Stream based I/O (java.io)

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.

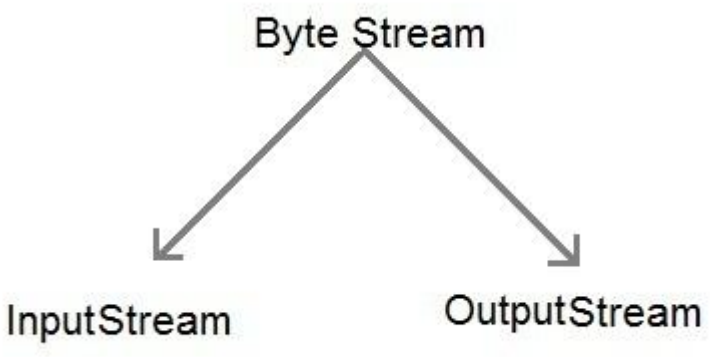
Java encapsulates Stream under **java.io** package. Java defines two types of streams. They are,

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

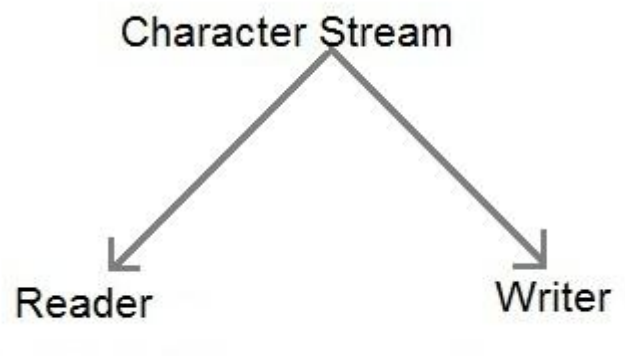
FileOutputStream	Output stream that write to a file.
InputStream	Abstract class that describe stream input.
OutputStream	Abstract class that describe stream output.
PrintStream	Output Stream that contain print() and println() method

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.

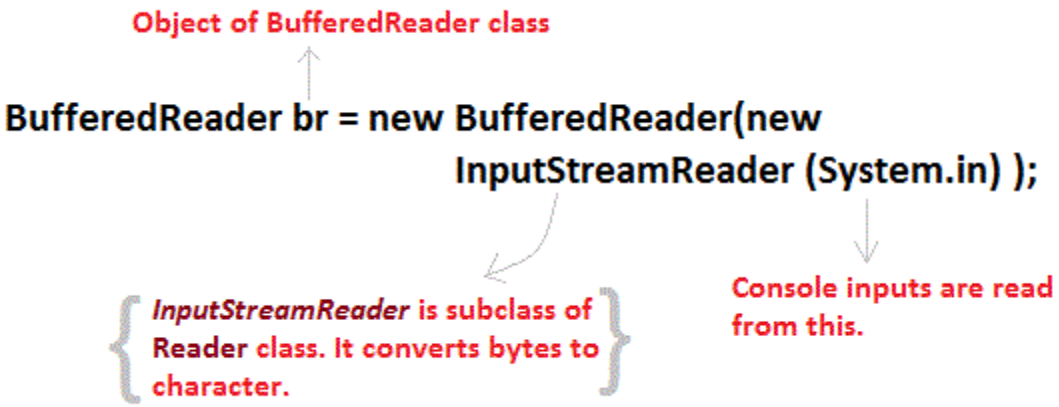
Some important Charcter stream classes

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
```

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
  }
}
```

Reading Strings in Java

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

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

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);
  }
}
```

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();
   }
 }
}
```

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(); }
```

```
 }
}
```

Java - RandomAccessFile

This class is used for reading and writing to random access file. A random access file behaves like a large array of bytes. There is a cursor implied to the array called file pointer, by moving the cursor we do the read write operations. If end-of-file is reached before the desired number of byte has been read than EOFException is thrown. It is a type of IOException.

Constructor

Constructor	Description
RandomAccessFile(File file, String mode)	Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument.
RandomAccessFile(String name, String mode)	Creates a random access file stream to read from, and optionally to write to, a file with the specified name.

Method

Modifier and Type	Method	Method
void	close()	It closes this random access file stream and releases any system resources associated with the stream.
FileChannel	getChannel()	It returns the unique FileChannel object associated with this file.
int	readInt()	It reads a signed 32-bit integer from this file.
String	readUTF()	It reads in a string from this file.
void	seek(long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

void	writeDouble(double v)	It converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the file as an eight-byte quantity, high byte first.
void	writeFloat(float v)	It converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the file as a four-byte quantity, high byte first.
void	write(int b)	It writes the specified byte to this file.
int	read()	It reads a byte of data from this file.
long	length()	It returns the length of this file.
void	seek(long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

Example

```
1.  import java.io.IOException;
2.  import java.io.RandomAccessFile;
3.
4.  public class RandomAccessFileExample {
5.     static final String FILEPATH ="myFile.TXT";
6.     public static void main(String[] args) {
7.        try {
8.           System.out.println(new String(readFromFile(FILEPATH, 0, 18)));
9.           writeToFile(FILEPATH, "I love my country and my people", 31);
10.       } catch (IOException e) {
11.          e.printStackTrace();
12.       }
13.    }
14.    private static byte[] readFromFile(String filePath, int position, int size)
15.          throws IOException {
16.       RandomAccessFile file = new RandomAccessFile(filePath, "r");
17.       file.seek(position);
18.       byte[] bytes = new byte[size];
19.       file.read(bytes);
20.       file.close();
21.       return bytes;
22.    }
23.    private static void writeToFile(String filePath, String data, int position)
```

```
24.             throws IOException {
25.         RandomAccessFile file = new RandomAccessFile(filePath, "rw");
26.         file.seek(position);
27.         file.write(data.getBytes());
28.         file.close();
29.       }
30.    }
```

The myFile.TXT contains text "This class is used for reading and writing to random access file."

after running the program it will contains

This class is used for reading I love my country and my peoplele.

Java Console Class

The Java Console class is be used to get input from console. It provides methods to read texts and passwords.

If you read password using Console class, it will not be displayed to the user.

The java.io.Console class is attached with system console internally. The Console class is introduced since 1.5.

Let's see a simple example to read text from console.

```
1.    String text=System.console().readLine();
2.    System.out.println("Text is: "+text);
```

Java Console class declaration

Let's see the declaration for Java.io.Console class:

```
1.    public final class Console extends Object implements Flushable
```

Java Console class methods

Method	Description
Reader reader()	It is used to retrieve the reader object associated with the console
String readLine()	It is used to read a single line of text from the console.
String readLine(String fmt, Object...	It provides a formatted prompt then reads the single line

args)	of text from the console.
char[] readPassword()	It is used to read password that is not being displayed on the console.
char[] readPassword(String fmt, Object... args)	It provides a formatted prompt then reads the password that is not being displayed on the console.
Console format(String fmt, Object... args)	It is used to write a formatted string to the console output stream.
Console printf(String format, Object... args)	It is used to write a string to the console output stream.
PrintWriter writer()	It is used to retrieve the PrintWriter object associated with the console.
void flush()	It is used to flushes the console.

How to get the object of Console

System class provides a static method console() that returns the singleton instance of Console class.

```
1.  public static Console console(){}
```

Let's see the code to get the instance of Console class.

```
1.  Console c=System.console();
```

Java Console Example

```
1.  import java.io.Console;
2.  class ReadStringTest{
3.  public static void main(String args[]){
4.  Console c=System.console();
5.  System.out.println("Enter your name: ");
6.  String n=c.readLine();
7.  System.out.println("Welcome "+n);
8.  }
9.  }
```

Output

```
Enter your name: Nakul Jain
Welcome Nakul Jain
```

Java Console Example to read password

```
1.  import java.io.Console;
2.  class ReadPasswordTest{
3.  public static void main(String args[]){
4.  Console c=System.console();
5.  System.out.println("Enter password: ");
6.  char[] ch=c.readPassword();
7.  String pass=String.valueOf(ch);//converting char array into string
8.  System.out.println("Password is: "+pass);
9.  }
10. }
```

Output

```
Enter password:
Password is: 123
```