

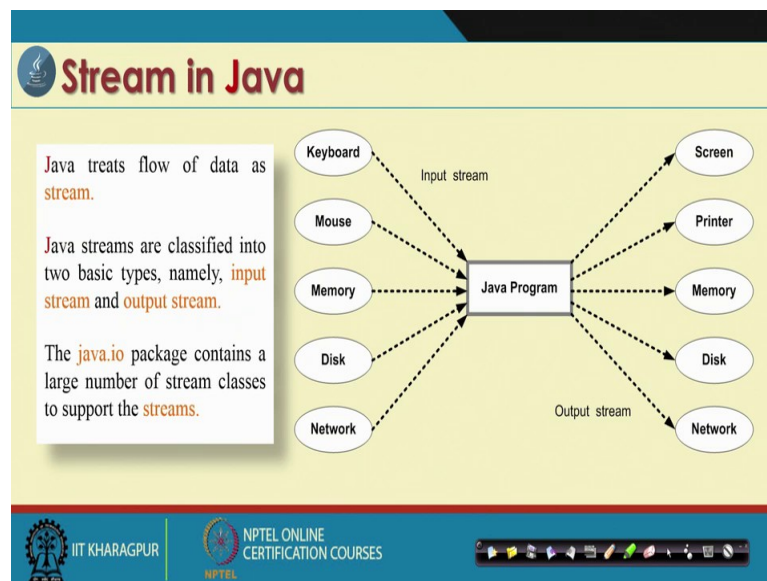
**Programming in Java**  
**Prof. Debasis Samanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 30**  
**I – O Stream – I**

We have learned about how the input-output is possible in Java program. So, the input-output that we have learned so far they are related to standard input and then standard output. Standard input namely the keyboard and standard output is the display screen.

Now, Java provides many more other than this input-output related to only standard input and output devices. So, today's module in today's module we will discuss about what are the different input-output mechanism is possible in Java. So, this is particularly called the input-output streams in Java. Now, again the new term that you are just listening right now is called the stream.

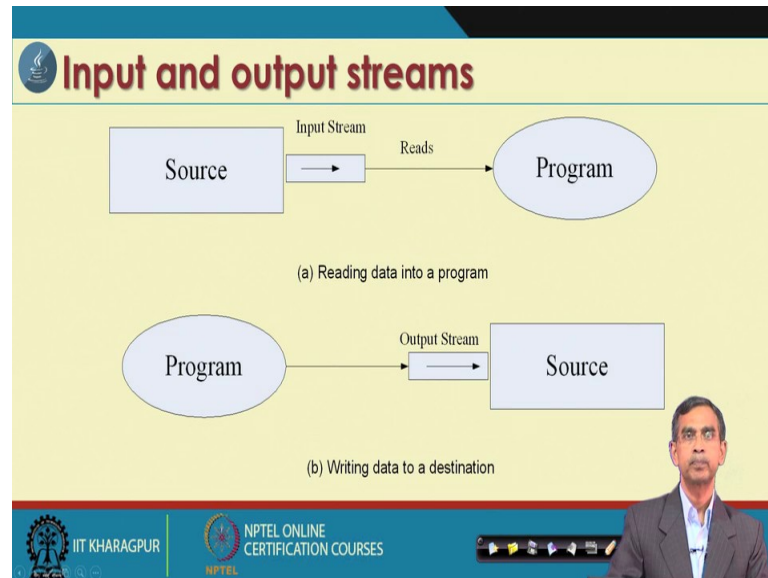
(Refer Slide Time: 01:15)



So, the idea here is that Java treats any flow of data as a stream. Now, Java streams are classified into two major categories. They are called the input stream and then output stream. Now, in order to provide versatility in a programming environment, Java developer provided say very good package deal with input-output stream. This package is called java.io. So, java.io can handle many what is called the types of devices for the source for input as well as many destination devices for the output.

For example, the keyboard, mouse, memory disk, network are the input from where data can be collected. Similarly, data can be streamed into other devices like display unit, printer, memory disk, network like this one. So, there are so many devices that are possible by which the input-output stream can be controlled. As I told you Java provides the java.io packages to dealing with this input-output mechanism.

(Refer Slide Time: 02:35)




Now, basically, the idea about the input stream and output stream is that, that your program can have any data streaming from any other sources. So, it is called the input stream mechanism. On the other hand from your program, you can pump data to some other source, then this is called the output stream which we have discussed in the figure in these slides. So, this is a basic concept of the input-output stream, that means from the program the data will be pushed to some that are the output stream or the program will read something from the input source.

(Refer Slide Time: 03:21)


## I-O stream classes in Java

Java provides `java.io` package which contains a large number of stream classes to process all types of data


- Byte stream classes
  - Support for handling I/O operations on bytes
- Character stream classes
  - Supports for handling I/O operations on characters



IIT KHARAGPUR

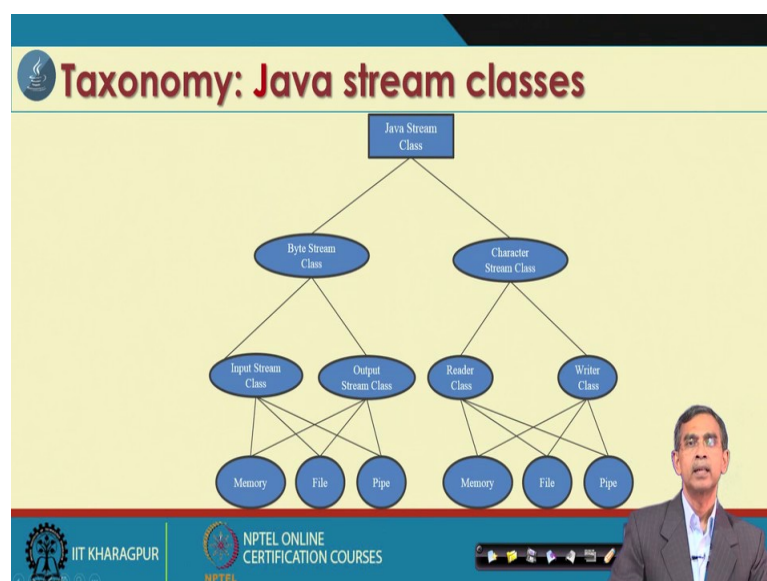


NPTEL ONLINE  
CERTIFICATION COURSES



Now, the `java.io` package is meant for this managing input-output stream and it is an exhaustive package. So now, we will discuss about the basics of what is called the structure in the `java.io` packages. Now, all the classes that are there in Java IO packages `java.io` rather I should say `java.io`, all classes can be categorized into two bar categories. They are called byte stream classes and then character stream classes. This category is basically divided in case of byte stream classes the data input-output mechanism is control in the form of a byte whereas, in the case of character stream classes data input-output mechanism is controlled by means of a stream of characters.

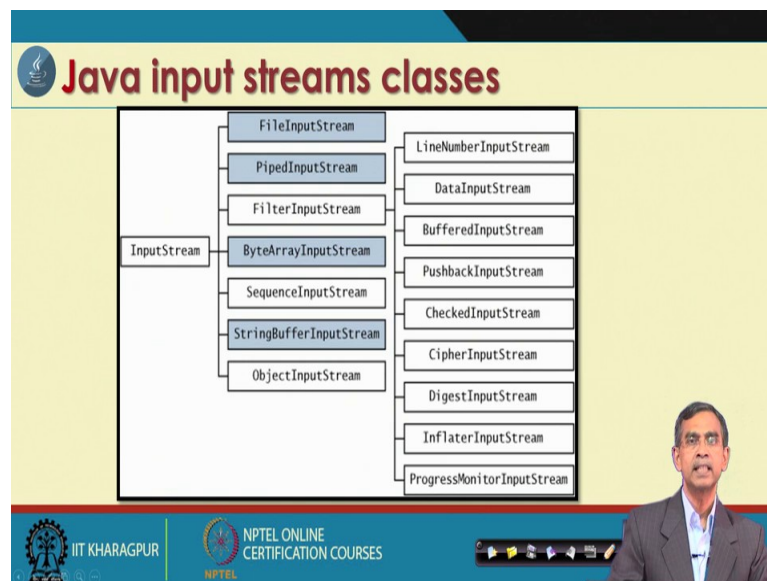
(Refer Slide Time: 04:21)



Now, let us have a quick look at the total different hierarchy of the different packages different classes which are there in this package. We can say that this is the Java stream classes. Now, as I told you it can be divided into two broad categories: the byte stream classes and then character stream classes. In the case of byte stream classes there again two categories called the input stream classes and then output stream classes. And so, for the character stream classes is concerned again there are two categories: reader class and then writer class.

So, these are the so, many classes are there and essentially all these classes are to deal with the different sources as an input as well as output like memory, file, pipe. Now, a pipe is basically related to the network channel. Now, let us first discuss each one class each class individually one by one. So, first let us have a discussion on Java input stream classes that is there in java.io package. Now, as I told you input stream classes are basically mean for in reading something from memory or from a file or from a pipe and as it belongs the byte stream class means, it will read in the form of a byte.

(Refer Slide Time: 05:41)

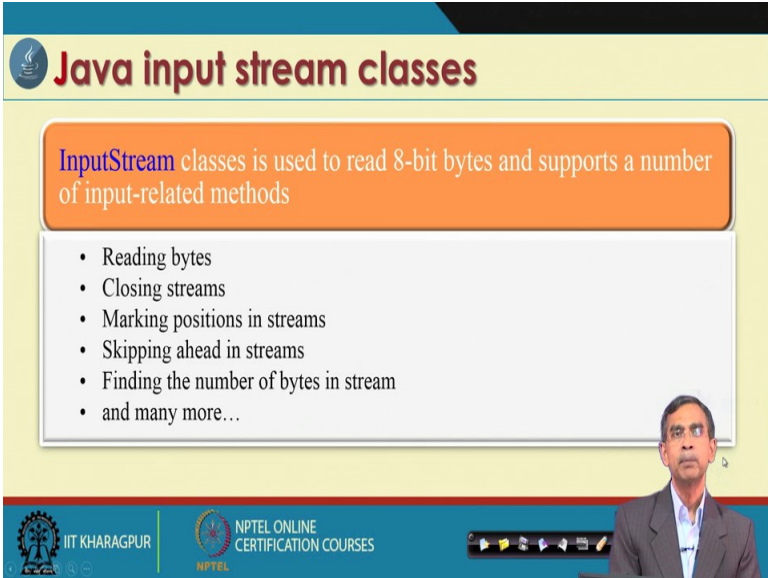


So, here is basically the total classes those are there in input stream classes as we see a large number of classes are there. And, all this classes can be mentioned like the `FileInputStream`, then `PipedInputStream`, `FilterInputStream`, `ByteArrayInputStream`, `SequenceInputStream`, then `StringBufferInputStream` and `ObjectInputStream`. Now, here basically so, many classes belong to input stream class because they are basically to deal

the different input in a different manner. For example, byte array input stream class is basically how to read data from an array which is stored in memory, it is just an example.

There are so, many classes like and then example again filter input stream class if we consider there are again many different classes belong to this class hierarchy it is. For example, `DataInputStream`, `DataOutputStream`, `DataInputStream`, `BufferedInputStream`, `PushbackInputStream` etcetera. So, we see that there is a lot of classes and I have just mentioned only one type it is called the filter input stream. Like likewise byte input stream there again are the subclasses are there. So, our objective here will be to learn it is not possible to learn all classes one by one, but at least some classes we can learn which are most frequently used.

(Refer Slide Time: 07:11)

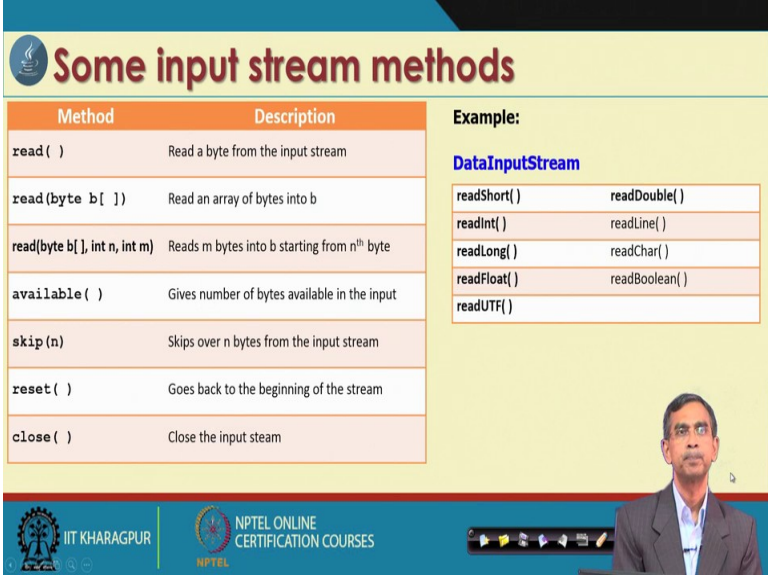


The slide is titled "Java input stream classes" in a large, bold, dark red font. Below the title, there is an orange box containing the text: "InputStream classes is used to read 8-bit bytes and supports a number of input-related methods". Below this, a white box with a thin border contains a bulleted list of methods: "Reading bytes", "Closing streams", "Marking positions in streams", "Skipping ahead in streams", "Finding the number of bytes in stream", and "and many more...". In the bottom right corner of the slide, there is a small inset image of a man in a suit, likely the speaker. At the bottom of the slide, there are logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES".

- Reading bytes
- Closing streams
- Marking positions in streams
- Skipping ahead in streams
- Finding the number of bytes in stream
- and many more...

Now so, input stream classes as I already mentioned that it belongs to the byte stream classes. This means it is used to read 8-bit bytes and supports a number of input related methods. And, few methods which I have mentioned here which are very common in any programming; like say how we can read a byte from an input source, how we can close an input source, how we can make a position of streaming in an input source. And, it is skipping ahead into some if we want to read the input one source, not in a sequential manner rather in a random manner. And, then we can also find a total number of bytes in stream etcetera. There are many such methods that includes in the input stream classes to process it.

(Refer Slide Time: 08:03)



**Some input stream methods**

Method	Description
<code>read( )</code>	Read a byte from the input stream
<code>read(byte b[ ])</code>	Read an array of bytes into b
<code>read(byte b[ ], int n, int m)</code>	Reads m bytes into b starting from n <sup>th</sup> byte
<code>available( )</code>	Gives number of bytes available in the input
<code>skip(n)</code>	Skips over n bytes from the input stream
<code>reset( )</code>	Goes back to the beginning of the stream
<code>close( )</code>	Close the input stream

**Example:**

**DataInputStream**

<code>readShort( )</code>	<code>readDouble( )</code>
<code>readInt( )</code>	<code>readLine( )</code>
<code>readLong( )</code>	<code>readChar( )</code>
<code>readFloat( )</code>	<code>readBoolean( )</code>
<code>readUTF( )</code>	

The slide also features the IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES logos at the bottom, along with a small video feed of a presenter.

And, the methods basically belongs to all classes are overridden method. For example, the read method that can be used for any say for byte array input stream, we can use the read array method for that. It is basically the overridden method that, those are there in each class. So, in general, a read method as we see that read a byte from the input stream. Now, this is an overridden method read byte b; that means, read an array of bytes into b. So, it will read and store into a temporary location that is there in the program. And, read also has another overridden method like byte b int n int m.

So, it is basically reading m bytes starting from n and put into the array say b. Available is another method it says that whether the input stream currently available or not, skip is basically for random accessing. So, it is skipping n means, it will skip from the current location to n bytes ahead and then reset also, reset the reading of location of in the input stream. Reset means it will start from the beginning and then close basically every stream has to be as it is open; so, as needs to be closed also. So, these are some standard methods we will learn all these methods when we will have some more detail explanations regarding a particular stream class.

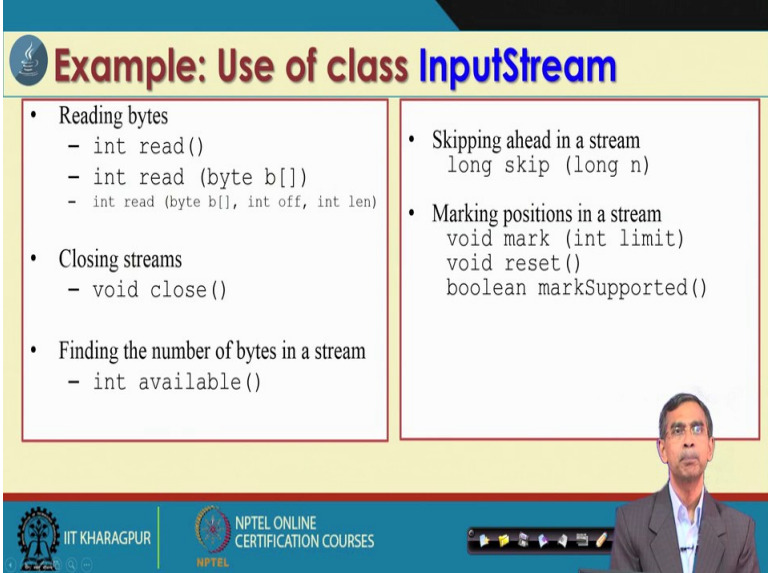
Now, here again another example data inputs term is a very popular one stream class and as we see they are also many methods that can be used. So, basically, the data input stream belongs to this byte; byte stream classes and it basically has many what is called



the utility. So, it can read as a sort integer, it can read as a floating-point number, it can read as a read normal integer, it can read at a line ok.

It can read long integer, it can read a character, it can read a Boolean, it can read a read some data in a UTF format; there are many other formats are there. So, so basically it basically tells that a data input stream, if we have an in, have an input then, we can read the data from that input source in a variety of ways. That is why the different methods have been planned and according to your requirement you can fix that which method you should invoke in order to read a data.

(Refer Slide Time: 10:35)



**Example: Use of class InputStream**

- Reading bytes
  - `int read()`
  - `int read (byte b[])`
  - `int read (byte b[], int off, int len)`
- Closing streams
  - `void close()`
- Finding the number of bytes in a stream
  - `int available()`
- Skipping ahead in a stream
  - `long skip (long n)`
- Marking positions in a stream
  - `void mark (int limit)`
  - `void reset()`
  - `boolean markSupported()`

The slide is part of an NPTEL presentation from IIT Kharagpur. It features a list of methods for the `InputStream` class, categorized into reading bytes, closing streams, finding bytes, skipping ahead, and marking positions. A small video inset of a presenter is visible in the bottom right corner of the slide area.

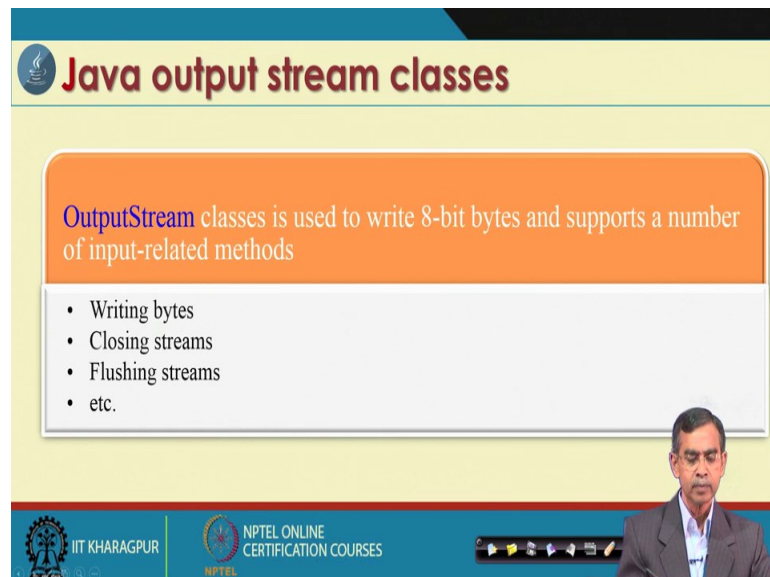
And, this is the syntax for example, whenever you declare a read is basically read an integer return an integer value and like this. So, read byte b we have already discussed about; so, it read as a byte and store into then temporary array b and like this one. And so, these are the few methods which will learn while we will have a more detail discussion on it. And so, the method is for example long skip void long skip are a parameter is a long integer. And, then it will skip to some location depending on a parameter values first, mark int is basically, it will mark a particular position in a stream.

And, then reset as I told you it is basically reset the file position, I mean stream position and Boolean markSupported whether any location stream can be supported or not, it will return true or false like. So, these are the many methods that are required to control our input mechanism, those are already defined in the in-stream input stream class. Likewise,

there is a class called the Java output stream classes. If the input stream is to read something from the input source the output stream is just opposite. It will push the data into some output destination.

Now, likewise the input stream classes it also belongs to the byte stream classes. These means that it will write something in the form of a byte and this class has again can output data into memory. It can output data into a file; it can output data into a network channel. So, it is same as the input stream, but only the thing is that input is taking and output is pushing.

(Refer Slide Time: 12:25)



The slide is titled "Java output stream classes" in a blue header. Below the title, an orange box contains the text: "OutputStream classes is used to write 8-bit bytes and supports a number of input-related methods". Below this, a white box with a grey border contains a bulleted list of methods: "Writing bytes", "Closing streams", "Flushing streams", and "etc.". The slide footer includes the IIT Kharagpur logo, the NPTEL logo, and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset in the bottom right corner shows a man in a suit.

## Java output stream classes

OutputStream classes is used to write 8-bit bytes and supports a number of input-related methods

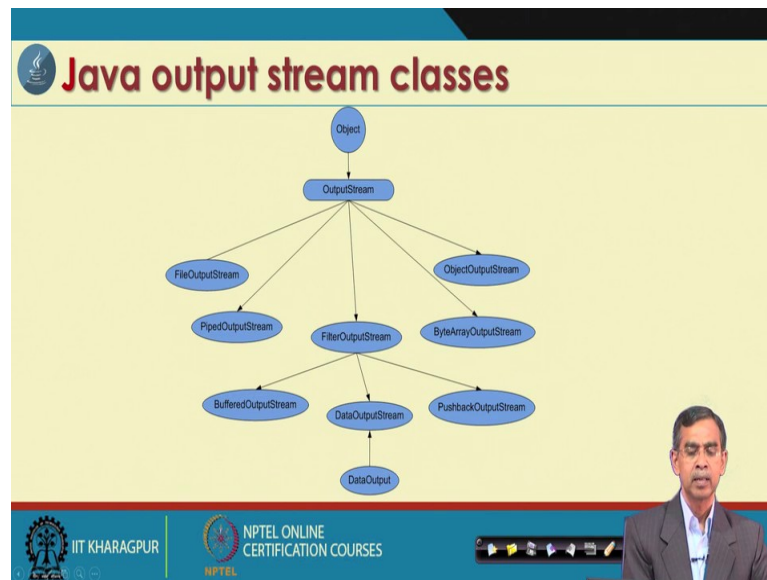
- Writing bytes
- Closing streams
- Flushing streams
- etc.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, output stream classes are therefore, used to write 8-bit bytes and like input stream classes, it also supports a number of methods. For example, writing a byte into an output source, output destination, closing the stream, flushing stream and etcetera.



(Refer Slide Time: 12:47)



Now, like the input stream also it has many methods in it. Now, one particular class belong to this Java OutputStream is called file and then the FilePipeOutputStream, BufferOutputStream, FilterOutputStream. FilterOutputStream, if we consider there again varieties DataOutputStream, PushbackOutputStream and like this one. So, it is like this from different sources mean into a different source, how we can put data into that; so, that is why it is there. For example, if we consider saying the byte array output stream is basically telling that how we can put some data into an array in memory. It is an example like this one.

(Refer Slide Time: 13:33)

Method	Description
<code>write ( )</code>	Write a byte from the input stream
<code>write (byte b[ ])</code>	Write all bytes in the array b to the output stream
<code>write (byte b[ ], int n, int m)</code>	Write m bytes from array b starting from n <sup>th</sup> byte
<code>close( )</code>	Close the output stream
<code>flush( )</code>	Flushes the output stream

**Example:**

**DataOutputStream**

<code>writeShort( )</code>	<code>writeDouble( )</code>
<code>writeInt( )</code>	<code>writeLine( )</code>
<code>writeLong( )</code>	<code>writeChar( )</code>
<code>writeFloat( )</code>	<code>writeBoolean( )</code>
<code>writeUTF( )</code>	

So, it is again there are many methods like write method; it basically gives an idea about how we can write a bite into a stream output stream write byte b. It basically say that an array of bytes how it can write into an output source, write byte b int n and int m. So, it basically says that how m bytes from the array b starting from n bytes can be pushed into an output destination and then close and flush is basically closing an output stream. And, flush is basically output stream is basically clearing cleaning, it is basically called output stream usually temporarily buffer. And, then whenever you read whenever you push something before placing it to an output we need to clean it.

If you do not clean it the previous stream that can be used it can be stored in the their buffer, that is why it is flushed. And, like the data input stream, there is one class called the data output stream. It is like the data input stream as it was it can write data into many forms starting beginning with the bytes actually. So, is writeShort means write a sorted integer, write Double it basically writes a floating-point number, writeInt, writeLine.

It is basically written a stream consisting of numbers or characters say together, it is called the writeLine. Then writeLong is basically writing a long integer, write character writing a character into the output stream, writeFloat and then WriteBoolean and writeUTF. So, writing data into UTF format. So, these are the different methods those are there in the data output stream.

(Refer Slide Time: 15:33)

**Use of class OutputStream**

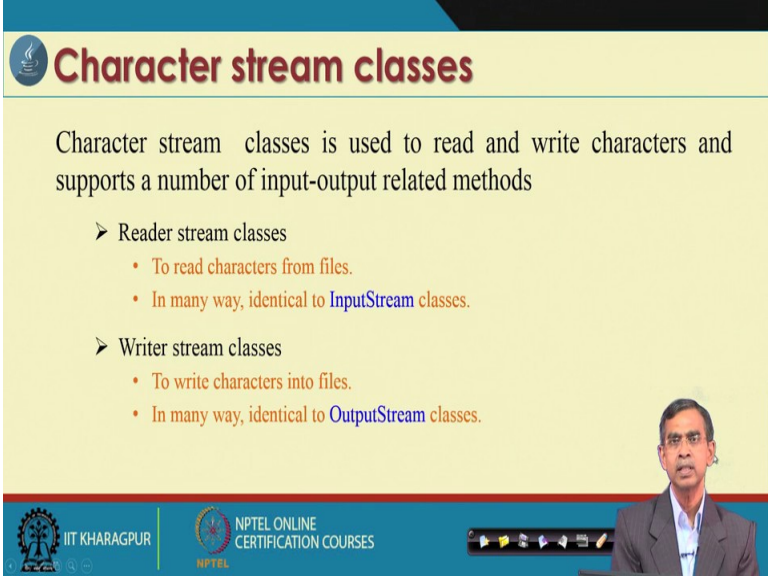
- Writing bytes**
  - void write (byte b)
  - void write (byte b[])
  - void write (byte b[], int off, int len)
- Closing a stream**
  - void close()
- Clearing a buffer**
  - void flush()

The slide is a presentation slide with a yellow background and a blue header. The header contains the IIT Kharagpur logo and the text 'Use of class OutputStream'. The main content is organized into three sections, each with an orange header box and a white content box. The first section is 'Writing bytes' with three bullet points. The second section is 'Closing a stream' with one bullet point. The third section is 'Clearing a buffer' with one bullet point. In the bottom right corner, there is a small video inset of a man in a suit. The bottom of the slide features a blue footer with the IIT Kharagpur logo, the text 'NPTEL ONLINE CERTIFICATION COURSES', and a navigation bar with icons.

Now, here is basically syntax that writes is basically return does not anything, this method it takes a byte b or just simply the byte that needs to be written into the output stream. It is an overridden method, there are three different overridden method as we see we which we have mentioned here three overridden method here. And, this is a close method and this is the syntax of the method void close, void flush that means they do not return anything just do the work for the output stream.

Now so, these are the byte stream classes we said they are reading as a byte form. Now, another way of input-output streaming it is called characters stream classes. Now, likewise there are many different methods that are there and broadly all the character stream classes can be divided into reader class and writer class. So, the reader class just like an input stream classes will read some data in the form of a character; from either memory location, from a file or from a network channel. And, write class is basically the same as the data output streaming, it basically writes some data into memory or in a file or it can pass the data into a network channel.

(Refer Slide Time: 16:59)



**Character stream classes**

Character stream classes is used to read and write characters and supports a number of input-output related methods

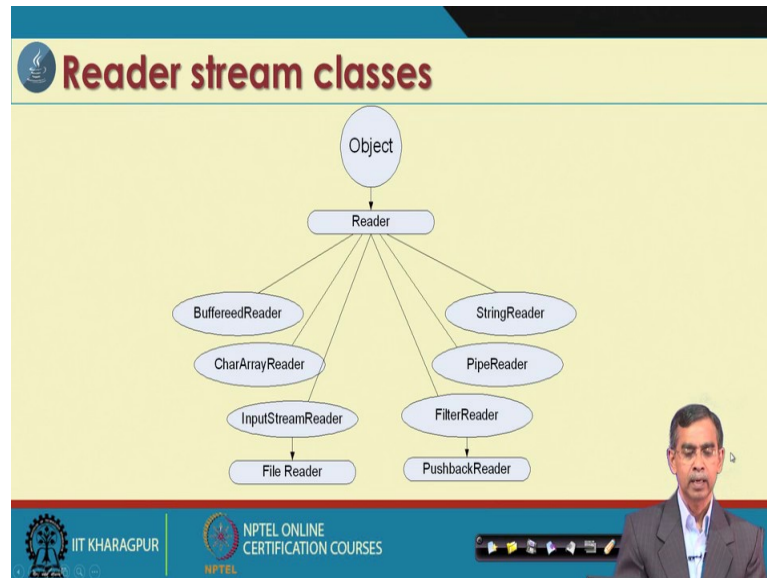
- Reader stream classes
  - To read characters from files.
  - In many way, identical to **InputStream** classes.
- Writer stream classes
  - To write characters into files.
  - In many way, identical to **OutputStream** classes.

The slide features a blue header with the title 'Character stream classes' in white. The main content area is yellow with black text. A small circular logo is in the top left. The bottom of the slide has a blue footer with logos for IIT Kharagpur and NPTEL, and a small video inset of a man in the bottom right corner.

Now, so these are the two classes and they are mostly in the same manner as this one. The difference is that the byte classes is basically deal the data in the form of bytes whereas, the character stream classes deal the data in the form of characters. Now so, reader stream class as I mentioned that it basically this class is to read characters from some input sources. And, it is in many ways very similar to **InputStream** classes. On the

other hand write stream classes writer stream classes basically, write characters into the output destination targets. It is also in many ways very similar to the OutputStream classes.

(Refer Slide Time: 17:41)



Now, here is the reader stream classes. So, reading input from some input source in the form of a character. And, as we see in this figure it can read like a BufferedReader, it can read like a character array reader, it can read like a File Reader. I am reading from a file, it can read just a StringReader read as a string of characters; it can read as a PipeReader, FilterReader, PushbackReader. So, there are different o a the reading is possible using the reader stream classes it is mentioned here.

(Refer Slide Time: 18:23)

List of Important tasks and their Classes		
Task	Character Stream Class	Byte Stream Class
Performing input operations	Reader	InputStream
Buffering input	BufferedReader	BufferedInputStream
Keeping track of line numbers	LineNumberReader	LineNumberInputStream
Reading from an array	CharArrayReader	ByteArrayInputStream
Translating byte stream into a character stream	InputStreamReader	(none)
Reading from files	FileReader	FileInputStream
Filtering the input	FilterReader	FilterInputStream
Pushing back characters/bytes	PushbackReader	PushbackInputStream
Reading from a pipe	PipedReader	PipedInputStream
Reading from a string	StringReader	StringBufferInputStream
Reading primitive types	(none)	DataInputStream
Performing output operations	Writer	OutputStream
Buffering output	BufferedWriter	BufferedOutputStream
Writing to an array	CharArrayWriter	ByteArrayOutputStream
Filtering the output	FilterWriter	FilterOutputStream
Translating character stream into a byte stream	OutputStreamWriter	(none)
Writing to a file	FileWriter	FileOutputStream
Printing values and objects	PrintWriter	PrintStream
Writing to a pipe	PipedWriter	PipedOutputStream
Writing to a string	StringWriter	(none)
Writing primitive types	(none)	DataOutputStream

Now, here is the one quick summary of the different methods we have mentioned here and it is just for a piece of information. And, the details and everything is really very difficult to discuss. As you see here that we have mentioned that, if we want to accomplish a particular task then which classes that we can consider.

Whether it is basically character stream classes or byte stream classes, as we see say suppose we can perform an input operation. If we want to do so, it can be done using both character stream class as well as the byte stream class. As an example as we see here performing input operation can be done using character read class Reader classes which are there in character stream classes or InputStream classes that are there in byte stream classes.

Similarly, if we want to perform a task say buffering input, buffering an input means storing some data into an into a buffer every what is called the input source, in general, has their own buffer. For example, a keyboard has its own buffer. So, whenever you are typing from our keyboard it is initially going to the keyboard buffer and from that keyboard buffer any program can read, either read as a character or it can read as a bytes. So, if we want to read as a character, then the class that you should consider BufferedReader and if we want to read as a byte; so, BufferedInputStream.

This means that if we want to read a buffer then and by means of say BufferedReader then we have to create an object of the class BufferedReader and then that object has its

method. For example, read method or the different methods as we have mentioned and you can invoke the method for that object to complete the particular task. So, it is like this as a particular example, if we want to write writing to a file. So, there is an again both character stream classes as well as write byte stream classes the method is there.

As we see writing to a file has the FileWriter, if we want to use the character stream classes and then FileOutputStream if we want to use the byte stream classes like this one. So, these basically give an idea about if we want to perform, not all I have mentioned only a few tasks that are very important. These are the different task if we want to perform then if we want to use a character stream class, then we have to use this one. And, if we want to use a byte stream class then we have to use this one. So, these are the just for the information that what are the different methods are there.

But, all these things are not possible to learn in just one module. So, we will follow step by step methods using each class and then each task rather than, how it can be done. So, this is the better way so, that we can learn about the input-output stream classes into the program ok. So, this is basically the topics for today. And, in the our next module we will discuss about how step by step method to handle the different way of input-output mechanism using in I O stream concept in Java programming.

Thank you very much.