File Handling in Java

Java

www.educba.com

Stream

- A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

OutputStream

- Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.

InputStream

- Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.

Java FileInputStream

- **Java FileInputStream** class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

Java FileInputStream class declaration

public class FileInputStream **extends** InputStream

Methods:

- int read()--→It is used to read the byte of data from the input stream.
- void close()-→It is used to closes the stream.

Java FileInputStream Example 1: read single character

```
import java.io.FileInputStream;
public class DataStreamExample {
    public static void main(String args[]){
        try{
          FileInputStream fin=new FileInputStream("D:\\example.txt");
          int i=fin.read();
          System.out.print((char)i);
                fin.close();
         }catch(Exception e)
              {System.out.println(e);}
        }
       }
```

Note:

Before running the code, a text file named as **“example.txt"** is required to be created.

Java FileInputStream example 2: read all characters

```
class ReadName {
public static void main(String[] args) {
try {
        FileInputStream fin=new FileInputStream("C:\\\\programs\\\\example.txt");
        try {
        int ch=fin.read();
        while(ch!=-1)
        {
        System.out.print((char)ch);
         ch=fin.read();
        }}
        catch(IOException e)  {
        System.out.println("wrong method");
        }}
catch(FileNotFoundException e){
System.out.println("file not found");
}}
```

Java FileOutputStream Class

Java FileOutputStream is an output stream used for writing data to a file. If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class.

FileOutputStream class declaration

public class FileOutputStream **extends** OutputStream .

Methods:

void write(int b)→It is used to write the specified byte to the file output stream.

void close()→

It is used to closes the file output stream.

Java FileOutputStream Example 1: write byte

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
    public static void main(String args[]){
          try{
            FileOutputStream fout=new FileOutputStream("D:\\Demo.txt");

            fout.write(65);
            fout.close();
            System.out.println("success...");
           }catch(Exception e){
            System.out.println(e);
}    }    }
```

Java PrintWriter

- Java PrintWriter class is the implementation of Writer class. It is used to print the formatted representation of objects to the text-output stream.

Class declaration

public class PrintWriter **extends** Writer

Methods:

1.void println(boolean x)→ It is used to print the boolean value

2. void println(char[] x)→ It is used to print an array of characters

3. void println(int x)→ It is used to print an integer

4. void flush()→ It is used to flushes the stream.

5. void close()→ It is used to close the stream

Java PrintWriter Example

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintWriter;
public class WriteName1 {
public static void main(String[] args) {
  try {
FileOutputStream fos=newFileOutputStream("C:\\\\programs\\\\example.txt");
PrintWriter pw=new PrintWriter(fos);
pw.println(“hello");
Pw.println();
pw.println(“hi");
pw.flush();
System.out.println("written");}
  catch (FileNotFoundException e) {
  System.out.println("wrong path");
}}}
```