Course Name - Object Oriented Programming using Java

Lecture 18– Command line arguments, Basics of I/O operations

Presented By
Dr. Sudipta Sahana
Asso. Prof.
Dept. of CSE
UEM – Kolkata

Topic of Interest

- Command-Line Arguments
- IO Stream
- Byte Stream Classes
- Character Stream Classes
- Reading Characters
- Reading Strings
- Program to take String input from Keyboard in Java
- Program to read from a file using Buffered Reader class
- Java Scanner class
- Scanner Example to get input from console

Command-Line Arguments

Sometimes you will want to pass some information into a program when you run it. This is accomplished by passing command-line arguments to main( ).

```
public class CommandLine {

   public static void main(String args[]) {
      for(int i = 0; i<args.length; i++) {
         System.out.println("args[" + i + "]: " +  args[i]);
      }
   }
}
```

Try executing this program as shown here –

```
java CommandLine this is a command line 200 -100
```

Output

```
args[0]: this
args[1]: is
args[2]: a
args[3]: command
args[4]: line
args[5]: 200
args[6]: -100
```

IO Stream

·

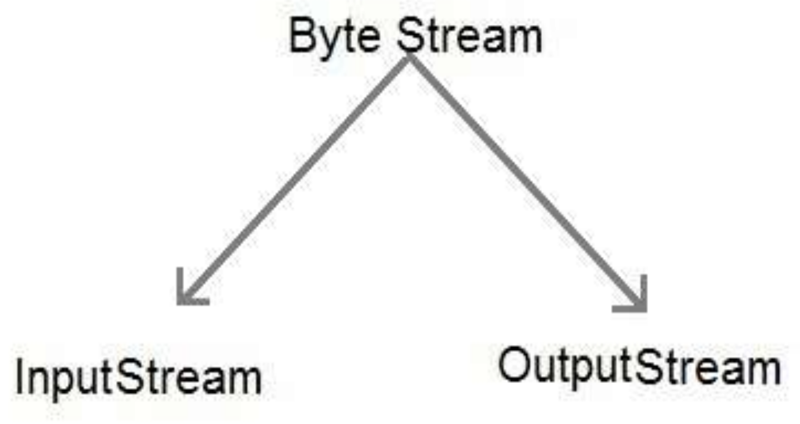
Java performs I/O through Streams. A Stream is linked to a physical layer by java I/O system to make input and output operation in java. In general, a stream means continuous flow of data. Streams are clean way to deal with input/output without having every part of your code understand the physical.

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.

Byte Stream Classes

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

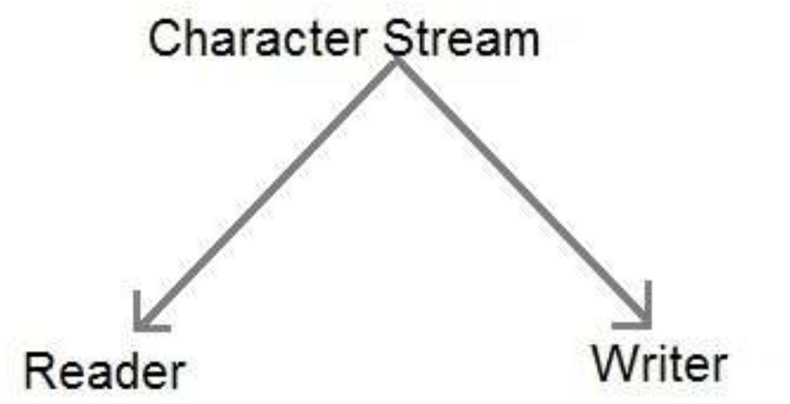


These two abstract classes have several concrete classes that handle various devices such as disk files, network connection etc.

.

Character Stream Classes

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

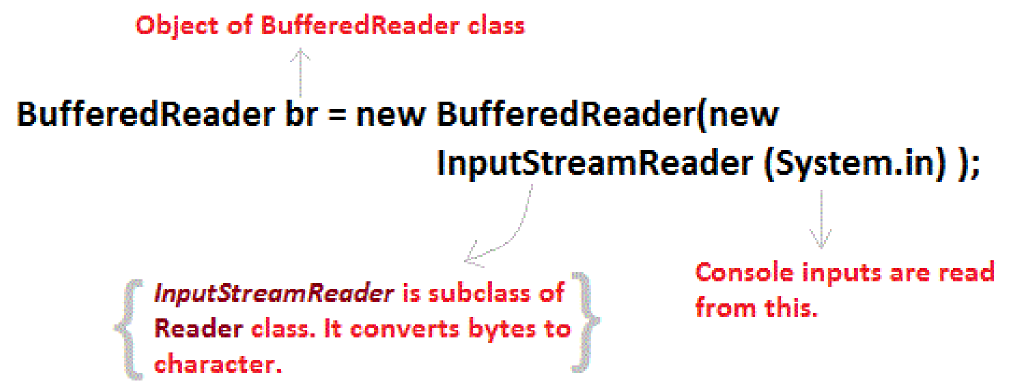


These two abstract classes have several concrete classes that handle unicode character.

Some important Character 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

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



```
BufferedReader br = new BufferedReader(new
                              InputStreamReader (System.in) );
```

Object of BufferedReader class
InputStreamReader is subclass of Reader class. It converts bytes to character.
Console inputs are read from this.

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.

Syntax

```
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

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

Syntax

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 Buffered Reader 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(); }
 }
}
```

Java Scanner class

There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.

The Java Scanner class breaks the input into tokens using a delimiter that is whitespace bydefault. It provides many methods to read and parse various primitive values.

Java Scanner class is widely used to parse text for string and primitive types using regular expression.

Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

Scanner Example to get input from console

```
import java.util.Scanner;
class ScannerTest{
 public static void main(String args[]){
   Scanner sc=new Scanner(System.in);
   System.out.println("Enter your rollno");
   int rollno=sc.nextInt();
   System.out.println("Enter your name");
   String name=sc.next();
   System.out.println("Enter your fee");
   double fee=sc.nextDouble();
   System.out.println("Rollno:"+rollno+"
name:"+name+" fee:"+fee);
   sc.close();
 }
}
```

Output

```
       Enter your rollno
       111
       Enter your name
       Ratan
       Enter your fee
       450000
       Rollno:111
name:Ratan fee:450000
```

Thank You