Console based IO

using System.in and System.out objects.

Streams

Streams in Programming

- A stream is an abstraction that either produces or consumes information.
- A stream is linked to a physical device.
- an input stream can abstract many different kinds of input: from a disk file, a keyboard, or a network socket
- an output stream may refer to the console, a disk file, or a network connection.

Two types of streams in JAVA

- **Byte Stream** (provide a convenient means for handling input and output of bytes) -binary data
- **Character Stream** (provide a convenient means for handling input and output of characters) -unicode

Note*: At the lowest level all I/O is byte-oriented.

Byte Stream Classes

- There are two abstract classes **InputStream** and **OutputStream** for Byte Streams
- The abstract classes InputStream and OutputStream define several key methods that the other stream classes implement.
- Two of the most important methods are **read( )** and **write( )**, which, respectively, read and write bytes of data
- Useful for reading and writing raw binary data, such as images, audio, video files, and non-textual data, and other files.

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
SequenceInputStream	Input stream that is a combination of two or more input streams that will be read sequentially, one after the other

Several Byte Stream Classes in **java.io** package

Character Stream

- Character streams are defined by using two class hierarchies. At the top are two abstract classes: **Reader** and **Writer**.
- These abstract classes handle Unicode character streams and can be internationalized easily.
- Two of the most important methods available to Reader and Writer are **read( )** and **write( )**, which read and write characters of data, respectively.
- Better to use with text files, configuration files, and documents. They handle character encoding and decoding automatically.

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

Character Stream Classes in **java.io** package

Predefined Streams in JAVA

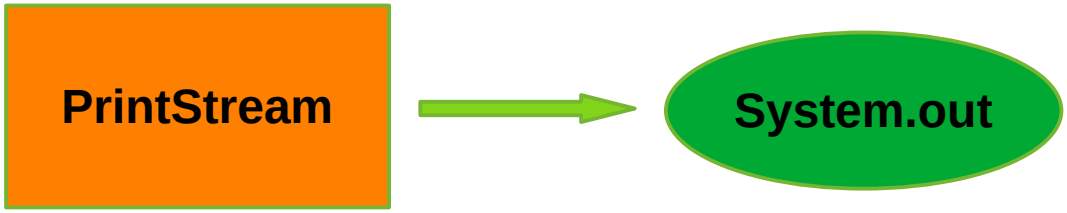
- Java programs automatically import the java.lang package which defines a class **System**
- **System** contains 3 predefined stream variables:

  i)**in**, ii) **out**, iii) **err**

  - **These fields are declared as public, static, and final within System** (can be used by any other part of your program and without reference to a specific System object.)

System.out

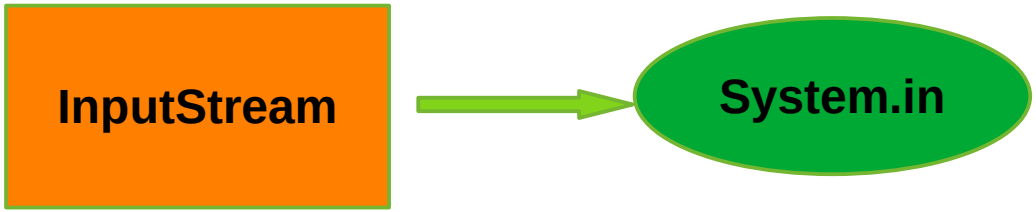
- System.out refers to standard output stream.
- It is an object of type **PrintStream (**Byte Stream**)**



- **Example:** System.out.println(“hello world”);
- System.err is also an object of PrintStream, both write output to the console

System.in

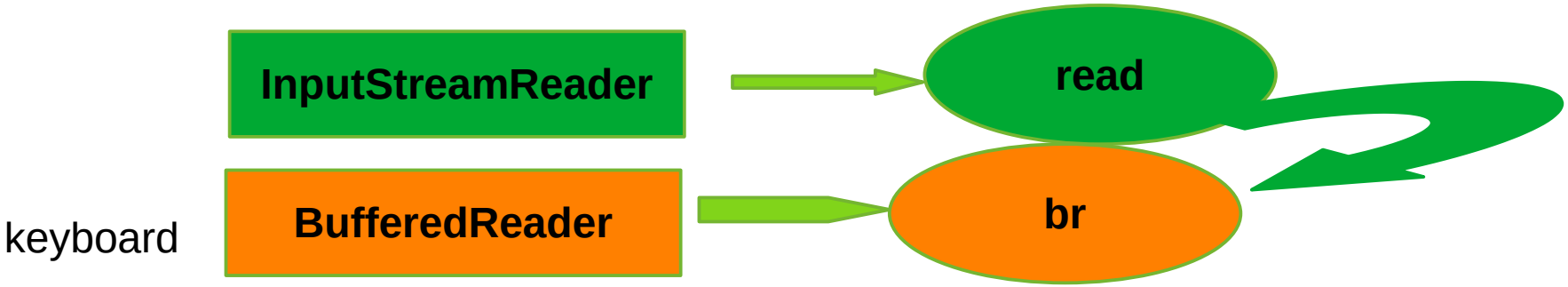
- System.in refers to standard input stream.
- It is an object of type **InputStream (**Byte Stream**)**



- **Example:** new Scanner(System.in)
- System.in is used to read input from the console

Reading Console Input (Character Stream)

- Use the concrete subclass InputStreamReader of the abstract class Reader



- InputStreamReader read = new InputStreamReader(System.in);
- BufferedReader br = new BufferedReader(read);
- Here br is character based stream i.e linked to console (System.in)

Writing Console Output

- Console output is most easily accomplished with print( ) and println( ) via System.out objects.
- Example: System.out.println(“hello World”);
- For Byte Stream we can also use write() method which is available to the PrintStream (derivative of OutputStream)
- Example: System.out.write();
- For a character based stream we can make use of **PrintWriter** stream
- Syntax:
  PrintWriter(OutputStream outputStream, boolean flushingOn)

PrintWriter

- Here, outputStream is an object of type OutputStream, and flushingOn controls whether Java flushes the output stream every time a println( ) method is called.
- PrintWriter also supports the print( ) and println( ) methods.
- PrintWriter makes your real-world applications easier to internationalize.
- But no advantage is gained by using a PrintWriter in the simple text based programs, so we can use System.out.println()