**Java Input Stream Class**

InputStream is an abstract superclass of all classes representing an input stream of bytes. The subclassess of InputStream are AudioInputStream, ByteArrayInputStream, FileInputStream, FilterInputStream, ObjectInputStream, PipedInputStream, SequenceInputStream, StringBufferInputStream. These are used to read data in bytes.

**public** **class** InputStreamDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

File f = **new** File("/Users/sabniss/Desktop/java-training/java-training/src/day\_20/sample1.txt"); // LINE A

InputStream is = **null**;

**if** (f.exists()) // LINE C

{

is = **new** FileInputStream(f); // LINE B

is.close(); // LINE G

System.***out***.println("File exists.");

// LINE D

**int** i = 0;

// LINE E

**while** ((i = is.read()) != -1) {

System.***out***.print((**char**) i); // LINE F

}

} **else** {

System.***out***.println("File not found.");

}

}

}

**Java OutputStream Class**

OutputStream is an abstract superclass of all classes representing an output stream of bytes. The subclass which inherit from OutputStream are ByteArrayOutputStream, FileOutputStream, FilterOutputStream, ObjectOutputStream, OutputStream, PipedOutputStream. These OutputStream sub classes are used to write bytes of data.

**public** **class** OutputStreamDemo {

**public** **static** **void** main(String[] args) **throws** Exception {

File f = **new** File("/Users/sabniss/Desktop/java-training/java-training/src/day\_20/sample1.txt"); // LINE A

OutputStream os = **new** FileOutputStream(f); // LINE B

**if**(f.exists()) // LINE C

{

System.***out***.println("File exists.");

// LINE D

**byte** b[] = {'i',' ','a','m',' ','f','i','l','e','O','n','e','.'};

// Writing into file fileOne

os.write(b);

}

**else**

System.***out***.println("File not found.");

os.close(); // LINE F

// Reading from fileOne

InputStream is = **new** FileInputStream(f);

**int** i = 0;

**while**((i = is.read()) != -1)

{

System.***out***.print((**char**) i);

}

is.close();

}

}

**Buffered Reader**

BufferedReader reads text from a character-input stream, to provide buffering for the efficient reading of characters, arrays and lines.

* The buffer size may be specified, or the default size can be used.
* Each read request causes a corresponding read request to character or byte stream. It is therefore advisable to wrap a BufferedReader around any Reader whose read() operations may be costly, such as [Java FileReader](http://java.meritcampus.com/core-java-topics/java-filereader-or-filereader-in-java) and InputStreamReaders. For example,

BufferedReader in = new BufferedReader(new FileReader("F:\\test.txt"));

this will buffer the input from the test.txt file.

**public** **class** BufferedReaderExample {

**public** **static** **void** main(String[] args) {

**try** {

BufferedReader br = **new** BufferedReader(**new** FileReader("/Users/sabniss/Desktop/java-training/java-training/src/day\_20/sample1.txt"));

System.***out***.println((**char**) br.read()); // LINE A

System.***out***.println((**char**) br.read());

**boolean** ready = **false**;

ready = br.ready(); // LINE C

System.***out***.println("Buffered reader is ready : " + ready);

br.skip(5); // LINE F

System.***out***.println("skipped characters");

System.***out***.println((**char**) br.read());

br.close(); // LINE G

br.read(); // LINE H

} **catch** (IOException e) {

System.***out***.println("You cannot read file because buffered Reader is closed.");

e.printStackTrace();

}

}

}

Read lines:

while((line=br.readLine())!=null)

{

sb.append(line); //appends line to string buffer

sb.append("\n"); //line feed

}

**Buffered Writer**

BufferedWriter writes text to a character-output stream, to provide buffering for the efficient writing of single characters, arrays and strings.

* The buffer size may be specified, or the default size may be used.
* A Writer sends its output immediately to the underlying character or byte stream. It is advisable to wrap a BufferedWriter around any Writer whose Write() operations may be costly, such as [Java FileWriter](http://java.meritcampus.com/core-java-topics/java-filewriter-or-filewriter-in-java) and OutputStreamWriters. For example,

**public** **class** BufferedWriterExample {

**public** **static** **void** main(String[] args) {

**try** {

String content = "This is the content to write into file.";

File file = **new** File("/Users/sabniss/Desktop/java-training/java-training/src/day\_20/sample1.txt");

**if** (!file.exists()) {

file.createNewFile();

}

FileWriter fw = **new** FileWriter(file.getAbsoluteFile());

BufferedWriter bw = **new** BufferedWriter(fw);

bw.write(content); // LINE A

bw.close();

System.***out***.println("Done");

} **catch** (IOException e) {

e.printStackTrace();

}

}

}