The **Reade**r class reads txt files and **Input -stream** reads Binary data. Is this the only difference present or is there any other view in the difference aspect.

**BufferedReader** reads characters (text), whereas the **BufferedInputStream** reads raw bytes example(The data are written as a stream of bytes. There are no gaps or markers that say where one value ends and another begins)

How to create file and check it exixt or not

File f =new File("./akash.txt");

System.out.println(f.exist());

**Append File to the Exiting File:-**

public static void main(String []args) throws IOException

{

FileWriter fw = new FileWriter("./akash.txt", **true**);//Must write True for Append

BufferedWriter br = new BufferedWriter(fw);

br.write("From Akola");

br.close();

}

FileWriterExample

1. {
2. **public** **static** **void** main(String args[]){
3. **try**{
4. FileWriter fw=**new** FileWriter("D:\\testout.txt");
5. fw.write("Welcome to javaTpoint.");
6. fw.close();
7. }**catch**(Exception e){System.out.println(e);}
8. System.out.println("Success...");
9. }

 FileReaderExample

1. {
2. **public** **static** **void** main(String args[])**throws** Exception{
3. FileReader fr=**new** FileReader("D:\\testout.txt");
4. **int** i;
5. **while((i=fr.read())!=-1)**  //Most important
6. System.out.print((**char**)i);
7. fr.close();
8. }

BufferedWriterExample

1. {
2. **public** **static** **void** main(String[] args) **throws** Exception {
3. FileWriter writer = **new** FileWriter("D:\\testout.txt");
4. BufferedWriter buffer = **new** BufferedWriter(writer);
5. buffer.write("Welcome to javaTpoint.");
6. buffer.close();
7. System.out.println("Success");
8. }
9. }

BufferedReaderExample

1. {
2. FileReader fr=**new** FileReader("D:\\testout.txt");
3. BufferedReader br=**new** BufferedReader(fr);
5. **int** i;
6. **while**((i=br.read())!=-1){
7. System.out.print((**char**)i);
8. }
9. br.close();
10. fr.close();
11. }

**Important Points:**

1.FileInputStream is used for reading binary files.

2.FileInputStream [reads bytes from a file](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using.html)( where Reader is used to character input/output)

3.Stream is used to binary input/output.

4.So convert string into byte for writing into file

Example:

String s="I love you Akash";

byte b[]=s.getBytes();

fout.write(b);

FileOutputStream

public static void main(String []args) throws IOException

{ FileOutputStream fout=new FileOutputStream("./akash.txt",true);

String s="I love you Nandini";

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

fout.write(b);

fout.close();

System.out.println("success");

}

BufferedOutputStreamExaple

{

     FileOutputStream fout=**new** FileOutputStream("D:\\testout.txt");

     BufferedOutputStream bout=**new** BufferedOutputStream(fout);

     String s="Welcome to javaTpoint.";

**byte b[]=s.getBytes();**

     bout.write(b);

     bout.flush();

     bout.close();

     fout.close();

     System.out.println("success");

FileInputStream

public static void main(String []args) throws IOException

{

FileInputStream fs= new FileInputStream("./akash.txt");

BufferedInputStream br = new BufferedInputStream(fs);

int i;

while**((i=br.read())!=-1) // Most Important**

{

System.out.print((char)i);

}

br.close();

BufferedInputStreamExample

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

**try**{

    FileInputStream fin=**new** FileInputStream("D:\\testout.txt");

    BufferedInputStream bin=**new** BufferedInputStream(fin);

**int** i;

**while((i=bin.read())!=-1)//Most Important**{

     System.out.print((**char**)i);

    }

    bin.close();

    fin.close();

  }**catch**(Exception e){System.out.println(e);}

 }

}

|  |  |  |
| --- | --- | --- |
|  | **BufferedReader** | **FileReader** |
| 1 | BufferedReader is buffered. | FileReader is not buffered. |
| 2 | BufferedReader [reads characters from another Reader (Eg - FileReader)](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using_42.html) | FileReader [reads characters from a file](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using_72.html). |
| 3 |  |  |

|  |  |  |
| --- | --- | --- |
|  | **BufferedInputStream** | **FileInputStream** |
| 1 | BufferedInputStream is buffered. | FileInputStream is not buffered. |
| 2 | BufferedInputStream [reads bytes from another InputStream (Eg - FileInputStream)](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using_76.html)  FileInputStream fis = new FileInputStream("c:/myFile.txt");  BufferedInputStream bis = new BufferedInputStream(fis); | FileInputStream [reads bytes from a file](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using.html).  FileInputStream fis = new FileInputStream("c:/myFile.txt"); |

**FileInputStream and FileReader**

|  |  |  |
| --- | --- | --- |
| 1 | Stream is Byte Based, it can be [used to read bytes](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using.html) or [write bytes](http://www.javamadesoeasy.com/2015/08/program-to-write-string-to-file-using.html). | Reader is Character Based, it can be [used to read](http://www.javamadesoeasy.com/2015/08/program-to-read-text-from-file-using_72.html) or [write characters](http://www.javamadesoeasy.com/2015/08/program-to-write-to-file-using.html). |
| 2 | Stream is used to binary input/output | Reader is used to character input/output |
| 3 | FileInputStream is Byte Based, it can be used to read bytes. | FileReader is Character Based, it can be used to read characters. |
| 4 | FileInputStream is used for reading binary files. | FileReader is used for reading text files in platform default encoding.  Now question comes comes, What is Platform default encoding?  Platform default encoding means the encoding used by the operating system on which  the JVM is running.  Example - Platform default encoding is user specific setting.  Windows (in US) - it's often CP1250,  Windows (in Europe) - it's often CP1252.  Windows (in China) - it's often Big5.  On many other windows and  Linux systems - it's often UTF-8.  On Macs - it’s often MacRoman. |
| 5 | FileInputStream and ObjectInputStream can be used for [Serialization and DeSerialization](http://www.javamadesoeasy.com/2015/02/serialize-and-deserialize-object.html), where serialized object can be persisted in file. In Serialization object is converted into byte stream and in deserialization it is converted back from byte to object. | FileReader is not used for Serialization and DeSerialization, as it reads characters not bytes. |
| 6 | FileInputStream.read()  reads 1 byte (8-bit) at a time. | FileReader.read() reads 2 bytes(16-bit) at a time, because [char is 16-bit data type](http://www.javamadesoeasy.com/2015/06/primitive-customreference-data-types.html). |
| 7 | FileInputStream must be used when we are reading audio, video or other multimedia files. | FileReader must be used when we are reading text files, pdfs or word documents. |

**Copy one file into another File:-**

**1.Program: Using FileReader and Writer**

public static void main(String []args) throws IOException

{

FileReader fr=new FileReader("./akash.txt");

FileWriter fw=new FileWriter("./akash1.txt",true);

int i;

while((i=fr.read())!=-1)

{

fw.write((char)i);

}

fr.close();

fw.close();

System.out.println("Success...");

}

**2.Program:- Using FileInputStream**

public static void main(String []args) throws IOException

{

FileInputStream fr=new FileInputStream("./akash.txt");

FileOutputStream fw=new FileOutputStream("./akash1.txt",true);

int i;

while((i=fr.read())!=-1)

{

fw.write(i);// *No Need to do type casting like - fw.write((char)i);*

*// bz we write already binary data in the file.*

}

fr.close();

fw.close();

System.out.println("Success...");

}

**2.Program:-**

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

FileInputStream fr=new FileInputStream("./akash.txt");

FileOutputStream fw=new FileOutputStream("./akash1.txt",true);

int i;

byte b[]= new byte [200];

while((i=fr.read(b))!=-1)

{

fw.write(b,0,i);}