

Unit-Java IO

Streams:

Streams are the sequence of data that are read from the source and written to the destination.

There are three standard I/O streams:

- System.out: standard output stream for console output operations.
- System.in: standard input stream for console input operations.
- System.err: standard error stream for console error output operations.

There are two types of streams:

1. Byte Stream **(For handling input and output of byte)**
2. Character Stream **(For handling input and output of characters)**

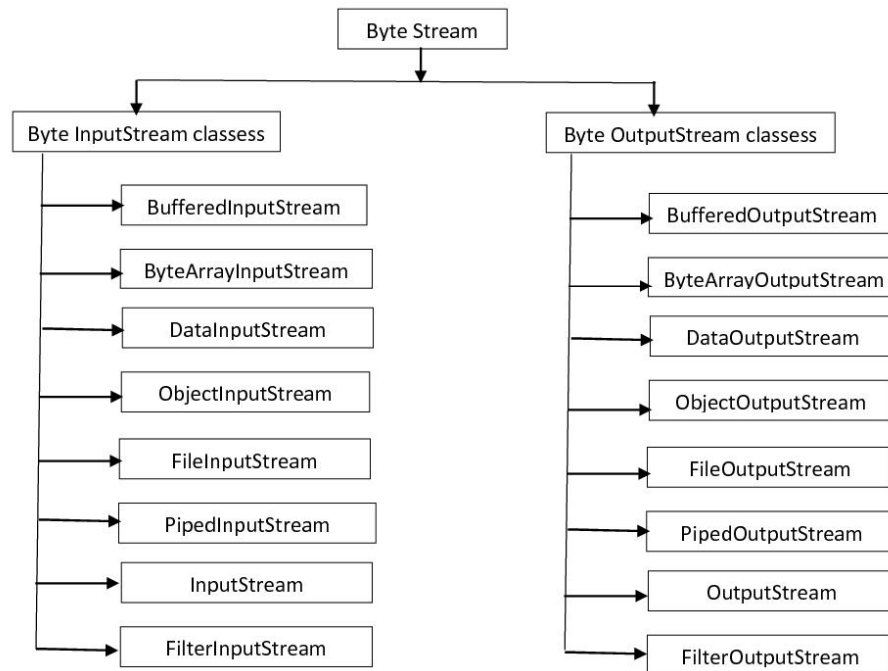
Byte Stream:

Java byte stream are used to perform input and output of 8-bits / 1 Byte at a time from binary file

Types of Byte Stream:

There are two types of Byte Stream

1. InputStream
2. OutputStream



Written by Nawaraj Prajapati (MCA From JNU)

Byte InputStream classes description:

BufferedInputStream:

Java BufferedInputStream [class](#) is used to read information from [stream](#).

Example of BufferedInputStream:

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
public class BufferedInputStreamDemo {
    public static void main(String[] args) {
        try {
            FileInputStream fileInputStream = new FileInputStream("C:\\\\bufferedinputstream.txt");
            BufferedInputStream bufferedInputStream = new BufferedInputStream(fileInputStream);

            int a;
            while ((a = bufferedInputStream.read()) != -1) {
                System.out.print((char) a);
            }
            bufferedInputStream.close();
            fileInputStream.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

ByteArrayInputStream:

A ByteArrayInputStream can retain bytes read from the input stream using its internal buffer.

Example of ByteArrayInputStream:

Output:

```
import java.io.ByteArrayInputStream;
import java.util.Scanner;
public class ByteArrayInputStreamDemo {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Please Enter Any Sentences : ");
        String str=input.nextLine();

        StringBuilder stringBuilder=new StringBuilder();
        ByteArrayInputStream byteArrayInputStream =new ByteArrayInputStream(str.getBytes());
        int chr;
        while((chr=byteArrayInputStream.read())!=-1){
            stringBuilder.append(Character.toUpperCase((char) chr));
        }
        System.out.println("Your output messages are Uppercase : "+stringBuilder.toString());
    }
}
```

Please Enter Any Sentences :
abhiraj
Your output messages are Uppercase : ABHIRAJ

Written by Nawaraj Prajapati (MCA From JNU)

DataInputStream:

It allows an application to read java's data type using an input stream which is independent of machine.

ObjectInputStream:

ObjectInputStream class deserializes primitive data and objects previously written using an ObjectOutputStream.

FileInputStream:

FileInputStream takes input bytes from a file.

PipedInputStream:

PipedInputStream. But before that it must connect to the piped output stream

InputStream:

Abstract class that describe stream input.

FilterInputStream:

FilterInputStream class contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

Methods of InputStream:

It has the following methods which have implemented by its concrete classes.

- **int available()** : It returns the number of bytes that can be read from the input stream.
- **int read()**: It reads the next byte from the input stream.
- **void close()**:It closes the input stream and also frees any resources connected with this input stream.

Byte OutputStream classes description:**BufferedOutputStream:**

The BufferedOutputStream provides the facility to contain the byte into the buffered output stream.

Example:

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
public class BufferedOutPutStreamDemo {
    public static void main(String[] args) {
        String str = "This is Buffered Output Streams";
        try {
            FileOutputStream file = new FileOutputStream("C:\\DeleteFiles\\filewriter11.txt");
            BufferedOutputStream output = new BufferedOutputStream(file);
            byte[] array = str.getBytes();
            output.write(array);
            output.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Written by Nawaraj Prajapati (MCA From JNU)

ByteArrayOutputStream:

ByteArrayOutputStream provides the facility to write data into a byte array using the output stream.

Example:

```
import java.io.ByteArrayOutputStream;
import java.io.FileOutputStream;
public class ByteArrayOutPutStreamsDemo {
    public static void main(String[] args) {
        try {
            FileOutputStream file1 = new FileOutputStream("C:\\DeleteFiles\\filewriter11.txt");
            FileOutputStream file2 = new FileOutputStream("C:\\DeleteFiles\\filewriter12.txt");
            ByteArrayOutputStream bout = new ByteArrayOutputStream();
            bout.write(65);
            bout.writeTo(file1);
            bout.writeTo(file2);
            bout.flush();
            bout.close();
        }
        catch (Exception e) {
        }
    }
}
```

DataOutputStream:

It allows an application to write java's data type (primitive) to an output stream.

ObjectOutputStream:

Java DataOutputStream class allows an application to write primitive Java data types to the output stream in a machine-independent way.

FileOutputStream:

FileOutputStream is used write data to a File or to a FileDescriptor.

PipedOutputStream:

For creating communication pipe, piped output stream should be connected to piped input stream.

OutputStream:

Abstract class that describe outputstream.

FilterOutputStream:

All the classes that filter output streams is the subclasses of the FilterOutputStream superclass.

Written by Nawaraj Prajapati (MCA From JNU)

Methods of OutputStream:

It has the following methods which have implemented by its concrete classes.

- **void write(int n)** : It writes byte(contained in an int) to the output stream.
- **void flush()**: It flushes the output steam by forcing out buffered bytes to be written out.
- **void close()**:It closes the output stream and also frees any resources connected with this output stream.

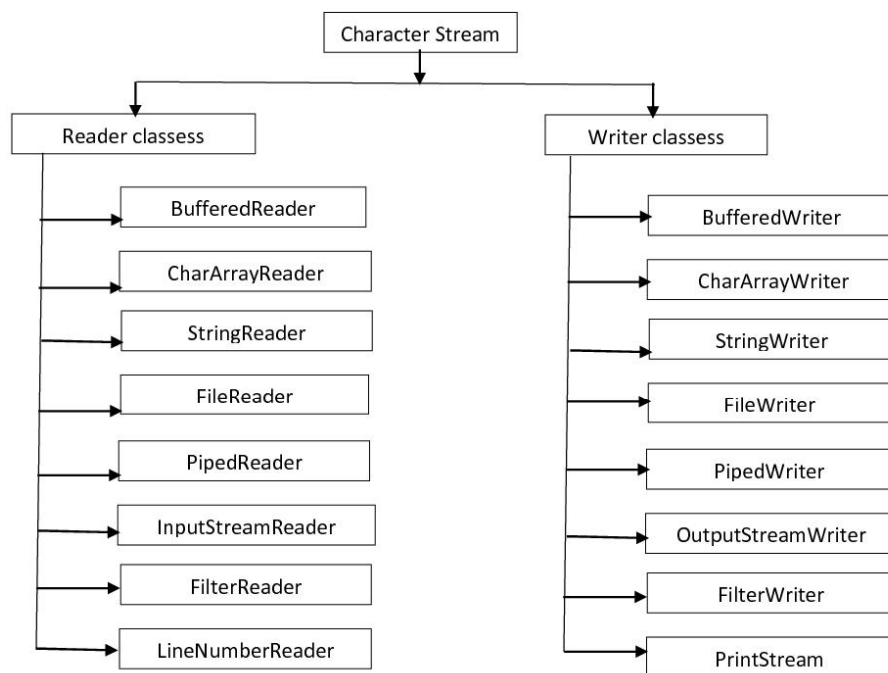
Character Stream:

Character streams are used to perform input and output of 16-bites/ 2bytes at a time from character file.

Types of Character Stream:

There are two types of **Character Stream**

1. Reader
2. Writer



Written by Nawaraj Prajapati (MCA From JNU)

Reader classes description:

These classes are subclasses of an abstract class, Reader and they are used to read characters from a source (file, memory or console).

BufferedReader:

This class is used to read characters from the buffered input character stream.

Example of BufferedReader:

```
package com.sun.iostream;
```

```
import java.io.BufferedReader;
```

```
import java.io.FileReader;
```

```
public class BufferedReaderDemo {  
    public static void main(String[] args) throws Exception {  
        FileReader fileReader= new FileReader("C:\\\\filereader.txt");  
        BufferedReader bufferedReader =new BufferedReader(fileReader);  
        String a="";  
        while((a=bufferedReader.readLine())!=null){  
            System.out.println(a);  
        }  
        bufferedReader.close();  
        fileReader.close();  
    }  
}
```

Output:

Abhiraj

CharArrayReader:

This class is used to read characters from the char array or character array.

Example of CharArrayReader:

```
import java.io.CharArrayReader;
```

```
public class CharArrayReaderDemo {  
    public static void main(String[] args) {
```

```
        try {  
            char arr[] = { 'J', 'A', 'V', 'A' };  
            CharArrayReader charArrayReader = new CharArrayReader(arr);  
            int a;  
            while ((a = charArrayReader.read()) != -1) {  
                System.out.println((char) a);  
            }  
        } catch (Exception e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

Output:

J
A
V
A

StringReader:

This class is used to read characters from a string.

Example of StringReader:

```
import java.io.IOException;
import java.io.StringReader;
public class StringReaderDemo {
    public static void main(String[] args) {
        String str = "Hello! \nHow are you?";
        StringReader stringReader = new StringReader(str);
        int a = 0;
        try {
            while ((a = stringReader.read()) != -1) {
                System.out.print((char) a);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Hello!
How are you?

FileReader:

This class is used to read characters (or contents) from a file.

Example:

```
import java.io.FileReader;
import java.io.IOException;
public class FileReaderDemo
{
    public static void main(String[] args) throws IOException {
        FileReader fr=new FileReader("C:\\DeleteFiles\\filewriter1.txt");

        int i;
        while((i=fr.read())!=-1)
            System.out.print((char)i);
        fr.close();
    }
}
```

PipedReader:

This class is used to read characters from the connected piped output stream.

InputStreamReader:

This class is used to translate (or convert) bytes to characters.

FilterReader:

This class is used to read characters from the underlying character input stream.

LineNumberReader:

This class is used to count lines.

Methods of Reader class:

It has the following methods which have implemented by its concrete classes.

int read(): This method reads a characters from the input stream.

int read(char buffer[]): It reads a chunk of characters from the input stream and store them in its byte array, buffer.

void close(): It closes the input stream and also frees any resources connected with this input stream.

void reset(): The reset() method is used to reset the input pointer to the preceding set mark.

Writer classes description:

These classes are subclasses of an abstract class, `Writer` and they used to write characters to a destination (file, memory or console).

BufferedWriter:

The java `BufferedWriter` is a class that is used to provide buffering for writing text to the character output stream. The `BufferedWriter` makes the fast performance and efficient writing of the character, string, and single array.

Example of BufferedWriter:

```
package com.sun.iostream;

import java.io.BufferedReader;
import java.io.FileReader;

public class BufferedReaderDemo {

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

        FileReader fileReader = new FileReader("C:\\filereader.txt");

        BufferedReader bufferedReader = new BufferedReader(fileReader);

        String a = "";
        while ((a = bufferedReader.readLine()) != null) {

            System.out.println(a);

        }

        bufferedReader.close();

        fileReader.close();

    }

}
```

Output:

File written Successfully

Written by Nawaraj Prajapati (MCA From JNU)

CharArrayWriter:

This class creates a buffer that is used to write data. This buffer has no fixed size and it grows automatically when data is written to the stream. There are other methods like `toCharArray()` and `toString()` to retrieve the data from the buffer.

Example of CharArrayWriter:

```
package com.sun.iostream;
import java.io.CharArrayWriter;
public class CharArrayWriterDemos {

    public static void main(String[] args) {
        CharArrayWriter charWriter = new CharArrayWriter();

        try {
            char[] ch= {'j','a','v','a'};
            charWriter.write(ch);
            String str=charWriter.toString();
            System.out.println(str);

            charWriter.write("Welcome");
            char[] chr=charWriter.toCharArray();
            for(char chrs:chr){
                System.out.println(chrs);
            }
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }

        charWriter.close();
    }
}
```

Output:

```
java
W
e
l
c
o
m
e
```

StringWriter:

This output stream class writes the characters to the string.

FileWriter:

This output stream class writes characters to the file.

PipedWriter:

This class writes the characters to the piped output stream.

OutputStreamWriter:

This output stream class translates or converts from bytes to characters.

FilterWriter:

This class is used to write characters on the underlying character output stream.

PrintStream:

This output stream class contains print() and println().

Methods of Reader class:

It has the following methods which have implemented by its concrete classes.

void write(int ch): This method writes a characters(contained in an int) to the output stream.

void write(): The write() method is used to write the data to the invoking output stream.

void write(char[] arr): This method writes a whole char array(arr) to the output stream

abstract void close(): This method closes this output stream and also frees any resources connected with this output stream.

abstract void flush():This method flushes the output steam by forcing out buffered bytes to be written out.

void close (): This method is used to close the output stream. It will produce an IOException if an attempt is made to write to the output stream after closing the stream.

Reading console input:

The Java Console class is used to get input from the console.

There are three ways to read console input. We can read input data from the console.

1. BufferedReader class
2. Scanner class
3. Console class

Example of console input:

```
package com.sun.iostream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;
public class ReadingConsoleInput {
    public static void main(String[] args) {
        BufferedReader inputBufferedReader = new BufferedReader(new InputStreamReader(System.in));
        Scanner inputScanner = new Scanner(System.in);
        int age;
        String name = "";
        try {
            System.out.print("Please Enter Your name :");
            name = inputBufferedReader.readLine();
            System.out.println("Your name is : " + name);

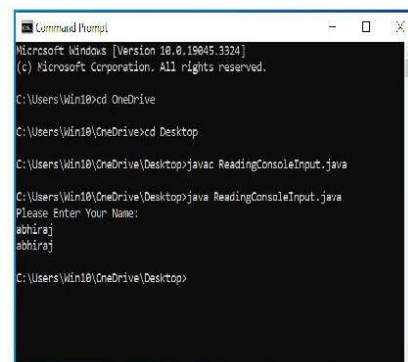
            System.out.print("Please Enter Your Age: ");
            age = inputScanner.nextInt();
            System.out.println("Your Enter Age : " + age);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
Please Enter Your name :abhiraj
Your name is : abhiraj
Please Enter Your Age: 7
Your Enter Age : 7
```

Example of console input:

```
package com.sun.iostream;
import java.io.Console;
public class ReadingConsoleInput {
    public static void main(String[] args) {
        Console console = System.console();
        String name;
        System.out.println("Please Enter Your Name: ");
        name = console.readLine();
        System.out.println(name);
    }
}
```



```
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Win10>cd OneDrive
C:\Users\Win10\OneDrive>cd Desktop
C:\Users\Win10\OneDrive\Desktop>javac ReadingConsoleInput.java
C:\Users\Win10\OneDrive\Desktop>java ReadingConsoleInput.java
Please Enter Your Name:
abhiraj
abhiraj

C:\Users\Win10\OneDrive\Desktop>
```

Written by Nawaraj Prajapati (MCA From JNU)