oriented streams do not use any **encoding scheme** while Character oriented streams use character encoding scheme(UNICODE). All byte stream classes are descended from **InputStream** and **OutputStream** .

Example

Java Tutorial for Beginners
Java Fundamentals
Java Control Structures
Java Collections
Java String Class
Java File Class
Learn Java Programming
Java Network Programming
Java Errors and Exceptions
Java Interview Questions

```
import java.io.*;
public class TestClass{
  public static void main(String[] args) {
    FileInputStream fis = null;
    FileOutputStream fos = null;
    try {
      fis = new FileInputStream("in.txt");
      fos = new FileOutputStream ("out.txt");
      int temp;
      while ((temp = fis.read()) != -1) //read byte b
      fos.write((byte)temp);         //write byte by b
      if (fis != null)
      fis.close();
      if (fos != null)
      fos.close();
    }catch(Exception e){
      System.out.println(e);
    }
  }
}
```

When to use:

Byte streams should only be used for the most primitive I/O

When not to use:

You should not use Byte stream to read Character streams

e.g. To read a text file

Character Streams

A **character stream** will read a file character by character. Character Stream is a higher level concept than **Byte Stream** . A Character Stream is, effectively, a Byte Stream that has been wrapped with logic that allows it to output characters from a specific **encoding** . That means, a character stream needs to be given the file's encoding in order to work properly. Character stream can support all types of character sets ASCII, Unicode, UTF-8, UTF-16 etc. All character stream classes are descended from **Reader** and **Writer** .

Example

```
import java.io.*;
public class TestClass{
  public static void main(String[] args) {
    FileReader reader = null;
    try {
      reader = new FileReader("in.txt");
      int fChar;
      while ((fChar = reader.read()) != -1) //read ch
        System.out.println((char)fChar);          //w
    }catch(Exception e){
      System.out.println(e);
    }
  }
}
```

When to use:

To read character streams either from Socket or File of characters

Summary

- Character oriented are tied to datatype. Only string type or character type can be read through it while byte oriented are not tied to any datatype, data of any datatype can be read(except string) just you have to specify it.
- Character oriented reads character by character while byte oriented reads byte by byte.
- Character oriented streams use character encoding scheme(UNICODE) while byte oriented do not use any encoding scheme.
- Character oriented streams are also known as reader and writer streams Byte oriented streams are known as data streams-Data input stream and Data output stream.

Character-stream class	Corresponding byte class
Reader	InputStream
BufferedReader	BufferedInputStream
LineNumberReader	LineNumberInputStream
CharArrayReader	
InputStreamReader	(none)
FileReader	FileInputStream
FilterReader	FilterInputStream
PushbackReader	PushbackInputStream
PipedReader	PipedInputStream
StringReader	StringBufferInputStream
Writer	OutputStream
BufferedWriter	BufferedOutputStream
CharArrayWriter	ByteArrayOutputStream
FilterWriter	FilterOutputStream
OutputStreamWriter	(none)
FileWriter	FileOutputStream
PrintWriter	PrintStream
PipedWriter	PipedOutputStream
StringWriter	(none)

Next : How to append text to an existing file in Java

Java Interview Questions-Core Faq - 1

Java Interview Questions-Core Faq - 2

Java Interview Questions-Core Faq - 3

Important features of Java

Difference between Java and JavaScript?

What is the difference between JDK and JRE?

What gives Java its 'write once and run anywhere' nature?

What is JVM and is it platform independent?

What is Just-In-Time (JIT) compiler?

What is the difference between ArrayList and LinkedList?

What is the difference between List and Set in Java

Difference between HashSet and HashMap in Java

Difference between HashMap and Hashtable in Java?

How does the hashCode() method of java works?

Difference between capacity() and size() of Vector in Java

What is a Java ClassNotFoundException?

How to fix java.lang.UnsupportedClassVersionError

More Source Code : Search Mail to : feedback@net-informations.com