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. Streams can be divided into two primary classes which can be further divided into other classes.

- **Java Byte Stream Classes**

This is used to process data byte by byte (8 bits). Though it has many classes, the FileInputStream and the FileOutputStream are the most popular ones. The FileInputStream is used to read from the source and FileOutputStream is used to write to the destination. Here is the list of various ByteStream 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

read() : reads byte of data.

write() : writes byte of data.

Example of Byte Stream Class:

```
import java.io.*;
class BStream {
	public static void main(String[] args) throws IOException
	{
		FileInputStream sourceStream = null;
		FileOutputStream targetStream = null;

		try {
			sourceStream = new FileInputStream("sourcefile.txt");
			targetStream = new FileOutputStream("targetfile.txt");
	// Reading source file and writing content to target file byte by byte
			int temp;
			while (( temp = sourceStream.read())!= -1)
				targetStream.write((byte)temp);
		}
		finally {
			if (sourceStream!=null)
				sourceStream.close();
			if (targetStream!=null)
				targetStream.close();
		}
	}
}
```

- **Java Character Stream Classes**

In Java, characters are stored using Unicode conventions. Character stream automatically allows us to read/write data character by character. Though it has many classes, the FileReader and the FileWriter are the most popular ones. FileReader and FileWriter are character streams used to read from the source and write to the destination respectively. Here is the list of various CharacterStream 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

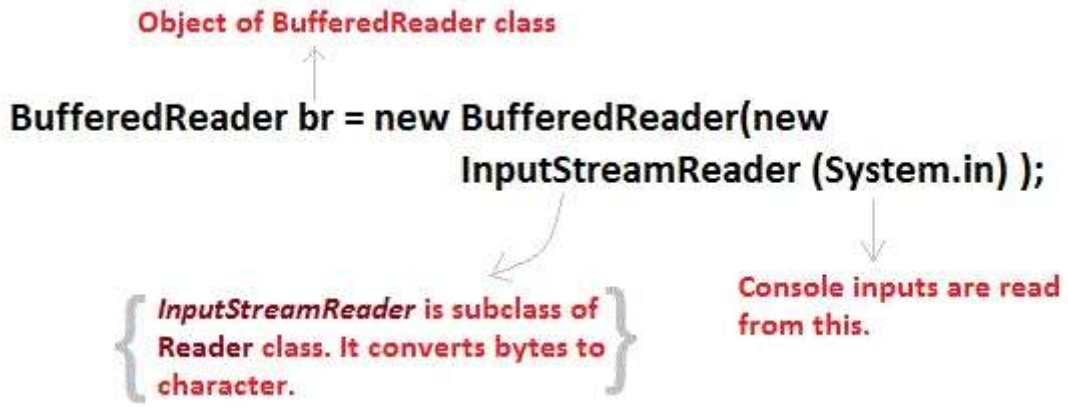
Example of Character Stream Class:

```
import java.io.*;
class CStream {
	public static void main(String[] args) throws IOException
	{
		FileReader sourceStream = null;
		try {
			sourceStream = new FileReader("sourcefile.txt");
```

```
                // Reading sourcefile character by character to console.
                int temp;
                while ((temp = sourceStream.read())!= -1)
                        System.out.println((char)temp);
        }
        Catch(IOException e) {
                System.out.println(e);
        }
    }
}
```

- **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, we need to use typecasting to convert it into char type.

int read() throws IOException

Below is a simple example explaining character input.

```
import java.io.*;
class Main
{
  public static void main( String args[])
  {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    try
    {
    char c = (char)br.read();   //Reading character
            System.out.println(c);
    }
    catch(IOException e)
    {
            System.out.println(e);
    }
  }
}
```

Reading Strings in Java

To read string we have to use readLine() method with the object of BufferedReader class.

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);
```

```
   try
   {
      text = br.readLine();    //Reading String
      System.out.println(text);
    }
    catch(IOException e)
   {
      System.out.println(e);
   }
  }
}
```

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("sourcefile.txt");
      BufferedReader br = new BufferedReader(new FileReader(fl)) ;
      String str;
      while ((str=br.readLine())!=null)
      {
        System.out.println(str);
      }
      br.close();
    }
    catch(IOException  e) {
      e.printStackTrace();
    }
  }
}
```