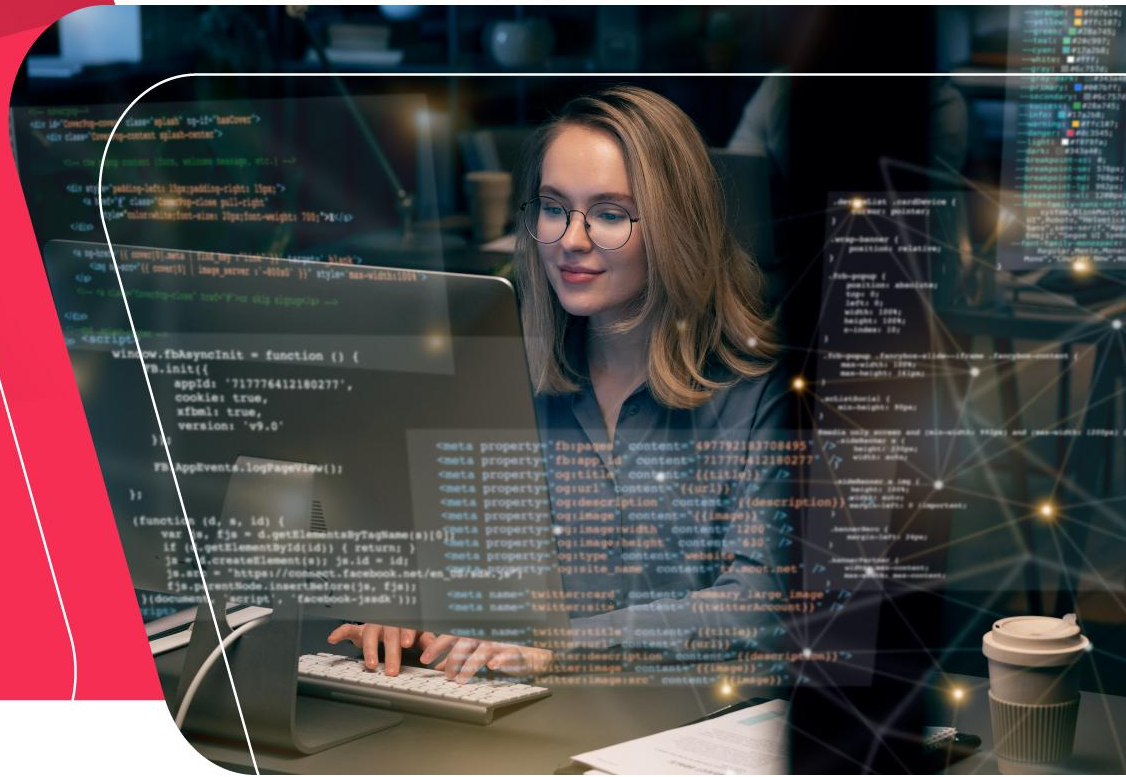
TIA-Academy

Java Bootcamp

Day 14



Technologies will be Use

- JDK 8/**11**/15
- JRE 8/**11**/15
- Writing Code using Notepad++ (For Windows) or Vim (For Linux and Mac)
- Compiling with Command Prompt (For Windows) or Terminal (For Linux and Mac)
- ***Forbidden of using IDE Based Development Tools as of Intellij IDEA or Eclipse***

TIA-Academy

I/O

Java I/O

- **Java I/O** (Input and Output) is used *to process the input* and *produce the output*.
- Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.
- We can perform **file handling in java** by Java I/O API.

Stream

- A stream is a sequence of data.In Java a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.
- In java, 3 streams are created for us automatically. All these streams are attached with console.
  - **`System.out:`** standard output stream
  - **`System.in:`** standard input stream
  - **`System.err:`** standard error stream

Stream

- Let's see the code to print **output and error** message to the console.

```
System.out.println("simple message");

System.err.println("error message");
```

- Let's see the code to get **input** from console.

```
int i=System.in.read();//returns ASCII code of 1st character

System.out.println((char)i);//will print the character
```

OutputStream vs InputStream

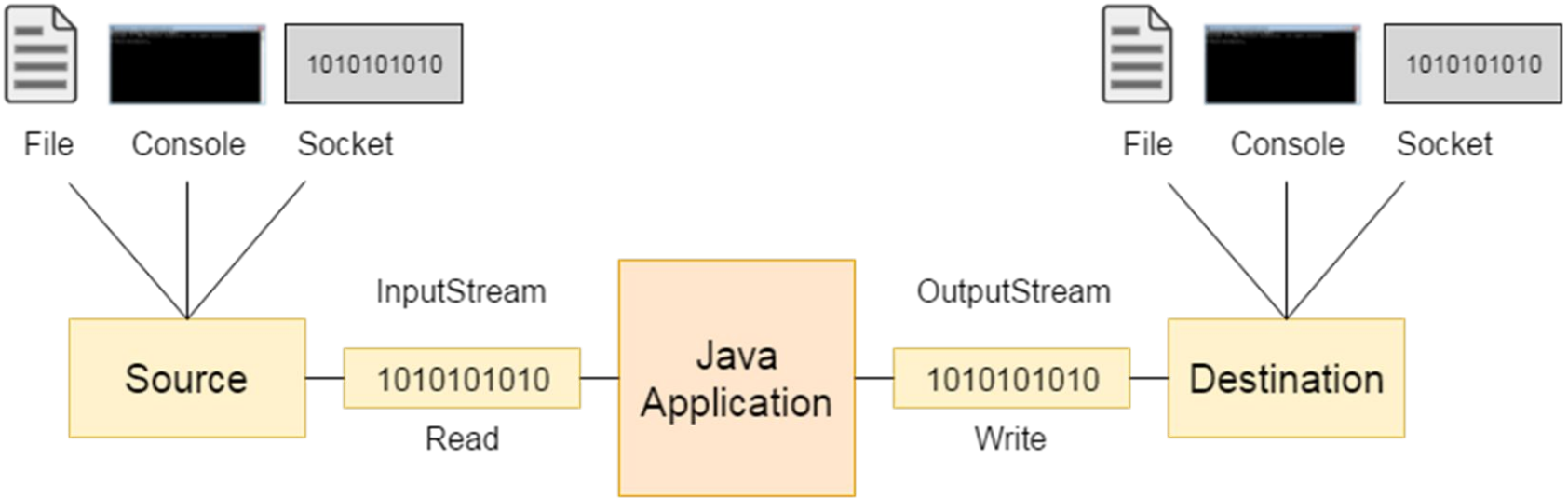
OutputStream

- Java application uses an output stream to **write data to a destination**, it may be a **file**, an array, peripheral device or socket.

InputStream

- Java application uses an input stream to **read data from a source**, it may be a **file**, an array, peripheral device or socket.

OutputStream vs InputStream

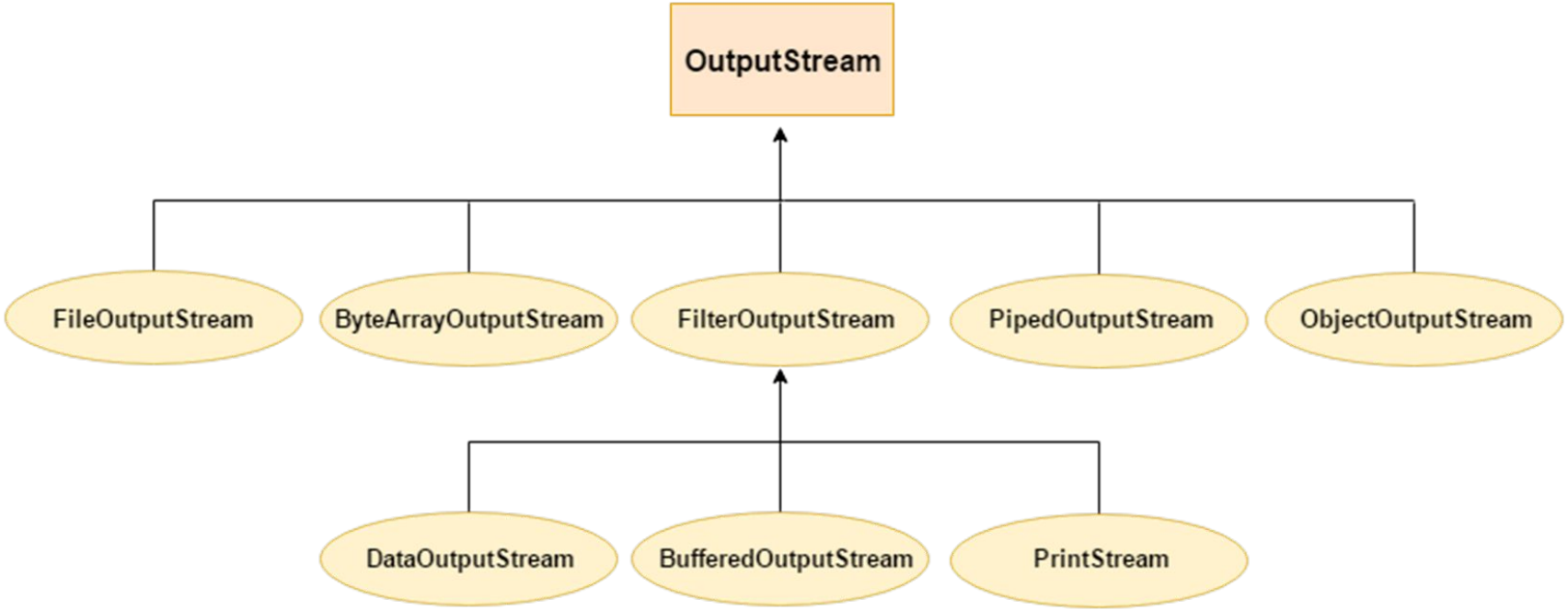


OutputStream class

- OutputStream class is an abstract class. It is the super class of all classes representing an output stream of bytes.
- An output stream accepts output bytes and sends them to some sink.

Method	Description
1) public void write(int)throws IOException	is used to write a byte to the current output stream.
2) public void write(byte[])throws IOException	is used to write an array of byte to the current output stream.
3) public void flush()throws IOException	flushes the current output stream.
4) public void close()throws IOException	is used to close the current output stream.

OutputStream Hierarchy



FileOutputStream Class

- Java **FileOutputStream** is an output stream used for **writing data to a file**.
- If you have to write primitive values into a file, use FileOutputStream class.
- You can **write byte-oriented** as well as **character-oriented** data through FileOutputStream class.
- But, for character-oriented data, it is preferred to use FileWriter than FileOutStream.

```
public class FileOutputStream extends OutputStream
```

FileOutputStream methods

Method	Description
protected void finalize()	It is sued to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write **ary.length** bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write **len** bytes from the byte array starting at offset **off** to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
FileChannel getChannel()	It is used to return the file channel object associated with the file output stream.
FileDescriptor getFD()	It is used to return the file descriptor associated with the stream.
void close()	It is used to closes the file output stream.

FileOutputStream

- Example 1: write byte
- Output:

```
Success...
```

- The content of a text file testout.txt is set with the data A. testout.txt

```
A
```

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
    public static void main(String args[]){
            try{
              FileOutputStream foutnew FileOutputStream("C:\\tmp\\test.txt");
              fout.write(65);
              fout.close();
              System.out.println("success...");
             }catch(Exception e){System.out.println(e);}
      }
}
```

FileOutputStream

- Example 2: write string
- Output:

```
Success...
```

- The content of a text file **testout.txt** is set with the data **Welcome to G2**. testout.txt

```
Welcome to G2.
```

```
import java.io.FileOutputStream;

public class FileOutputStreamExample2 {
    public static void main(String args[]){
           try{
             FileOutputStream fout=new FileOutputStream("C:\\tmp\\test2.txt");
             //String s=args[0];
             String s="ABC";
             byte b[]=s.getBytes();//converting string into byte array
             fout.write(b);
             fout.close();
             System.out.println("success...");
            }catch(Exception e){System.out.println(e);}
      }
}
```

FileInputStream Class

- Java FileInputStream class obtains input bytes from a file.
- It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc.
- You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

```
public class FileInputStream extends InputStream
```

FileInputStream methods

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to **b.length** bytes of data from the input stream.
int read(byte[] b, int off, int len)	It is used to read up to **len** bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
FileChannel getChannel()	It is used to return the unique FileChannel object associated with the file input stream.
FileDescriptor getFD()	It is used to return the FileDescriptor object.
protected void finalize()	It is used to ensure that the close method is call when there is no more reference to the file input stream.
void close()	It is used to closes the stream.

FileInputStream

- Before running the code, a text file named as "testout.txt" is required to be created. In this file, we are having following content.

```
“Welcome to G2”
```

- After executing the above program, you will get a single character from the file which is 87 (in byte form). To see the text, you need to convert it into character.

FileInputStream

- Example 1
- Output:

```
W
```

```
import java.io.FileInputStream;
public class DataStreamExample {
     public static void main(String args[]){
          try{
            FileInputStream fin=new FileInputStream("D:\\testout.txt");
            int i=fin.read();
            System.out.print((char)i);

            fin.close();
          }catch(Exception e){System.out.println(e);}
     }
}
```

FileInputStream

- Example 2: read all characters.
- Output:

```
Welcome to G2
```

```
import java.io.FileInputStream;
public class DataStreamExample {
     public static void main(String args[]){
          try{
            FileInputStream fin=new FileInputStream("D:\\testout.txt")
;
            int i=0;
            while((i=fin.read())!=-1){
             System.out.print((char)i);
            }
            fin.close();
          }catch(Exception e){System.out.println(e);}
    }
}
```

BufferedOutputStream Class

- Java BufferedOutputStream class is used for buffering an output stream.
- It internally uses buffer to store data.
- It adds more efficiency than to write data directly into a stream and makes the performance fast.
- For adding the buffer in an OutputStream, use the BufferedOutputStream class. Let's see the syntax for adding the buffer in an OutputStream:

```
OutputStream os =
    new BufferedOutputStream(new FileOutputStream("D:\\IO Package\\testout.txt"));
```

```
public class BufferedOutputStream extends FilterOutputStream
```

BufferedOutputStream constructors

Constructor	Description
BufferedOutputStream(OutputStream os)	It creates the new buffered output stream which is used for writing the data to the specified output stream.
BufferedOutputStream(OutputStream os, int size)	It creates the new buffered output stream which is used for writing the data to the specified output stream with a specified buffer size.

BufferedOutputStream methods

Method	Description
void write(int b)	It writes the specified byte to the buffered output stream.
void write(byte[] b, int off, int len)	It write the bytes from the specified byte-input stream into a specified byte array, starting with the given offset
void flush()	It flushes the buffered output stream.

Example of BufferedOutputStream

- In this example, we are writing the textual information in the BufferedOutputStream object which is connected to the FileOutputStream object.
- The flush() flushes the data of one stream and send it into another.
- It is required if you have connected the one stream with another.

Example of BufferedOutputStream

- Output:

```
Success
```

- testout.txt:

```
Welcome to G2
```

```
import java.io.*;
public class BufferedOutputStreamExample{
public static void main(String args[])throws Exception{
     FileOutputStream fout=new FileOutputStream("D:\\testout.txt");

     BufferedOutputStream bout=new BufferedOutputStream(fout);
     String s="Welcome to G2.";
     byte b[]=s.getBytes();
     bout.write(b);
     bout.flush();
     bout.close();
     fout.close();
     System.out.println("success");
}
}
```

BufferedInputStream Class

- BufferedInputStream class is used to read information from stream.
- It internally uses buffer mechanism to make the performance fast.
- When the bytes from the stream are skipped or read, the internal buffer automatically refilled from the contained input stream, many bytes at a time.
- When a BufferedInputStream is created, an internal buffer array is created

```
public class BufferedInputStream extends FilterInputStream
```

BufferedInputStream constructors

Constructor	Description
BufferedInputStream(InputStream IS)	It creates the BufferedInputStream and saves it argument, the input stream IS, for later use.
BufferedInputStream(InputStream IS, int size)	It creates the BufferedInputStream with a specified buffer size and saves it argument, the input stream IS, for later use.

BufferedInputStream methods

Method	Description
int available()	It returns an estimate number of bytes that can be read from the input stream without blocking by the next invocation method for the input stream.
int read()	It read the next byte of data from the input stream.
int read(byte[] b, int off, int ln)	It read the bytes from the specified byte-input stream into a specified byte array, starting with the given offset.
void close()	It closes the input stream and releases any of the system resources associated with the stream.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readlimit)	It sees the general contract of the mark method for the input stream.
long skip(long x)	It skips over and discards x bytes of data from the input stream.
boolean markSupported()	It tests for the input stream to support the mark and reset methods.

Example of BufferedInputStream

- Here, we are assuming that you have following data in **"testout.txt"** file:

  G2

- Output:

  G2

```
import java.io.*;
public class BufferedInputStreamExample{
 public static void main(String args[]){
  try{
    FileInputStream fin=new FileInputStream("D:\\testout.txt");
    BufferedInputStream bin=new BufferedInputStream(fin);
    int i;
    while((i=bin.read())!=-1){
     System.out.print((char)i);
    }
    bin.close();
    fin.close();
  }catch(Exception e){System.out.println(e);}
 }
}
```

TIA-Academy

ASSIGNMENT 01

FileWriter Class

- Java FileWriter class is used to write character-oriented data to a file. It is character-oriented class which is used for file handling in java.
- Unlike FileOutputStream class, you don't need to convert string into byte array because it provides method to write string directly.

```
public class FileWriter extends OutputStreamWriter
```

FileWriter constructors

Constructor	Description
FileWriter(String file)	Creates a new file. It gets file name in string.
FileWriter(File file)	Creates a new file. It gets file name in File object.

FileWriter methods

Method	Description
void write(String text)	It is used to write the string into FileWriter.
void write(char c)	It is used to write the char into FileWriter.
void write(char[] c)	It is used to write char array into FileWriter.
void flush()	It is used to flushes the data of FileWriter.
void close()	It is used to close the FileWriter.

Example of FileWriter

- In this example, we are writing the data in the file testout.txt using Java FileWriter class.
- Output:

```
Success...
```

- testout.txt:

```
Welcome to G2
```

```
import java.io.FileWriter;
public class FileWriterExample {
    public static void main(String args[]){
         try{
           FileWriter fw=new FileWriter("D:\\testout.txt");

           fw.write("Welcome to G2");
           fw.close();
          }catch(Exception e){System.out.println(e);}
          System.out.println("Success...");
     }
}
```

FileReader Class

- Java FileReader class is used to read data from the file. It returns data in byte format like FileInputStream class.
- It is character-oriented class which is used for file handling in java.

```
public class FileReader extends InputStreamReader
```

FileReader constructors

Constructor	Description
FileReader(String file)	It gets filename in string. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.
FileReader(File file)	It gets filename in file instance. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.

FileReader methods

Method	Description
int read()	It is used to return a character in ASCII form. It returns -1 at the end of file.
void close()	It is used to close the FileReader class.

Example of FileReader

- In this example, we are reading the data from the text file testout.txt using Java FileReader class.
- Here, we are assuming that you have following data in "testout.txt" file:

```
Welcome to G2
```

Example of FileReader

```
import java.io.FileReader;
public class FileReaderExample {
    public static void main(String args[])throws Exception{

          FileReader fr=new FileReader("D:\\testout.txt");

          while((int i=fr.read())!=-1)
            System.out.print((char)i);

          fr.close();
    }
}
```

- Output:

```
Welcome to G2
```

BufferedWriter Class

- Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits Writer class.
- The buffering characters are used for providing the efficient writing of single arrays, characters, and strings.

```
public class BufferedWriter extends Writer
```

BufferedWriter constructors

Constructor	Description
BufferedWriter(Writer wrt)	It is used to create a buffered character output stream that uses the default size for an output buffer.
BufferedWriter(Writer wrt, int size)	It is used to create a buffered character output stream that uses the specified size for an output buffer.

BufferedWriter methods

Method	Description
void newLine()	It is used to add a new line by writing a line separator.
void write(int c)	It is used to write a single character.
void write(char[] cbuf, int off, int len)	It is used to write a portion of an array of characters.
void write(String s, int off, int len)	It is used to write a portion of a string.
void flush()	It is used to flushes the input stream.
void close()	It is used to closes the input stream

Example of BufferedWriter

- Let's see the simple example of writing the data to a text file testout.txt using Java BufferedWriter.
- Output:

```
success
```

- testout.txt:

```
Welcome to G2
```

```
import java.io.FileWriter;
public class FileWriterExample {
    public static void main(String args[]){
         try{
           FileWriter fw=new FileWriter("D:\\testout.txt");

           fw.write("Welcome to G2");
           fw.close();
          }catch(Exception e){System.out.println(e);}
          System.out.println("Success...");
     }
}
```

BufferedReader Class

- Java BufferedReader class is used to read the text from a character-based input stream.
- It can be used to read data line by line by readLine() method.
- It makes the performance fast & inherits Reader class.

```
public class BufferedReader extends Reader
```

BufferedReader constructors

Constructor	Description
BufferedReader(Reader rd)	It is used to create a buffered character input stream that uses the default size for an input buffer.
BufferedReader(Reader rd, int size)	It is used to create a buffered character input stream that uses the specified size for an input buffer.

BufferedReader methods

Method	Description
int read()	It is used for reading a single character.
int read(char[] cbuf, int off, int len)	It is used for reading characters into a portion of an array.
boolean markSupported()	It is used to test the input stream support for the mark and reset method.
String readLine()	It is used for reading a line of text.
boolean ready()	It is used to test whether the input stream is ready to be read.
long skip(long n)	It is used for skipping the characters.
void reset()	It repositions the stream at a position the mark method was last called on this input stream.
void mark(int readAheadLimit)	It is used for marking the present position in a stream.
void close()	It closes the input stream and releases any of the system resources associated with the stream.

Example of BufferedWriter

- In this example, we are reading the data from the text file testout.txt using Java BufferedReader class.
- Here, we are assuming that you have following data in "testout.txt" file:

```
Welcome to G2
```

- Output:

```
Welcome to G2
```

```
import java.io.*;
public class BufferedReaderExample {
    public static void main(String args[])throws Exception{

        FileReader fr=new FileReader("D:\\testout.txt");
        BufferedReader br=new BufferedReader(fr);

        int i;
        while((i=br.read())!=-1){
        System.out.print((char)i);
        }

        br.close();
        fr.close();
    }
}
```

Reading data from console by InputStreamReader and BufferedReader

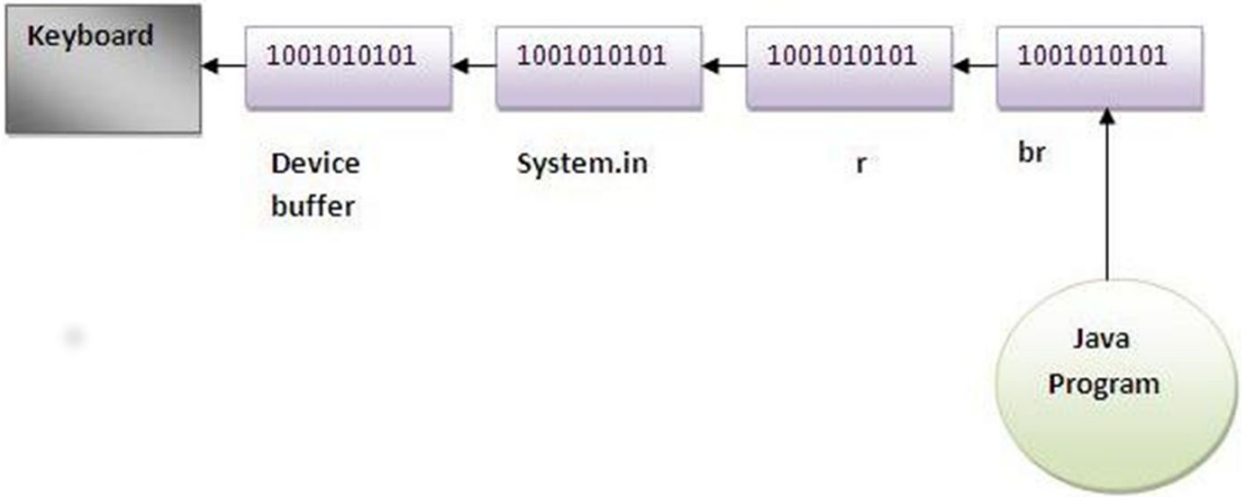
- In this example, we are connecting the BufferedReader stream with the InputStreamReader stream for reading the line by line data from the keyboard..
- Output:

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

```
import java.io.*;
public class BufferedReaderExample{
public static void main(String args[])throws Exception{

    InputStreamReader r=new InputStreamReader(System.in);
    BufferedReader br=new BufferedReader(r);
    System.out.println("Enter your name");
    String name=br.readLine();
    System.out.println("Welcome "+name);
}
}
```

Reading data from console by InputStreamReader and BufferedReader



Reading data from console by InputStreamReader and BufferedReader

- In this example, we are reading and printing the data until the user prints stop.
- Output:

```
Enter data: Nakul
data is: Nakul
Enter data: 12
data is: 12
Enter data: stop
data is: stop
```

```
import java.io.*;
public class BufferedReaderExample{
public static void main(String args[])throws Exception{

     InputStreamReader r=new InputStreamReader(System.in);
     BufferedReader br=new BufferedReader(r);
     String name="";
     while(!name.equals("stop")){
      System.out.println("Enter data: ");
      name=br.readLine();
      System.out.println("data is: "+name);
     }
    br.close();
    r.close();
    }
}
```

TIA-Academy

ASSIGNMENT 02
(HOME ASSIGNMENT)

TIA-Academy

Thank You