# IOSTREAMS

**In any programming language, in any application,providing input to the Applications and getting output from the Applications is essential.**

**In case of C and C++ applications,we are able to perform input and output operations by using some predefined library in the form of printf(),scanf(),cin>>,cout<<….**

**Similarly in Java Applications,to perform input and output operations we have to use streams.**

**Java has represented all the streams in the form of predefined classes in "java.io" package.**

# Stream:

**Stream is medium or channel, it will allow the data in continuous flow from input devices to java program and from Java program to output devices.**

**In Java IOStreams are divided into following ways:**

1. **Byte oriented Streams.**
2. **Character-Oriented Streams**
3. **Byte-Oriented Streams:**

**These are Streams, which will allow the data in the form of bytes from input devices to Java program and from java program to output devices.**

**The length of the data in byte-oriented streams is 1 byte.**

There are two types of Byte-Oriented Streams:

1. **InputStream**
2. **OutputStream**
3. **InputStream:**

**It is a byte-oriented Stream, it will allow data in the form of bytes from input devices to Java Applications.**

**EX:**

**ByteArrayInputStream**

**FilterInputStream**

**DataInputStream**

**ObjectInputStream**

**FileInputStream**

**StringBufferInputStream**

**BufferedInputStream….**

# OutputStream:

**It is a byte-oriented Stream, it will allow the data in the form of bytes from Java applications to output devices.**

**EX:**

**ByteArrayOutputStream**

**FilterOutputStream**

**DataOutputStream**

**FileOutputStream**

**PrintStream**

**BufferedOutputStream..**

**NOTE: All the ByteOrientedStream classes are terminated with "Stream" word.**

**NOTE: The length of data items in Byte Oriented Streams is 1 byte.**

# 2.Character-Oriented Streams:

**These are the Streams, which will allow the data in the form of characters from input devices to java program and form java program to output devices.**

**There are two types of character-oriented streams:**

1. **Reader**
2. **Writer**
3. **Reader:**

**It is a character-oriented stream, it will allow the data in the form of characters from input devices to java program.**

**EX:**

**CharArrayReader**

**FilterReader**

**BufferedReader**

**FileReader**

**InputStreamReader….**

1. **Writer:**

**It is a character-oriented stream, it will allow the data in the form of characters from java program to output devices.**

**EX:**

**CharArrayWriter**

**FilterWriter**

**FileWriter**

**PrintWriter**

**BufferedWriter….**

**NOTE: All the predefined Classes of character-oriented streams are terminated with either Reader or Writer.**

**NOTE: The length of the data in characters-oriented stream is 2 bytes.**

# FileOutPutStream:

**It is byte-oriented Stream, it can be used to transfer the data from Java program to a particular target file.**

**To transfer the data from Java program to a particular target file by using FileOutPutstream we have to use the following Steps.**

1. **Create FileOutPutStream between Java program and target file:**

**If we want to create FileOutPutStream class object then we have to use the following constructors :**

**public FileOutPutStream(String target\_File)**

**public FileOutPutStream(String target\_File,boolean b)**

**EX:**

**FileOutPutStream fos=new FileOutPutStream("abc.txt");**

--> **It will override the existed data in the target file at each and every write operation.**

**FileOutPutStream fos=new FileOutPutStream("abc.txt",true);**

**--> It will not override the existed data in the target file, it will append the specified new data to the existed data in the target file.**

**When JVM encounters the above instruction, JVM will perform the following tasks.**

1. **JVM will take the specified target file.**
2. **JVM will search for the specified target file at the respective location.**
3. **If the specified target file is available then JVM will establish FileOutPutStream from java program to target file.**
4. **If the specified target file is not available then JVM will create a file with the target file name and establish FileOutPutStream from Java program to target file.**
5. **Declare the data and convert into byte[]:**

**String data="Hello";**

**byte[] b=data.getBytes();**

1. **Write Byte Array data into FileOutPutStream:**

**To write byte[] data into FileOutPutStream,we have to use the following method.**

**public void write(byte[] b) throws IOException**

**EX:**

**fos.write(b);**

1. **Close FileOutPutStream:**

**fos.close();**

# FileInputStream:

**It is a byte-oriented Stream, it can be used to transfer the data from a particular source file to Java Program.**

**If we want to transfer the data from source file to java program, by using FileInputStream,we have to use the following Steps:**

1. **Create FileInputStream class Object:**

**To create FileInputStream class object, we have to use the following constructor from java.io.FileInputStream class.**

**public FileInputStream(String file\_name) throws FileNotFoundException**

**EX:**

**FileInputStream fis=new FileInputStream("abc.txt");**

**When JVM encounters the above instruction then JVM will perform the following actions:**

1. **JVM will take the specified source file name.**
2. **JVM will search for the specified source file at the respective location.**
3. **If the source file is not available at the respective location then JVM will raise an excpetion like "java.io.FileNotFoundException".**
4. **If the required source file is available then JVM will establish FileInputStream from source file to JAVA program.**
5. **After creating FileInputStream, JVM will transfer the data from source file to FileInputStream in the form bytes.**
6. **Get the size of the data from FileInputStream and prepare byte[] with the data size: To get the size of the data from FileInputStream,we have to use the following method**

**public int available()**

**EX:**

**int size=fis.available();**

**byte[] b=new byte[size];**

**Read the data from FileInputStream into byte[]:**

**To read the data from FileInputStream into byte[],we have to use the following method.**

**public void read(byte[] b)throws IOException**

**EX:**

**fis.read(b);**

4. **Convert data from byte[] to String:**

**String data=new String(b);**

**System.out.println(data);**

5.**close FileInputStream:**

**fis.close();**

**Write a Java program to display a particular file content on command prompt by taking filename as command line input?**

**import java.io.\*;**

**class DisplayEx{**

**public static void main(String args[]) throws Exception{**

**String file\_Name=args[0];**

**FileInputStream fis=new FileInputStream(file\_Name);**

**int size=fis.available();**

**byte b[]=new byte[size];**

**fis.read();**

**String data=new String(b);**

**System.out.println(data);**

**fis.close();**

**}**

**}**

**Write a Java program to count no. of words available in a particular text file and how many times the word "Hk" is repeated?**

**import java.io.\*;**

**import java.util.\*;**

**class Word\_Count\_Ex{**

**public static void main(String args[]) throws Exception{**

**FileInputStream fis=new FileInputStream("abc.txt");**

**int size=fis.available();**

**byte b[]=new byte[size];**

**fis.read();**

**String data=new String(b);**

**StringTokenizer st=new StringTokenizer(data);**

**int tokens=st.countTokens();**

**System.out.println("No of words :"+tokens);**

**int count=0;**

**while(st.hasMoreTokens()){**

**String token=st.nextToken();**

**if(token.equals("Hk")){**

**count=count+1;**

**}**

**}**

**System.out.println("'Hk' is repeated :"+count);**

**fis.close();**

**}**

**}**

**Write a Java program to copy an image from a source file to a particular target file?**

**import java.io.\*;**

**public class Image\_Copy\_Ex{**

**public static void main(String args[]){**

**FileInputStream fis=new FileInputStream(“E:/xyz.jpg”);**

**int size=fis.available();**

**byte[] b=new byte[size];**

**fis.read(b);**

**FileOutPutStream fos=new FileOutPutStream("F:/abc.jpg");**

**fos.write(b);**

**fis.close();**

**fos.close();**

**}**

**}**

# FileWriter:

**This character-oriented Stream can be used to transfer the data from Java application to a particular target File.**

**If we want to transfer the data from java application to a particular target file by using FileWriter then we have to use the following steps:**

1. **Create FileWriter object:**

**To create FileWriter class object,we have to use the following constructor.**

**public FileWriter(String target\_File)**

**EX: FileWriter fw=new FileWriter("abc.txt");**

-->**It will override the existed content with the new content at each and every write operation.**

**public FileWriter(String target\_File,boolean b)**

**EX:FileWriter fw=new FileWriter("abc.txt",true);**

-->**It will append new content to the existed content available in the file at each and every write operation.**

**When JVM encounters the above instructions, JVM will take the specified file and JVM searches for the specified file at the respective location, if the required target file is available then JVM will establish FileWriter from Java application to the target file. If the required target file is not available at the respective location then JVM will create a new file with the same specified file name and establish FileWriter from Java application to the target file.**

1. **Declare the data which we want to transfer and convert that data into char[]:**

**String data="Hello";**

**char[] ch=data.toCharArray();**

1. **Write char[] data into FileWriter:**

**To write char[] data into FileWriter,we have to use the following method.**

**public void write(char[] ch)throws IOException**

**EX: fw.write(ch)**;

4.**Close FileWriter:**

**fw.close();**

**EX:**

**import java.util.\*;**

**public class FileWriterEx{**

**public static void main(String args[])throws Exception{**

**FileWriter fw=new FileWriter("abc.txt",true); String data="Brain Coders Solutions";**

**char[] ch=data.toCharArray();**

**fw.write(ch);**

**fw.close();**

**}**

**}**

# FileReader:

**This character-oriented stream can be used to transfer the data from a particular source file to Java program.**

**If we want to transfer the data from a particular source file to Java program by using FileReader then we have to use the following steps:**

1. **Create FileReader class Object:**

**To create FileReader class object,we have to use the following constructor.**

**public FileReader(String file\_Name)throws FileNotFoundException**

**EX:**

**FileReader fr=new FileReader("abc.txt");**

**when JVM encounters the above instruction,JVM will perform the following steps.**

1. **JVM will take source file name from FileReader constructor.**
2. **JVM will check whether the specified file is available or not at the respective location.**
3. **If the specified source file is not available at the respective location then JVM will rise an exception like "java.io.FileNotFoundException".**
4. **If the specified file is existed at the respective location then JVM will establish FileReader from source file to Java program.**
5. **After creating FileReader,JVM will transfer the content of source file to FileReader object in the form of characters.**
6. **Read data from FileReader:**

**To read data from FileReader,we have to use the following steps.**

1. **Read character by character from FileReader in the form of ASCII values.**
2. **Convert that ASCII values into the respective characters.**
3. **Append the converted characters to a String variable.**

**Repeat the above steps upto all the characters which are available in the respective source file or upto the end-of-file character i.e "-1".**

**To read an ASCII value from FileReader,we have to use the following method.**

**public int read() throws IOException**

**3.Close FileReader:**

**fr.close();**

**EX:**

**import java.util.\*;**

**public class FREx{**

**public static void main(String args[])throws Exception{**

**FileWriter fr=new FileWriter("abc.txt");**

**String data="";**

**int val=fr.read();**

**while(val!=-1){**

**data=data+(char)val;**

**val=fr.read();**

**}**

**System.out.println(data);**

**fr.close();**

**}**

**}**

**Write a JAVA programme to copy a document from one file to another file by using character oriented Streams?**

**import java.io.\*;**

**public class FileCopyEx{**

**public static void main(String args[])throws Exception{**

**FileReader fr=new FileReader("hibernatecgf.xml");**

**String data="";**

**int val=fr.read();**

**while(val!=-1){**

**data=data+(char)val;**

**val=fr.read();**

**}**

**char[] ch=data.toCharArray();**

**FileWriter fw=new FileWriter("abc.xml");**

**fw.write(ch);**

**fr.close();**

**fw.close();**

**}**

**}**

# BufferedReader:

**If we want to take dynamic input by using BufferedReader in java applications then we have to use the following statement.**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**where "in" is static variable, it will refer a predefined "InputStream" object which is connected with command prompt.**

**If we provide data on command prompt then that data will be transfered to InputStream object in the form of binary data.**

**where "InputStreamReader can be used to convert the data from binary representation to character representation.**

**where BufferedReader can be used to improve the performance of Java application while performing input operation.**

**To read the data from BufferedReader,we will use the following method**

1**.readLine()**

1. **read()**

Q)**What is the difference between readLine() method and read() method?**

**Ans:**

**readLine() method will read a line of text from command prompt[BufferedReader] and it will return that data in the form of String.**

**public String readLine() throws IOException**

**read() method will read a single character from command prompt[BufferedReader] and it will return that character in the form of its ASCII value.**

**public int read()throws IOException**

**EX:**

**import java.io.\*;**

**public class BufferedReaderEx{**

**public static void main(String args[])throws Exception{**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**System.out.println("Enter Text :");**

**String data1=br.readLine();**

**System.out.println("Enter the same text again :");**

**int data2=br.read();**

**System.out.println("First Entered :"+data1);**

**System.out.println("Second Entered :"+data2+"--->"+(char)data2);**

**}**

**}**

**Consider the following programme:**

**import java.io.\*;**

**public class BufferedReaderEx{**

**public static void main(String args[])throws Exception{**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**System.out.println("First value :");**

**String val1=br.readLine();**

**System.out.println("Second value :");**

**String val2=br.readLine();**

**System.out.println("Addition :"+val1+val2);**

**}**

**}**

**If we provide 10 and 20 as dynamic input to the above program then the above program will display "1020" value instead of 30 that is the above program has performed String concatenation instead of performing Arithmetic Addition because br.readLine() method has return 10 and 20 values in the form String data.**

**In the above program, if we want to perform Arithmetic operations over dynamic input then we have to convert String data into the respective primitive data ,for this we have to use Wrapper Classes.**

**ThereFore,BufferedReader dynamic input approach is depending on wrapper classes while reading primitive data as dynamic input.**

**EX:**

**import java.io.\*;**

**public class BufferedReaderEx{**

**public static void main(String args[])throws Exception{**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**System.out.println("First value :");**

**String val1=br.readLine();**

**System.out.println("Second value :");**

**String val2=br.readLine();**

**int f\_Val=Integer.parseInt(val1);**

**int s\_Val=Integer.parseInt(val2);**

**System.out.println("Addition :"+(f\_Val+s\_Val));**

**}**

**}**

# OutputStreamWriter:

**An OutputStreamWriter is a bridge from character streams to byte streams.**

**Characters written to it are encoded into bytes using a specified charset.**

**The charset that it uses may be specified by name or may be given explicitly, or the platform’s default charset may be accepted.**

**Each invocation of a write() method causes the encoding converter to be invoked on the given character(s).**

**The resulting bytes are accumulated in a buffer before being written to the underlying output stream.**

**The size of this buffer may be specified, but by default it is large enough for most purposes.**

**Note that the characters passed to the write() methods are not buffered.**

# InputStreamReader

**An InputStreamReader is a bridge from byte streams to character streams.**

**It reads bytes and decodes them into characters using a specified charset.**

**The charset that it uses may be specified by name or may be given explicitly, or the platform’s default charset may be accepted.**

**Each invocation of one of an InputStreamReader’s read() methods may cause one or more bytes to be read from the underlying byte-input stream.**

**OutputStreamWriter is a [class](https://www.javatpoint.com/object-and-class-in-java) which is used to convert character stream to byte stream, the characters are encoded into byte using a specified charset.**

**write() method calls the encoding converter which converts the character into bytes.**

**The resulting bytes are then accumulated in a buffer before being written into the underlying output stream.**

**The characters passed to write() methods are not buffered.**

**We optimize the performance of OutputStreamWriter by using it with in a BufferedWriter so that to avoid frequent converter invocation.**

**public class OutputStreamWriterExample {**

**public static void main(String[] args) {**

**try {**

**OutputStream outputStream = new FileOutputStream("output.txt");**

**Writer outputStreamWriter = new OutputStreamWriter(outputStream);**

**outputStreamWriter.write("Hello World");**

**outputStreamWriter.close();**

**} catch (Exception e) {**

**e.getMessage();**

**}**

**}  }**

# Java InputStreamReader

**An InputStreamReader is a bridge from byte streams to character streams.**

**It reads bytes and decodes them into characters using a specified charset.**

**The charset that it uses may be specified by name or may be given explicitly, or the platform's default charset may be accepted.**

**public class InputStreamReaderExample {**

**public static void main(String[] args) {**

**try  {**

**InputStream stream = new FileInputStream("file.txt");**

**Reader reader = new InputStreamReader(stream);**

**int data = reader.read();**

**while (data != -1) {**

**System.out.print((char) data);**

**data = reader.read();**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**