Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to CSE , EEE, EIE, ICE, IT)

UNIT NO 3

INPUT/OUTPUT BASICS, STREAMS- BYTE STREAMS

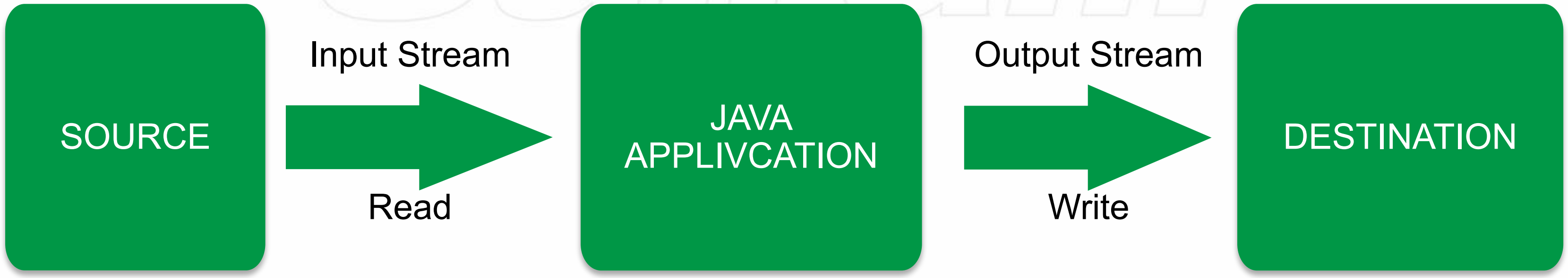
3.6 INPUT / OUTPUT STREAMS
3.6 BYTE STREAMS

COMPUTER SCIENCE & ENGINEERING

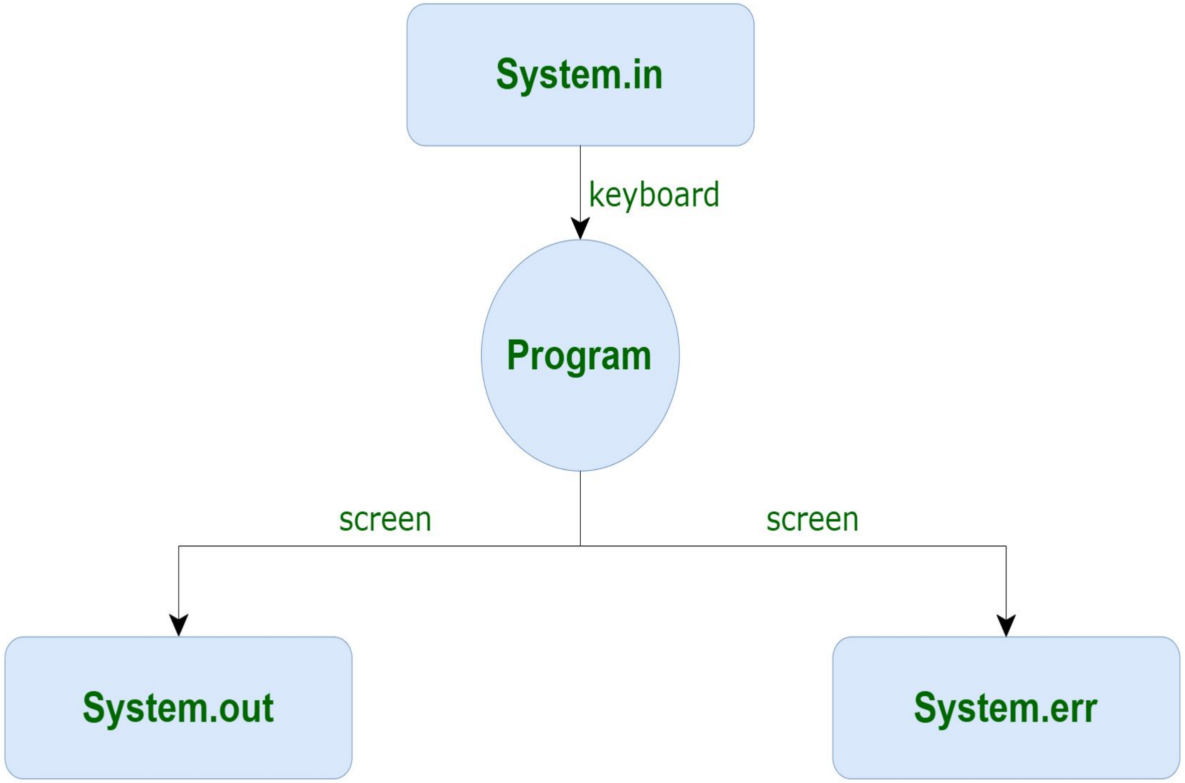
Input/output Streams

Java IO : Input-output in Java with Examples

Java brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations.



3 standard or default streams



Input/output Streams

System.in: This is the **standard input stream** that is used to read characters from the keyboard or any other standard input device.

System.out: This is the **standard output stream** that is used to produce the result of a program on an output device like the computer screen.

println(): This method in Java is also used to display a text on the console. It prints the text on the console and the cursor moves to the start of the next line at the console

System.out.println(*parameter*);

System.err: This is the **standard error stream** that is used to output all the error data that a program might throw, on a computer screen or any standard output device.This stream also uses all the 3 above-mentioned functions to output the error data:

- print()
- println()
- printf()

Example Input/output Streams

```
import java.io.*;
public class SimpleIO {

    public static void main(String args[])
        throws IOException
    {
          // InputStreamReader class to read input
        InputStreamReader inp = null;

        // Storing the input in inp
        inp = new InputStreamReader(System.in);

        System.out.println("Enter characters, "
                           + " and '0' to quit.");
        char c;
        do {
            c = (char)inp.read();
            System.out.println(c);
        } while (c != '0');
    }
}
```

Input:

```
DogsCats
```

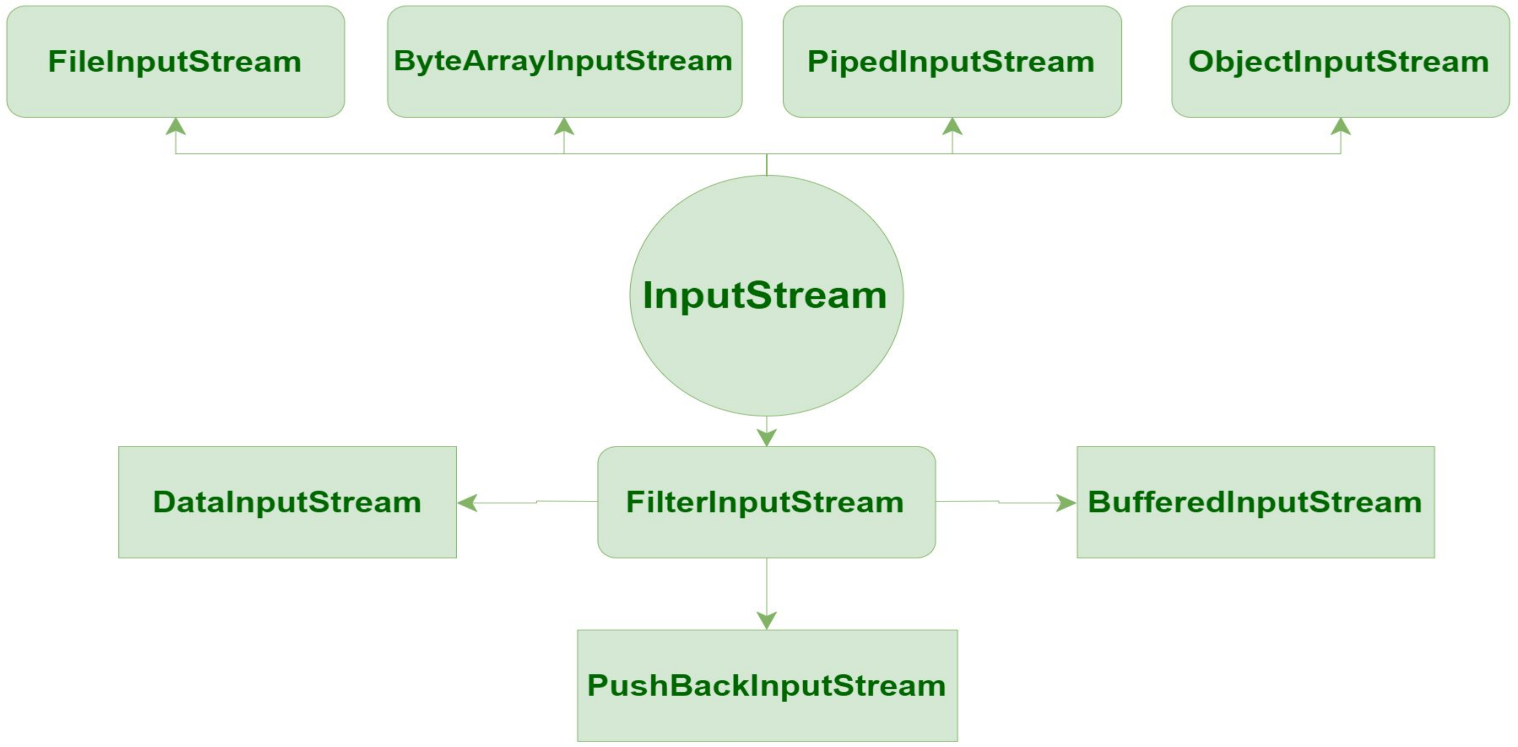
Output:

```
Enter characters, and '0' to quit.
D
o
g
s
C
a
t
S
0
```

Types of Streams

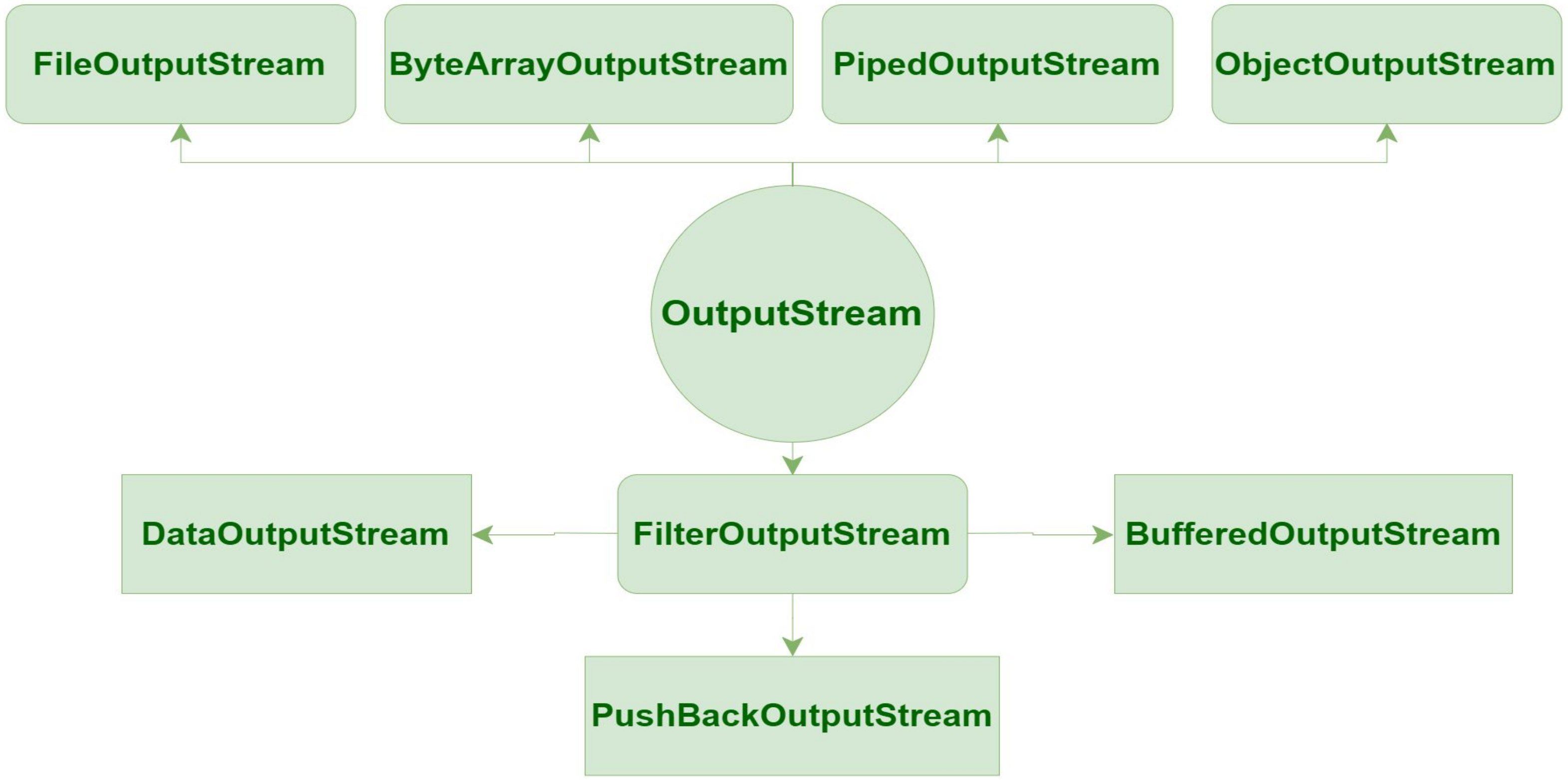
Depending on the type of operations, streams can be divided into two primary classes:

Input Stream: These streams are used to read data that must be taken as an input from a source array or file or any peripheral device. For eg, FileInputStream, BufferedInputStream, ByteArrayInputStream etc.



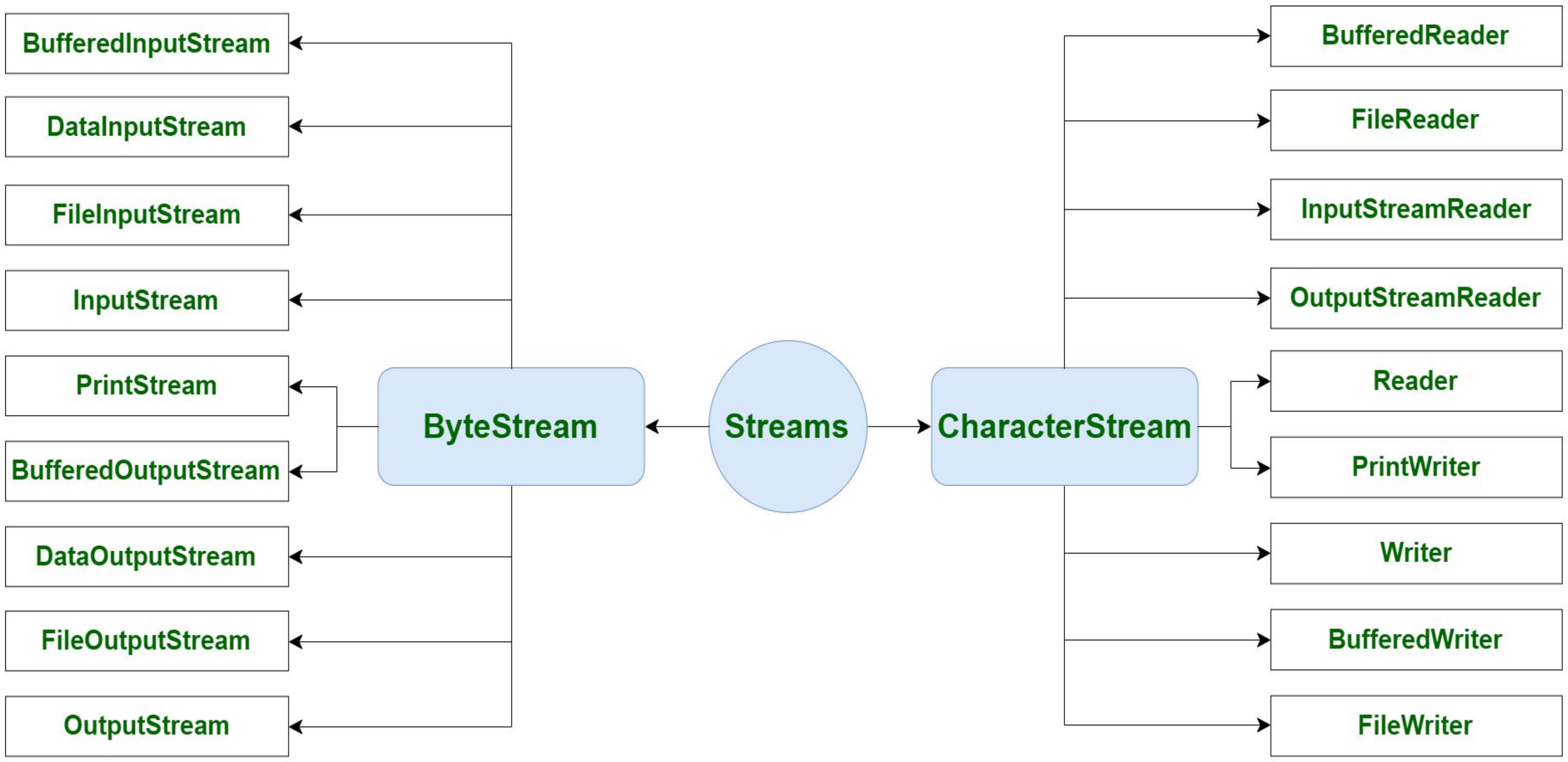
Types of Streams

Output Stream: These streams are used to write data as outputs into an array or file or any output peripheral device. For eg., FileOutputStream, BufferedOutputStream, ByteArrayOutputStream etc.



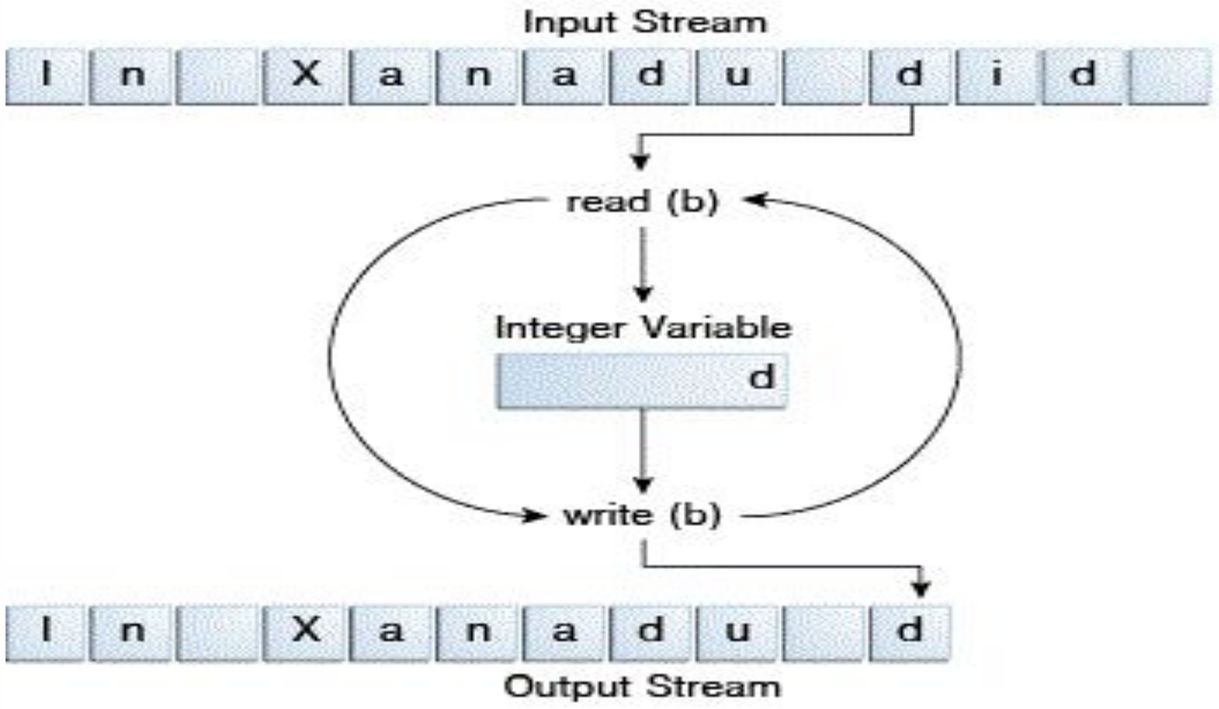
Types of Streams

Depending on the types of file, Streams can be divided into two primary classes which can be further divided into other classes as can be seen through the diagram below followed by the explanations.



Byte Streams

ByteStream: 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:



Byte Streams

Stream Class	Description
BufferedInputStream	It is used for Buffered Input Stream.
DataInputStream	It contains method for reading java standard datatypes.
FileInputStream	This is used to reads from a file
InputStream	This is an abstract class that describes stream input.
PrintStream	This contains the most used print() and println() method
BufferedOutputStream	This is used for Buffered Output Stream.
DataOutputStream	This contains method for writing java standard data types.
FileOutputStream	This is used to write to a file.
OutputStream	This is an abstract class that describe stream output.

Example for Byte Streams

```
import java.io.*;
public class CopyFile
{ public static void main(String args[]) throws IOException {
FileInputStream in = null;
FileOutputStream out = null;
try {
 in = new FileInputStream("input.txt");
out = new FileOutputStream("output.txt");
int c;
while ((c = in.read()) != -1)
{ out.write(c);
} }
finally
{ if (in != null)
{ in.close();
} if (out != null) { out.close(); } } } }
```

Input:
This is test for copy file.

Note: Compile the above program and execute it, which will **result in creating output.txt** file with the same content as we have in input.txt.

Video Link

https://www.youtube.com/watch?v=QfGVi89AV4c