**Java IO: Streams:** is a logical connection between java program and File.

* Java IO streams are flows of data we can either read from or write to (Java IO streams are typically either byte-based or character-based).
* Streams are typically connected to a data source or destination. (like a file, network connection etc.)

**File class**: Java File Object. (createNewFile used to create a file/ mkdir to create directory)

File f=**new** File(**"C:\\Users\\sball\\Desktop\\IOSample\\t1.txt"**);  
System.***out***.println(f.exists());  
f.createNewFile();  
File f1=**new** File(**"C:\\Users\\sball\\Desktop\\IOSample\\mynewdir"**);  
f1.mkdir();

**FileWriter**: to write a data to file.

[Note: **flush()** method is **mandatory and** it is from **Flushable (I)** because when we wrie data into stream the data is not stored in file but it is stored in OutputStream memory when we use flush then it stored in File and **close()** from **Closable(I)**]

FileWriter fw=new FileWriter(filename); - - > will override the content of file

FileWriter fw1=new FileWriter(t1.txt); - - > will override the content of file

fw.write(100);

FileWriter fw=new FileWriter(filename, boolean append); - - > will appends to existing content.

FileWriter fw1=new FileWriter(t1.txt, boolean append); - - > will appends to existing content. file

Eg:

FileWriter fw=**new** FileWriter(**"C:\\Users\\sball\\Desktop\\IOSample\\t5.txt"**, **true**);  
fw.write(100 );  
fw.write(**"\n"**);  
**char**[] arr={**'w'**,**'e'**};  
fw.write(arr);  
fw.write(**"\n"**);  
String s=**"hello world"**;  
fw.write(s);  
fw.flush();  
fw.close();

[Note: while using FileWriter we need to use line separator “\n”, in order solve this we use BufferedWriter]

**BufferedWriter**:[similar to FileWriter but newLine() method is additional and BufferedWriter cannot communicate with File directly so new require FileWriter]

FileWriter f4=**new** FileWriter(**"C:\\Users\\sball\\Desktop\\IOSample\\t10.txt"**);  
BufferedWriter bw=**new** BufferedWriter(f4);

FileReader: to read data from file. [ int read() returns Unicode value until no char then we will get -1 and should be typecasted, int read( char[] and close() methods ]

FileReader fr=**new** FileReader(**"C:\\Users\\sball\\Desktop\\IOSample\\t10.txt"**);  
**int** i=fr.read();  
System.***out***.println(i);  
**while**(i != -1)  
{   
 System.***out***.print((**char**)i);  
 i=fr.read();  
}  
fr.close();

Or

File f=**new** File(**"C:\\Users\\sball\\Desktop\\IOSample\\t10.txt"**);  
 FileReader fr=**new** FileReader(f);  
 **char**[] ch=**new char**[(**int**)f.length()];  
 fr.read(ch);  
 **for**(**char** ch1:ch)  
 {  
 System.***out***.print(ch1);  
 }  
 fr.close();

**BufferedReader**: [ using BufferedReader we can read data character by character and Line by Line using readLine() method. After reading all file we will get null. It cannot communicate directly with file, it requires a Reader object].

FileReader fr=**new** FileReader(**"C:\\Users\\sball\\Desktop\\IOSample\\t10.txt"**);  
BufferedReader br= **new** BufferedReader(fr);  
String line=br.readLine();  
**while**(line != **null**) {  
 System.***out***.println(line);  
 line=br.readLine();  
}  
br.close();

PrintWriter: [ similar to BufferedWriter but apart from write() method it provides print() method and printLine() methods available].

**Important: [normal text data and binary data (means video file, audio file etc.)].**

* **Reader and writer are for character and text data.**
* If we want to handle Character data/text data we should go for Reader and Writer.

**Stream concept is for binary data.**

* If we want handle binary data, we have to go for Streams.

InputStream – reading binary data from a source.

OutputStream – write to binary data to destination.