Java IO Stream

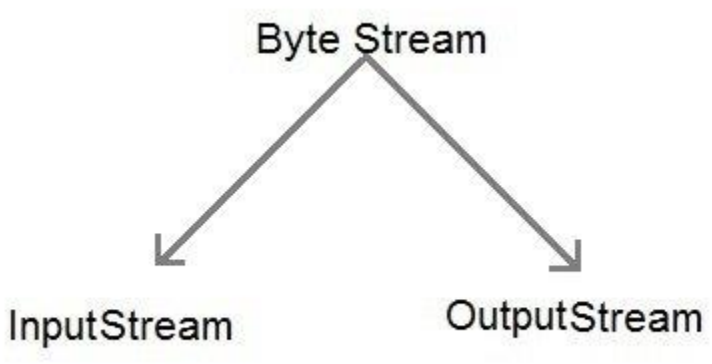
Java performs I/O through **Streams**. A Stream is linked to a physical layer by java I/O system to make input and output operation in java. In general, a stream means continuous flow of data. Streams are clean way to deal with input/output without having every part of your code understand the physical.

Java encapsulates Stream under **java.io** package. Java defines two types of streams. They are,

1. **Byte Stream :** It provides a convenient means for handling input and output of byte.
2. **Character Stream :** It provides a convenient means for handling input and output of characters. Character stream uses Unicode and therefore can be internationalized.

Java Byte Stream Classes

Byte stream is defined by using two abstract class at the top of hierarchy, they are InputStream and OutputStream.



These two abstract classes have several concrete classes that handle various devices such as disk files, network connection etc.

Some important Byte stream classes.

Stream class	Description
BufferedInputStream	Used for Buffered Input Stream.
BufferedOutputStream	Used for Buffered Output Stream.
DataInputStream	Contains method for reading java standard datatype
DataOutputStream	An output stream that contain method for writing java standard data type

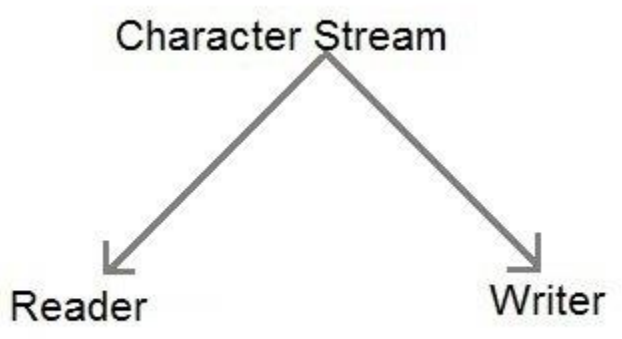
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that write to a file.
InputStream	Abstract class that describe stream input.
OutputStream	Abstract class that describe stream output.
PrintStream	Output Stream that contain `print()` and `println()` method

These classes define several key methods. Two most important are

1. `read()` : reads byte of data.
2. `write()` : Writes byte of data.

Java Character Stream Classes

Character stream is also defined by using two abstract class at the top of hierarchy, they are Reader and Writer.



These two abstract classes have several concrete classes that handle unicode character.

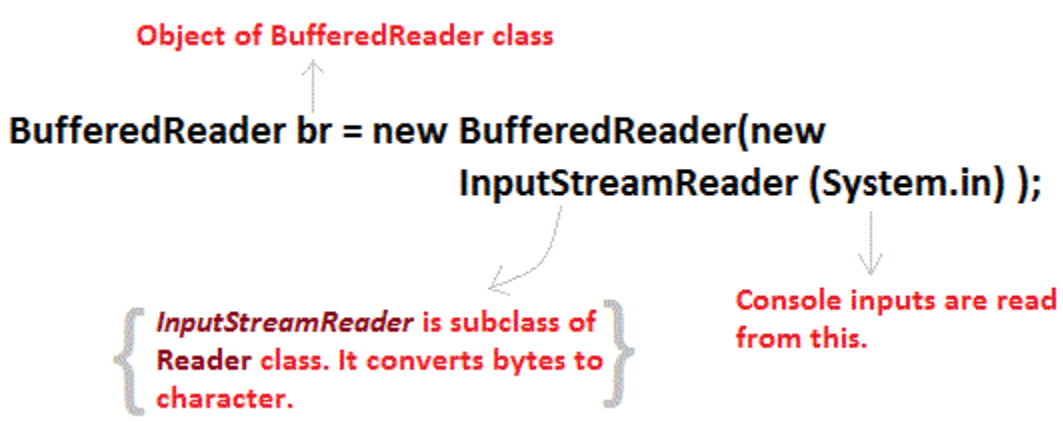
Some important Charcter stream classes

Stream class	Description
BufferedReader	Handles buffered input stream.
BufferedWriter	Handles buffered output stream.

FileReader	Input stream that reads from file.
FileWriter	Output stream that writes to file.
InputStreamReader	Input stream that translate byte to character
OutputStreamReader	Output stream that translate character to byte.
PrintWriter	Output Stream that contain `print()` and `println()` method.
Reader	Abstract class that define character stream input
Writer	Abstract class that define character stream output

Reading Console Input

We use the object of BufferedReader class to take inputs from the keyboard.



Reading Characters

`read()` method is used with BufferedReader object to read characters. As this function returns integer type value has we need to use typecasting to convert it into **char** type.

```
int read() throws IOException
```

Copy

Below is a simple example explaining character input.

```
class CharRead
```

```
{

  public static void main( String args[])

  {

    BufferedReader br = new Bufferedreader(new
InputstreamReader(System.in));

    char c = (char)br.read();   //Reading character

  }

}
```

Copy

Reading Strings in Java

To read string we have to use readLine() function with BufferedReader class's object.

```
String readLine() throws IOException
```

Copy

Program to take String input from Keyboard in Java

```
import java.io.*;

class MyInput

{

  public static void main(String[] args)

  {

    String text;

    InputStreamReader isr = new InputStreamReader(System.in);

    BufferedReader br = new BufferedReader(isr);

    text = br.readLine();    //Reading String

    System.out.println(text);

  }

}
```

Copy

Program to read from a file using BufferedReader class

```
import java. Io *;

class ReadTest
```

```
{

  public static void main(String[] args)

  {

    try

    {

      File fl = new File("d:/myfile.txt");

      BufferedReader br = new BufferedReader(new FileReader(fl)) ;

      String str;

      while ((str=br.readLine())!=null)

      {

        System.out.println(str);

      }

      br.close();

      fl.close();

    }

    catch(IOException  e) {

      e.printStackTrace();

    }

  }

}
```

Copy

Program to write to a File using FileWriter class

```
import java. Io *;

class WriteTest

{

 public static void main(String[] args)

 {

  try

  {

   File fl = new File("d:/myfile.txt");

   String str="Write this string to my file";

   FileWriter fw = new FileWriter(fl) ;
```

```
   fw.write(str);

   fw.close();

   fl.close();

  }

  catch (IOException  e)

  { e.printStackTrace(); }

 }

}
```