CS205 Object Oriented Programming in Java

Module 3 - **More features of Java (Part 3)**

Prepared by

Renetha J.B.

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

Topics

- **More features of Java :**
  - **Input / Output:**
    - I/O Basics
    - Reading Console Input
    - Writing Console Output
    - PrintWriter Class

I/O Basics

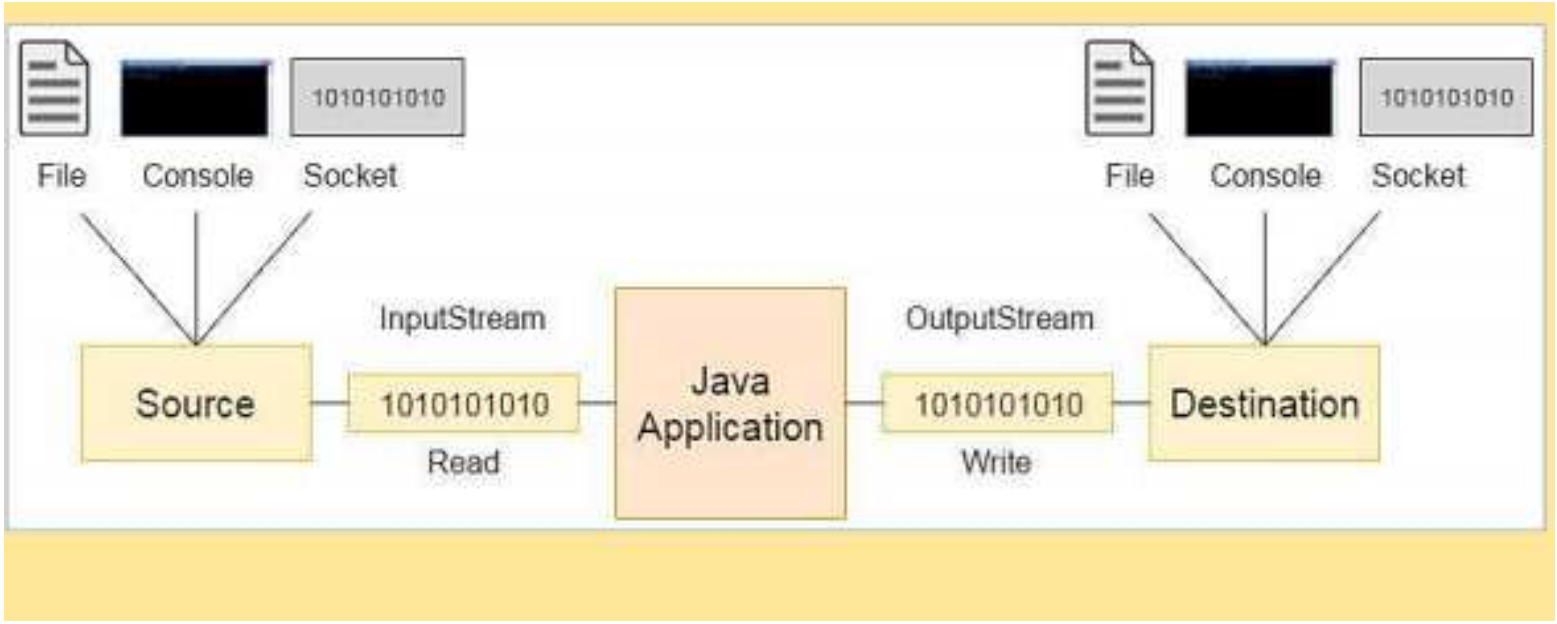
- Only **print( ) and println( )** are used frequently. All other I/O methods are not used significantly.
  - Because most real applications of Java are not text-based, console programs.
- Java's support for console I/O is limited.

Streams

- Java programs perform I/O through **streams**.
- A stream is an abstraction that either **produces or consumes** information.
- A stream is a sequence of objects that supports various methods.
- A stream is linked to a physical device by the Java I/O system.
  - **Input stream** may refer to different kinds of input:
    - from a disk file, a keyboard, or a network socket
  - **Output stream** may refer to
    - the console, a disk file, or a network connection.

Working of Javava I/O stream

- Stream is like a flow of data.



Streams

- The java.io package contains all the classes required for input and output operations.
- Java defines two types of streams: **byte** and **character**.
- ***Byte streams*** provides a means for handling input and output of bytes.
  - Byte streams are used when reading or writing binary data.
- ***Character streams*** provide a means for handling input and output of characters.
  - they use Unicode.
  - in some cases, character streams are more efficient than byte streams.

ByteStream classes

- Byte streams are defined by using two class hierarchies.

  At the top are two abstract classes:

  - **InputStream** and **OutputStream.**

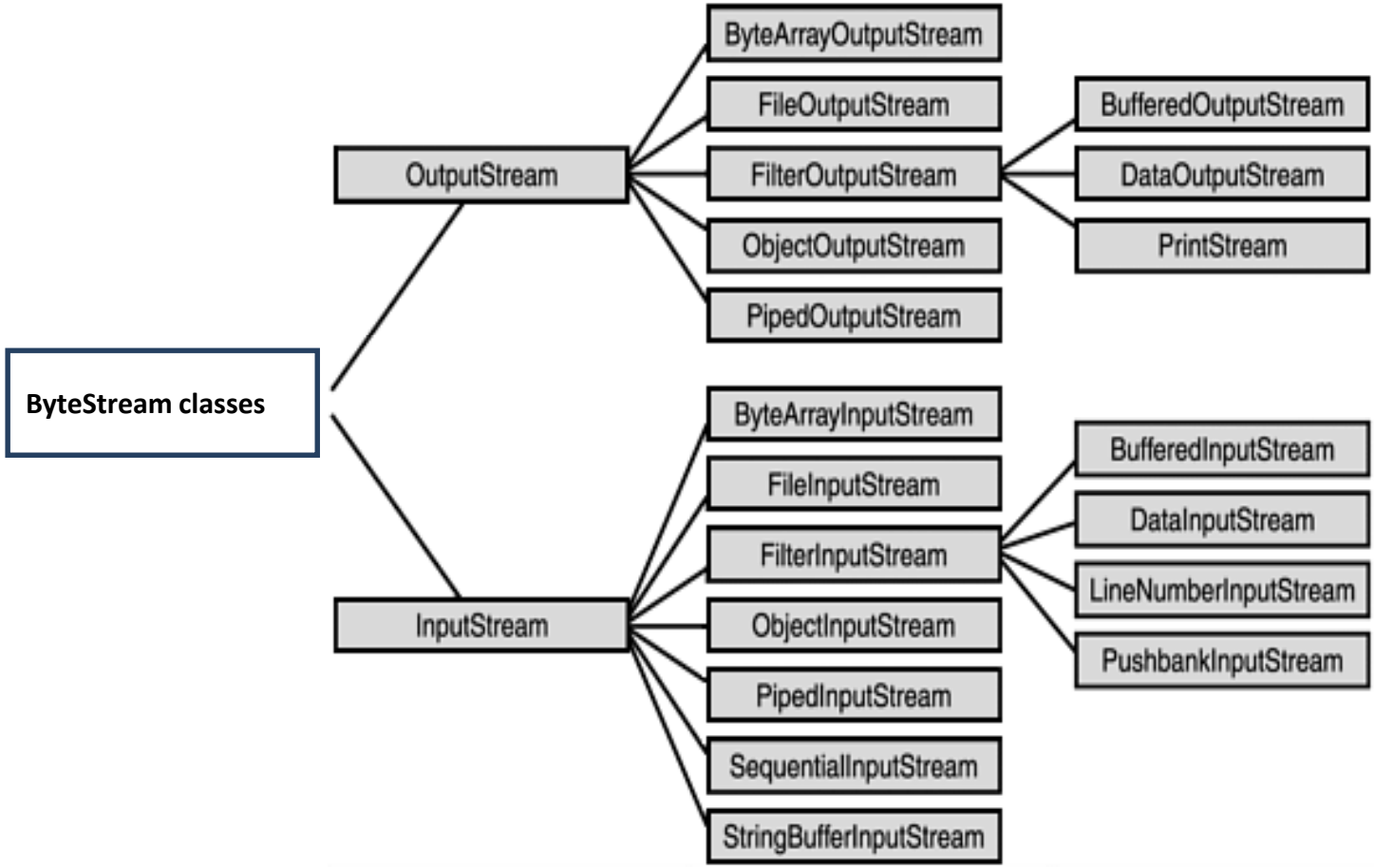
- Each of these abstract classes has several concrete subclasses

  - that handle the differences between various devices, such as disk files, network connections, and even memory buffers.

- Two of the most important are **read( ) and write( ),**
  - These methods are overridden by derived stream classes.

ByteStream classes

- **ByteStream classes**



Byte Stream I/O classes

Stream Class	Meaning
BufferedInputStream	Buffered input stream
BufferedOutputStream	Buffered output stream
ByteArrayInputStream	Input stream that reads from a byte array
ByteArrayOutputStream	Output stream that writes to a byte array
DataInputStream	An input stream that contains methods for reading the Java standard data types
DataOutputStream	An output stream that contains methods for writing the Java standard data types
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that writes to a file
FilterInputStream	Implements **InputStream**
FilterOutputStream	Implements **OutputStream**
InputStream	Abstract class that describes stream input
ObjectInputStream	Input stream for objects
ObjectOutputStream	Output stream for objects
OutputStream	Abstract class that describes stream output
PipedInputStream	Input pipe
PipedOutputStream	Output pipe
PrintStream	Output stream that contains **print( )** and **println( )**
PushbackInputStream	Input stream that supports one-byte "unget," which returns a byte to the input stream
RandomAccessFile	Supports random access file I/O
SequenceInputStream	Input stream that is a combination of two or more input streams that will be read sequentially, one after the other

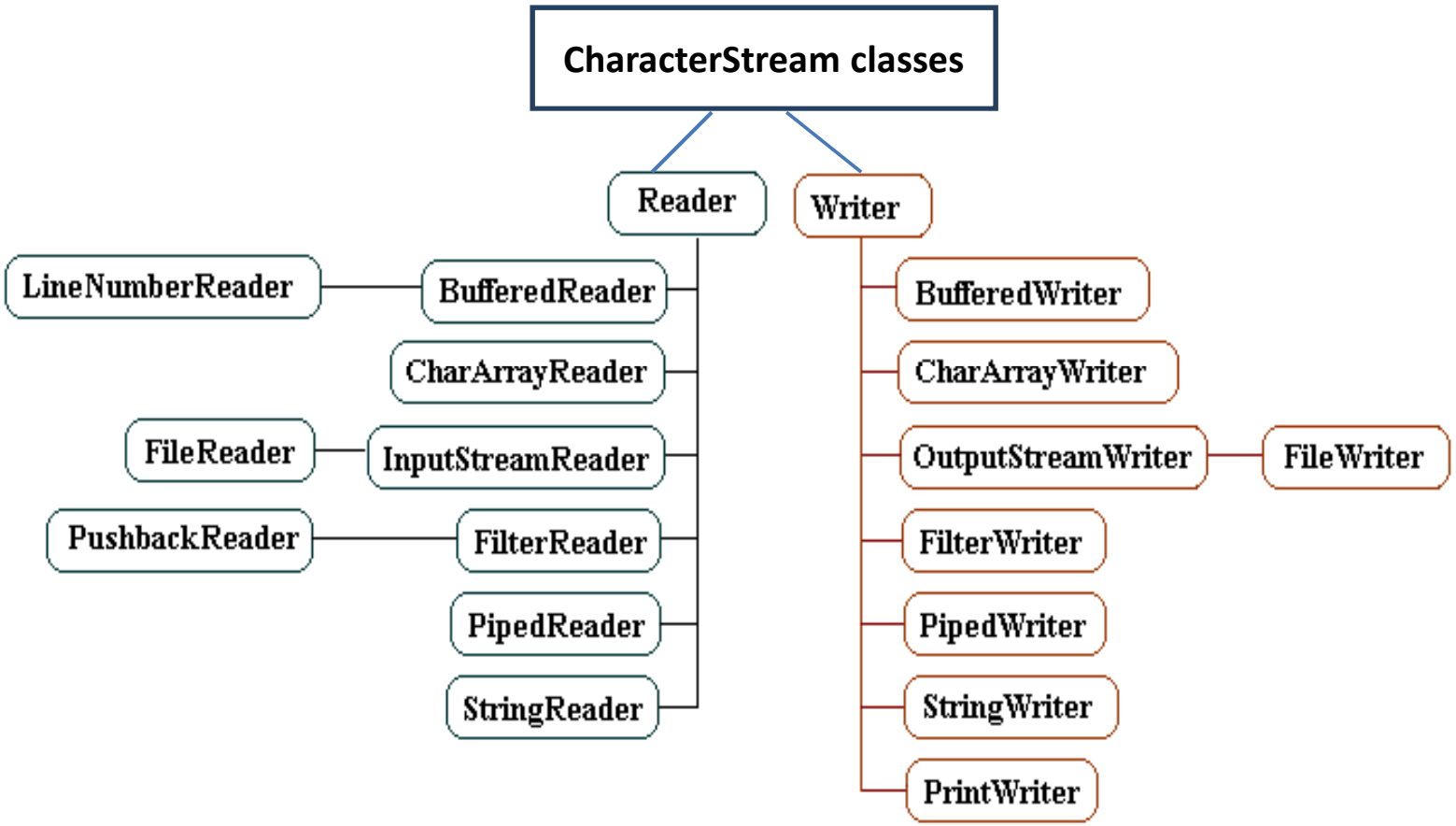
Character streams

- Character streams are defined by using two class hierarchies.

  At the top are two abstract classes,

  - **Reader** and **Writer.**

- Java has several concrete subclasses of each of these.
- Two of the most important methods are **read( )** and **write( ).**
  - These methods are overridden by derived stream classes.


CharacterStream classes
Reader
Writer
LineNumberReader
BufferedReader
CharArrayReader
FileReader
InputStreamReader
PushbackReader
FilterReader
PipedReader
StringReader
BufferedWriter
CharArrayWriter
OutputStreamWriter
FileWriter
FilterWriter
PipedWriter
StringWriter
PrintWriter

Character Stream I/O classes

Stream Class	Meaning
BufferedReader	Buffered input character stream
BufferedWriter	Buffered output character stream
CharArrayReader	Input stream that reads from a character array
CharArrayWriter	Output stream that writes to a character array
FileReader	Input stream that reads from a file
FileWriter	Output stream that writes to a file
FilterReader	Filtered reader
FilterWriter	Filtered writer
InputStreamReader	Input stream that translates bytes to characters
LineNumberReader	Input stream that counts lines
OutputStreamWriter	Output stream that translates characters to bytes
PipedReader	Input pipe
PipedWriter	Output pipe
PrintWriter	Output stream that contains **print( )** and **println( )**
PushbackReader	Input stream that allows characters to be returned to the input stream
Reader	Abstract class that describes character stream input
StringReader	Input stream that reads from a string
StringWriter	Output stream that writes to a string
Writer	Abstract class that describes character stream output

The Predefined Streams

- All Java programs automatically import the **java.lang** package. This package defines a class called System.
- **System** contains three **predefined stream variables**:
  - **in, out,** and **err**.
  - These fields are declared as **public, static,** and **final** within System.
- **System.out** refers to the standard output stream.
- **System.in** refers to standard input, which is the keyboard by default.
- **System.err** refers to the standard error stream, which is the console by default.
- System.in is an object of type InputStream; System.out and System.err are objects of type PrintStream.

Reading Console Input

- The preferred method of reading console input is to use a **character-oriented stream.**
- In Java, console input is accomplished by reading from **System.in.**
  - To obtain a character based stream that is attached to the console, wrap System.in in a BufferedReader object.

Reading Console Input(contd.)

- BufferedReader supports a **buffered input stream.**
  - Its most commonly used constructor is:

  BufferedReader(Reader inputReader)

- Here, inputReader is the stream that is linked to the instance of BufferedReader that is being created. Reader is an abstract class.

Reading Console Input(contd.)

- One of the concrete subclasses of Reader is **InputStreamReader.**
- **InputStreamReader** converts bytes to characters.
  - It reads bytes and decodes them into characters using a specified charset.
- To obtain an InputStreamReader object that is linked to System.in, the constructor that can be used is :

InputStreamReader(**InputStream** inputStream)

- Following line of code creates a **BufferedReader** that is connected to the keyboard:

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

- By wrapping the **System.in** (standard input stream) in an **InputStreamReader** which is wrapped in a **BufferedReader**, we can read input from the user in the command line.
- After this statement executes, **br** is a character-based stream that is linked to the console through System.in

Reading console-summary

To accept data from keyboard, we use **System.in. (bytestream)**

We need to **connect keyboard** to an **input stream object**.

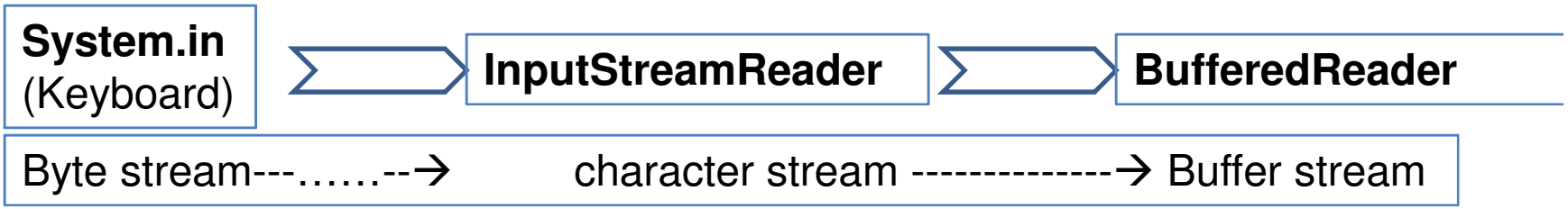
Here we can use InputStreamReader that can read data from the keyboard (**convert byte stream to character stream**)

Now our data reaches InputSreamReader..

BufferedReader class is used to read the text from a character-based input stream.

To make program run fast and to make reading efficient , ***buffering can be done using BufferedReader class.*** It can read data from stream.

Create BufferedReader object and **connect InputStreamReader object to it**




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

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

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

- **System.in** → keyboard(Byte stream)
- Convert **byte steam** to **character stream** using **InputStreamReader**.
- Then wrap that character stream in a **buffered stream BufferedReader**

Reading Characters

- To read a character from a **BufferedReader, use read( ).** The version of read( ) is

  int **read( )** throws **IOException**

- Each time that **read( )** is called, it **reads a character** from the input stream and **returns it as an integer value**. It returns –1 when the end of the stream is encountered

Reading Characters(E.g.)


```
import java.io.*;
class Readinp
{
public static void main(String a[]) throws IOException
{
char c;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter a letter");
c=(char)br.read();
System.out.println("Letter="+c);


}
}
```

OUTPUT

```
Enter a letter
a
Letter=a
```

Enter characters one by one when you type q it will stop reading


```
class BRRead {
public static void main(String args[])
throws IOException
{
char c;
BufferedReader br = new
BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter characters, 'q' to quit.");
// read characters
do {
    c = (char) br.read();
    System.out.println(c);
    } while(c != 'q');
}
}
```

OUTPUT

```
Enter characters, 'q' to quit.
123abcq
1
2
3
a
b
c
q
```

No input is actually passed to the program until you
press ENTER.

Reading Strings

- To read a string from the keyboard, use the version of **readLine( )** that is a member of the **BufferedReader class.**
- Its general form is

String readLine( ) throws IOException

This returns a **String** object.

Reading Strings(E.g.)


```
import java.io.*;
class Readinp
{
public static void main(String a[]) throws IOException
{
char c;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter a line of text");
c=(char)br.read();
s=br.readLine();
System.out.println("Line is "+s);
}
}
```

OUTPUT

```
Enter a line of text
How are you
Line is How are you
```

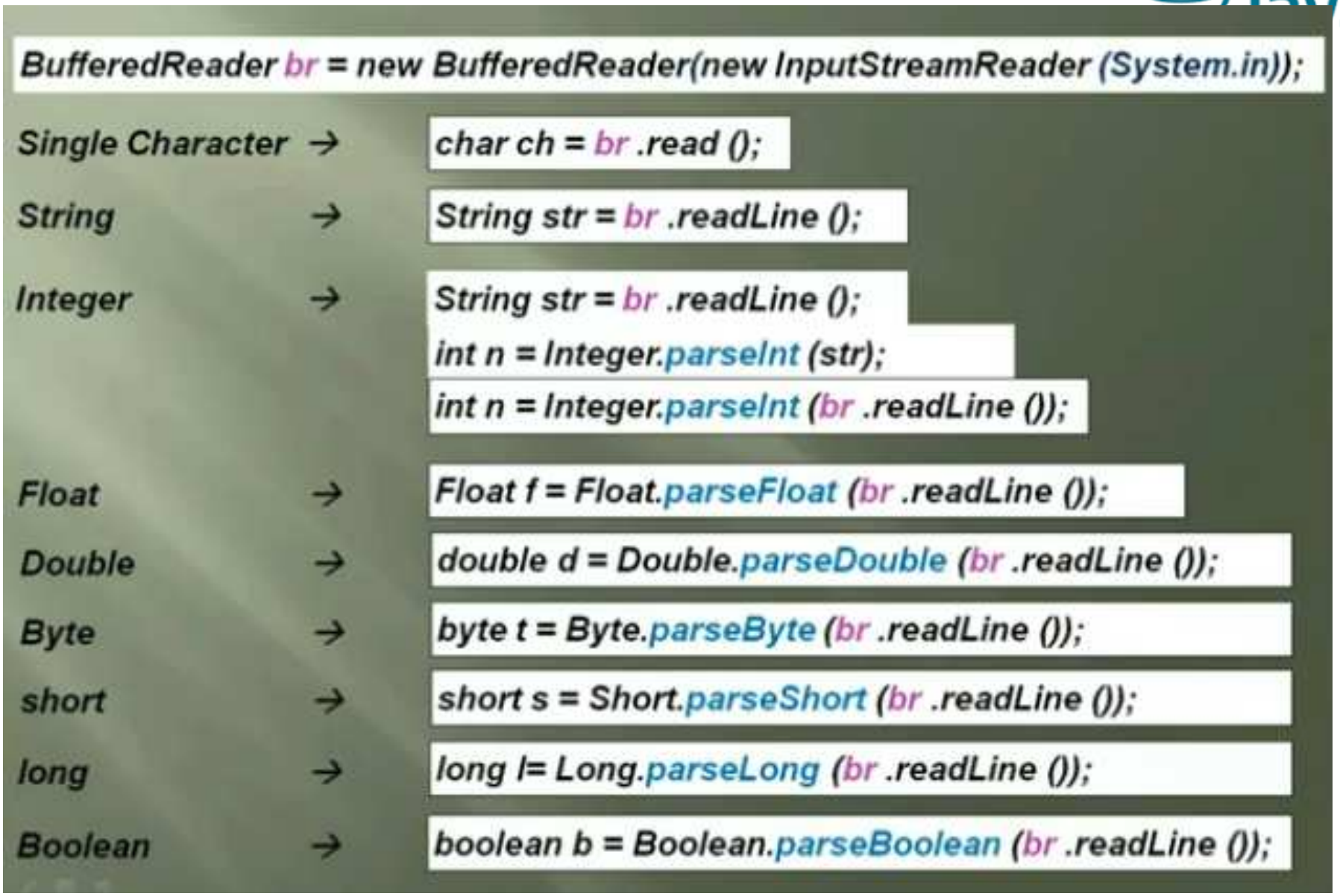
Enter lines of text one by one when you type stop it will stop reading


```
import java.io.*;
class Readlinetillstop
{
public static void main(String a[]) throws IOException
{
String s[] = new String[100];

BufferedReader br=new BufferedReader(new  InputStreamReader(System.in));
System.out.println("Enter a line of text");
System.out.println("Enter 'stop' to quit.");
for(int i=0; i<100; i++)
{
s[i] = br.readLine();
if(s[i].equals("stop")) break;
}
System.out.println("Lines are ");
for(int i=0; i<100; i++)
{

if(s[i].equals("stop")) break;
System.out.println(s[i]);
}  }}
```

OUTPUT

```
Enter a line of text
Enter 'stop' to quit.
what
how are you
ok
Stop
Lines are
what
how are you
ok
```

Read console inputs



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

Single Character →	`char ch = br .read ();`
String →	`String str = br .readLine ();`

Integer →	`String str = br .readLine ();`
	`int n = Integer.parseInt (str);`
	`int n = Integer.parseInt (br .readLine ());`

Float →	`Float f = Float.parseFloat (br .readLine ());`
Double →	`double d = Double.parseDouble (br .readLine ());`
Byte →	`byte t = Byte.parseByte (br .readLine ());`
short →	`short s = Short.parseShort (br .readLine ());`
long →	`long l= Long.parseLong (br .readLine ());`
Boolean →	`boolean b = Boolean.parseBoolean (br .readLine ());`

Writing Console Output

- Console output is usually done through **print( ) and println().**
- These methods are defined by the class **PrintStream.**
  - It is the type of object **referenced by System.out.**
  - **System.out** is a byte stream,
- **PrintStream** is an output stream derived from **OutputStream**,
  - **PrintStream** also implements the low-level method write( ).
  - write( ) can be used to write to the console.

Writing Console Output

- The simplest form of write( ) defined by PrintStream is

  void write(int *byteval)*

  - This method writes the byte specified by *byteval* to the stream
  - *byteval is declared* as an integer, only the low-order eight bits are written.

```
// Demonstrate System.out.write(). Write letter ‘A’ to console.
class WriteDemo {
public static void main(String args[]) {
int b;
b = 'A';
System.out.write(b);
System.out.write('\n');
}
}
```

OUTPUT
A

PrintWriter class

- For real-world programs, the recommended method of writing to the console using Java is through a **PrintWriter stream.**
- **PrintWriter** is one of the **character-based classes.**
- **PrintWriter** defines several constructors.

  PrintWriter(**OutputStream** *outputStream,* ***boolean*** *flushOnNewline)*

- Here, *outputStream* is an object of type OutputStream, and *flushOnNewline* controls whether Java flushes the output stream every time a println( ) method is called.
  - If flushOnNewline is true, flushing automatically takes place. If false, flushing is not automatic.

- **PrintWriter** supports the **prin)** and **println( ) methods**
- If an argument is not a simple type, the **PrintWriter methods** call the object's **toString( )** method and then print the result.
- To **write to the console** by using a **PrintWriter,** specify **System.out** for the output stream and flush the stream after each new line.

PrintWriter pw = new **PrintWriter**(System.out, **true**);

```
// Demonstrate PrintWriter
import java.io.*;
public class PrintWriterDemo {
public static void main(String args[])
    {
    PrintWriter pw = new PrintWriter(System.out, true);
    pw.println("This is a string");
    int i = -7;
    pw.println(i);
    double d = 4.5e-7;
    pw.println(d);
    }
}
```

OUTPUT

```
This is a string
-7
4.5E-7
```

System.out

- System.out is a byte stream.
- **System.out** refers to the standard output stream(monitor).
- **System:** It is a final class defined in the java.lang package.
- **out:** This is an instance of PrintStream type, which is a public and static member field of the System class.

PrintWritter

- **PrintWriter** should be used to write a stream of characters
- PrintWriter is a subclass of **Writer** (character stream class)
- It is used in real world programs to make it easier to internationalize the program

Reference

- Herbert Schildt, Java: **The Complete Reference, 8/e, Tata McGraw Hill, 2011**.