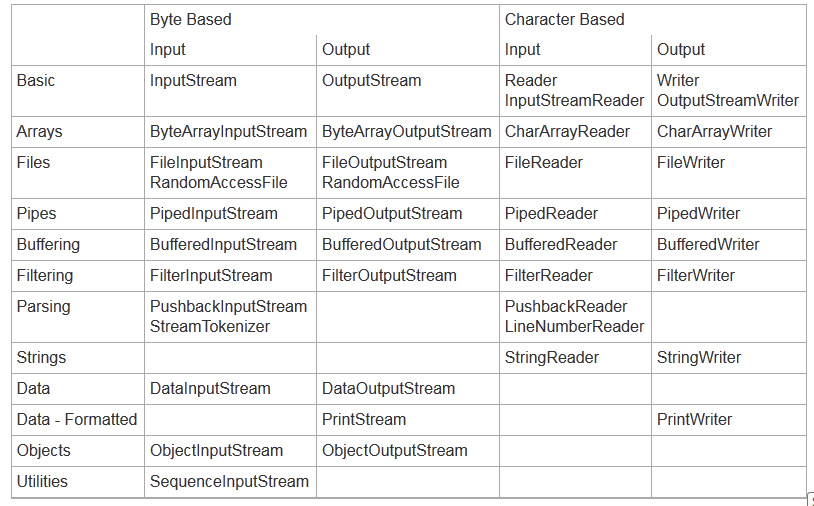
**JAVA I/O Classification based on Byte Based and Character Based**



**DataInputStream:**

The Java DataInputStream class enables you to read Java primitives (int, float, long etc.) from an InputStream instead of only raw bytes. You wrap an InputStream in a DataInputStream and then you can read Java primitives from the DataInputStream.

The DataInputStream is handy if the data you need to read consists of Java primitives larger than one byte each, like int, long, float, double etc. The DataInputStream expects the multi byte primitives to be written in network byte order.

**Syntax of DataInputStream:**

**public** DataInputStream(InputStream instream)

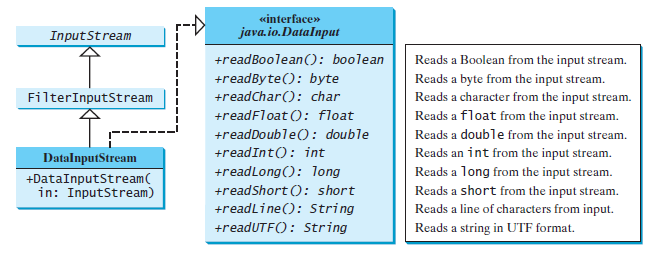
**public** DataOutputStream(OutputStream outstream)

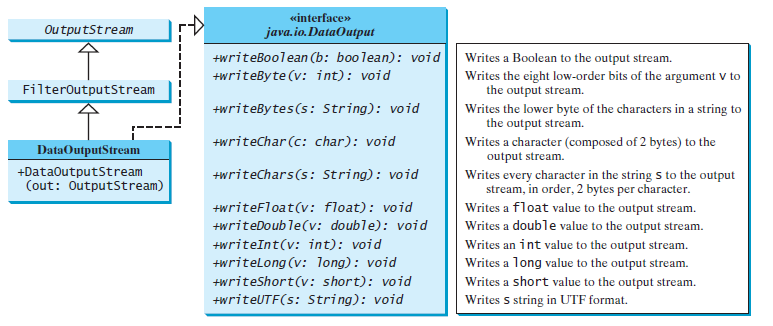
The statements given below create data streams. The first statement creates an input stream

for file **in.dat**; the second statement creates an output stream for file **out.dat**.

DataInputStream input = (**new** FileInputStream(**"in.dat"**)); //creates InputStream

DataOutputStream output = (**new** FileOutputStream(**"out.dat"**)); //Creates OutputStream



****

**Program 1:**

Write a Java Program to write Java Primitive data-type values using DataOutputStream and read Java Primitive values using DataInputStream Class.

|  |
| --- |
| package javaiodemo;  import java.io.DataInputStream;  import java.io.DataOutputStream;  import java.io.DataInput;  import java.io.DataOutput;  import java.io.FileInputStream;  import java.io.FileOutputStream;  public class DataInDemo {  public static void main(String[] args) throws Exception  {  DataOutputStream out=new DataOutputStream(new FileOutputStream("abc1.txt"));  DataInputStream in=new DataInputStream(new FileInputStream("abc1.txt"));    out.writeInt(24);  out.writeFloat(1.4F);  out.writeLong(1222);  out.writeUTF("amit");  out.close();  int invalue1=in.readInt();  System.out.println(" "+invalue1);  float f=in.readFloat();  System.out.println(" "+f);  long doublevalue1=in.readLong();    System.out.println(" "+doublevalue1);  String s=in.readUTF();  System.out.println(" "+s);  in.close();  }  } |

**BufferedInputStream**/**BufferedOutputStream** can be used to speed up input and output by reducing the number of reads and writes. **BufferedInputStream**/ **BufferedOutputStream** does not contain new methods. All the methods in **BufferedInputStream**/**BufferedOutputStream** are inherited from the **InputStream**/ **OutputStream** classes. **BufferedInputStream**/**BufferedOutputStream** adds a buffer in the stream for storing bytes for efficient processing.

**Program 1:**

Program to Write the contents of one to another File using BufferedReader class and Printwiter class.

|  |
| --- |
| package javaiodemo;  import java.io.PrintWriter;  import java.io.BufferedReader;  import java.io.\*;  public class BufferReaderExample {  public static void main(String[]args)throws Exception  {  PrintWriter o;  BufferedReader i;  String lineone;  String linetwo;  String linethree;  i = new BufferedReader(new FileReader("b12.dat"));  o=new PrintWriter(new FileWriter("c12.dat"));  lineone =new String(i.readLine());  linetwo =new String(i.readLine());  i.close();  System.out.println(" "+lineone);  System.out.println(" "+linetwo);  o.print(lineone);  o.print(linetwo);  o.close();  }  } |