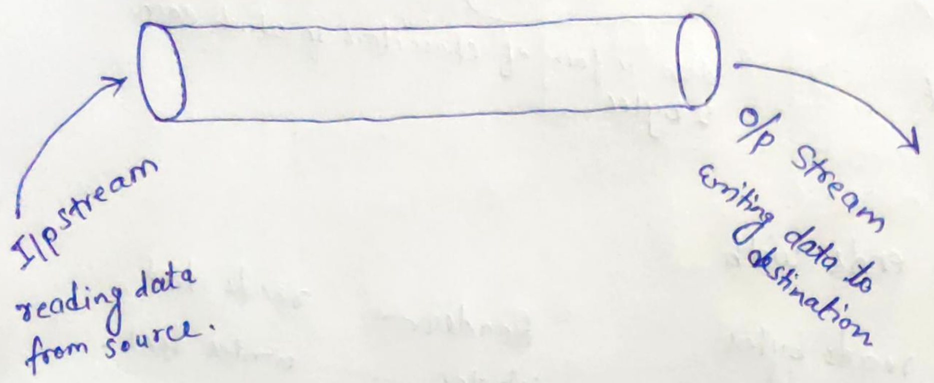
FILE OPERATIONS in Java

Stream : transporting data from one place to another.



java. io package

import java. io. *;

DataInputStream dis = new DataInputStream(System. in);

object

Keyboard is represented

System class has 3 fields — in, out, err

System. in represents InputStream object. [Keyboard]

System. out represents PrintStream object. [monitor]

Stream

byte Stream

data in form of bytes (binary form)

text Stream

data in form of charectors in which each 2 bytes

if a clau name ends with

InputStream - reads bytes

OutputStream - writes bytes

Reader - reads text.

Writer - writes text.

Byte Streams are used to handle

text

images

audio

video

files.

InputStream
class

- ByteArrayInputStream
- FileInputStream
- PipedInputStream
- FilterInputStream
  - BufferedInputStream
  - InflaterInputStream
  - LineNumberInputStream
  - DataInputStream.
- ObjectInputStream

Byte Stream Class for reading data.

OutputStream

- ByteArrayOutputStream
- FileoutputStream
- FilterOutputstream
- **PipedOutputStream**
  - BufferedOutputStream
  - DeflaterOutputStream
  - PrintStream
  - DataOutputStream
- ObjectOutputStream

byte Stream classes for writing data

---

text stream classes for readig data

Reader

- BufferedReader — **LineNumberReader**
- CharArrayReader
- FilterReader
- **InputStreamReader** — **FileReader**
- PipedReader
- StringReader

text stream classes

Writer

- BufferedWriter
- CharArrayWriter
- filterWriter
- **OutstreamWriter** → **FileWriter**
- PipedWriter
- PrintWriter
- StringWriter.

How to create text files in Java ?

FileOutputStream

It is a class belongs byte Stream
It stores data in form of individual bytes

It is used to create text file

Step 1 Code for using DataInputStream class for reading data from Keyboard —

```
DataInputStream dis=new DataInputStream(System.in);
```

Step2 Attach the file where data is to be stored
Code —

```
FileOutputStream fout=new FileOutputStream("myfile.txt")
```

this class is used to send data to a file

object

file in which you want to store data.

Step3: Read data from DataInputStream and write it into FileOutputStream

Read data from dis object & write it into fout object

```
ch=(char)dis.read();
fout.write(ch);
```

Step 4: file should be closed after performing i/p or o/p operation

to ensure saftey of data

```
fout.close();
```

It will close the FileOutputStream

W.A.P. which shows how to read data data from Keyboard and write it to any file myfile.txt.

```
import java.io.*;
Class    Create_File_Ex
    {
      psum(String[] arg)
      throws  IOException
      {
        DataInputStream dis = new DataInputStream(System.in);
     FileOutputStream  fout = new FileOutputStream("myfile.~~txt~~");
     S.o.pln("Enter text (@ at the end): ");
     Char ch;
    while ((ch=(char) dis. read()) != '@')
     fout.write(Ch);
     fout.close();

  }
}
```

```
javac Create_file_ex.java
java Create_file_ex
```

The previous data in the myfile.txt get lost whenever we recompile or reexequte our program.

So if we don't want to loose our data and want to append our new data to previous one existing data we should code in our program like

```
FileOutputStream fout = new FileOutputStream("myfile.txt",
                                             true);
```

after using true along with the file.name all previous data will be preserved.

In to Entry

improving Efficiency of reading / writing to a file :

```
import java.io.*;
Class CreateFile
 {
   psum( String [] arg )
   throws IOException
   {
  DataInputStream dis = new DataInputStream (System. in)
  DataInputStream dis = new DataInputStream (System. in);
File OutputStream fout = new FileOutput stream ("myfile.txt", true);
BufferedOutputStream bout = new BufferedOutputStream(fout, 1024);
```

Size of the buffer memory in bytes

```
S.o.pln( " Enter text ( @ at the end ):" );
 Char ch;
while(( ch = ( char) dis. read ( ) ) != '@' )
  bout . write( Ch );
  bout . close ( );
 }
}
```

Write a program that will read data from a file and display it on monitor

```
import java.io.* ;
class ReadFile
  { psum(String[] arg)
    throws IOException
    {
  File~~O~~InputStream fin=new FileInputStream("myfile.txt");
 S.o.pln("file Contents are:");

int ch;
 while((ch=fin.read() != -1)
   Sop( (char) ch);
   fin.close();

  }
}
```

above program works with dedicated file myfile.txt only
To make thas program work with any file
We use $\boxed{\text{Buffered Reader}}$ class.

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

String fnam = br.readline ( );
FileInputStream fin = new FileInputStream (fname);
```

Suppose if file is not there with fname name then try catch

```
try {
      fin = new fileInputStrean( fname );
  }
catch ( FileNotFound Exceptionfe)
    {
      S.o.pln ("file not found");
      return;
    }
```

W.A.P. which is used to read data from any text file.

```
import java.io.*;
Class ReadFile
 { psum(String[] arg)
  throws IOException
  {
  BufferedReader br = new BufferedReader(
                        new InputStreamReader(System.in));

  S.o.pln("Enter file name:");
  String fname = br.readLine();
 FileInputStream fin = null;
  try {   fin = new FileInputStream(fname);
     }
 catch(FileNotFoundException fe)
    { S.o.pln("file not found");
      return();
    }
 BufferedInputStream bin=new BufferedInputStream(fin);
  S.o.pln("file Contents are:");
  int ch;
   while((ch= bin.read())!=-1)
    S-o.p((char)Ch);
    bin.close();
 } }
```

In to Entry)

File Class

File class is found in java.io package

Step 1: Create a file class object

```
File obj = new File("path", filename);
```

File class Methods

```
boolean isFile()

boolean isDirectory()

boolean canRead()

boolean canWrite()

boolean canExecute()

boolean exist()

String getParent()

String getPath()

String getAbsolutePath()
```

long length( )

boolean delete( )

boolean createNewFile ( )

boolean mkdir( )

boolean renameTo(File filename)

String [ ] list( )