Data Structures and Algorithm using Java

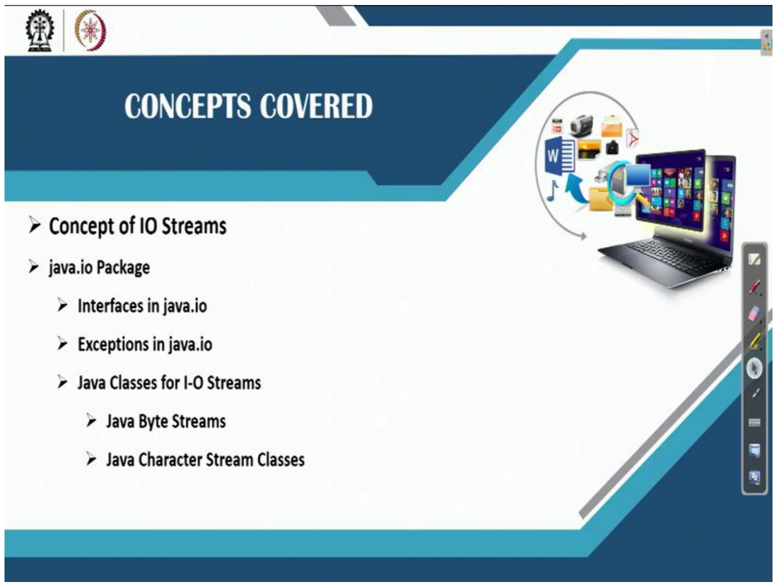
Professor Debasis Samanta
Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur

Module 12
IO Streams and Files
Lecture 42
Introduction to Java.io

Let us start one another important concept related to Programming with Java, particularly data structure related programming, it is basically handling data. So, data can be available to your program from different source, different sources like file it can be from network line or it can be from another program. So, there are many sources rather from which the data can be available to your program. Also, you may need to push data from your program to different destinations like you want to store data after manipulation into a file, or you can pump data to some other network line or like this one or any database system whatever it is there.

So, data therefore needs to be processed from several sources as well as to different destinations. So, in this regard. Java developer has proposed one very elegant and systematic package. This package is called Java dot io package.

(Refer Slide Time: 2:02)



CONCEPTS COVERED

- Concept of IO Streams
- java.io Package
  - Interfaces in java.io
  - Exceptions in java.io
  - Java Classes for I-O Streams
    - Java Byte Streams
    - Java Character Stream Classes

So, today our topics basically is related to learning the java dot io package, and actually this java dot io package is very exhaustive one package actually, many classes and in each class, many methods are there. Around 70 different classes are there around so many interfaces also there. And for each classes around, on the average, maybe 20 methods are there.

So, you can understand so many things are already define. So, it is bit exhaustive and then very voluminous on right package we can say. And try to understand about the concept in as much as details it is possible. So, this module we consists of few lecture videos, we will cover it one by one. So, today it is an introductory coverage. So, in this class, we will cover basic concept about the stream rather we can mention more specifically, it is io stream, io stands for input output.

And we will discuss about the basic architecture about the java dot io package. There are many interfaces. The many classes, few classes are related to the exception handling also are there. Now, broadly all the classes those are defined in java dot io packages can be divided into two broad categories, they are called byte stream classes and character stream classes.

So, today we have an introduction about these byte stream and characters stream classes. And then in our subsequent discussion, we will study all these stream classes in detail with elastration as much as possible. So, today now let us first discuss about the concept of io stream. So, as you know stream is basically is a flow, so is a flow of data. So, flow of data in Java it is called the stream. Now, you consider the data, how actually data, data are stored in computer memory.

So, in the raw form, the data is stored in the form of a byte few bytes for say, integer 4 bytes or long maybe 8 bytes like this one. So, is a basically bytes as a raw form of data. So, byte is the one sort of stream. So, whenever we receive data from say file, actually we receive in the flow of bytes. So, it is called the byte stream actual stream of bytes. On the other hand, there is an another concept is called the character stream.

So, here I want to say few things about, see all these bytes are machine friendly rather machine can understand better. But we the user usually familiar to the data, which we can understand. For example, some data we say that it is integer or numeric data, some data we can say that is a character or string type or something like mixing of both integral character and whatever it is there.

Now, for this purpose, Java programming language like any other programming language, they provide certain data types. Those are called primitive data, like int, long, short, Boolean, char, string also on sort of type in that sense, of course, although it is not a data type it is an of class rather object anyway. So, these are the different primitive data types are there. What I want to say here, that from the programmer point of view, all data are primitive data, whatever the data we manipulate, they are in the form of primitive data ultimately, but at the other end, when it goes to computer, it basically format the byte from so byte data.

So, if we the programmer, right one to manipulate data better we like the data should be should flow in the form of a primitive type. Now, how this primitive type of flow can be there or possible? In this regard again Java is another what is called the unique programming language, which supports all data or whatever primitive type, whatever you can say can be stored in some form. Particularly Java, in order to make it platform independent.

It follows some unique format or rather unique code to represent all what is called the alphabets, rather symbols that a programming language can handle. Now, you probably know C or C++ programming language, they follow all the symbols or language alphabets using only 256 symbols it is, some are printable, some are non-printable, and they are basically in the form of ASCII code. ASCII code is basically international standard organization they provided a code and that is the ASCII code actually, and it is basically 8 bit code.

So, 0 to 255 is the value of each code like. For example, capital A the ASCII code of capital A is 65 and Z is 91 like this one. So, they are different, what is called the ASCII code have been formulated, and it basically machine follows these code. So, whenever we type capital A from our keyboard ultimately it interchange, according to these ASCII that is called American Standard Code of Information Interchange, we will convert it into the bytes, so it is basically 65 we convert in the binary bits then it will be there.

So, 65 as we know as capital A actually it is stored in the form of a binary of 65 numbers. So, this is called the ASCII but in Java, Java follow a little bit sophisticated more smarter way of reading code or representing code, and it follows Unicode. Unicode is a coding format where unlike the ASCII it is basically 16 bits rather we can say 2 bytes code it is there. And this Unicode follow different encoding format.

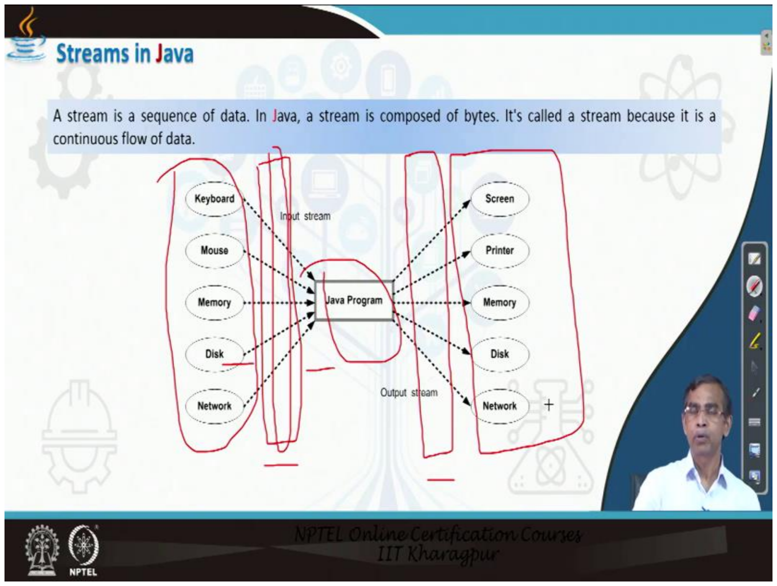
There are many encoding formats. Those are known and those are format is used for our web page development, JavaScript or PHP or any other programming purpose in internet programming. They are called UTF8 and UTF16. So, it is called the universal transformation format. It is 8 means, it is 8 it can be one byte. It can also varies from 1byte, 2byte, 3byte, 4byte is a concatenated. But UTF 16, it is basically a better way of converting unicode to this unified transforms format that is for the encoding.

So, Unicode is basically is a character set and as you can understand that which say 8 bits ASCII code you can have limited 0 to 255, that means 256 different codes. But using Unicode, you can have many different representation rather. And one more important thing is that the Unicode set varies from language to language. So, there is a different Unicode values for say, Hindi Devanagari language, similarly the different for Chinese for different other languages also.

Now, what is the advantage of storing the data in this Unicode format is that it basically makes platform independent and it can also allow you to represent the information or data in any language specific actually. It can be in Bengali, it can be Hindi, or this one. So, in that sense I can say that there are certain java programming tools also available, which you can consider to write your programs in Chinese language or in French language like this one also. Anyway, so it is possible because of this Unicode standard that java follow actually.

Now so here the character stream is basically is a stream of Unicode flow it is there, which is more readable from the programmer point of view. But from the machine point of view is a byte stream is more readable.

(Refer Slide Time: 11:52)


Streams in Java

A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is a continuous flow of data.

So, these are the concepts it is there. Now let us have idea about this one that byte stream verses character stream, as you have already told about that byte stream, is basically is a in the form of a binary bits, where character stream is in the form of a Unicode.

Now the actually this stream concept coming into the picture, because your programs should manipulate data and this data can be obtained from the different source. And that data can be in different stream, like maybe byte stream can be character or whatever it is there. So, what exactly the requirement is that, as you have mentioned here, that these are the different source of data. It can be file, it can be network, it can be some other program or whatever it is there. It can be standard device input like keyboard, like this one.

So, your program should have the capability to read data from different type of source of data. Now, these capabilities therefore is a very difficult because the data that can be in different form. And different source have different ways to manipulate data and which is very difficult if the programmer has to deal so many things here. Now java programming has been planned so that it can be made universal and it can be applied to anywhere in any settings, any platform.

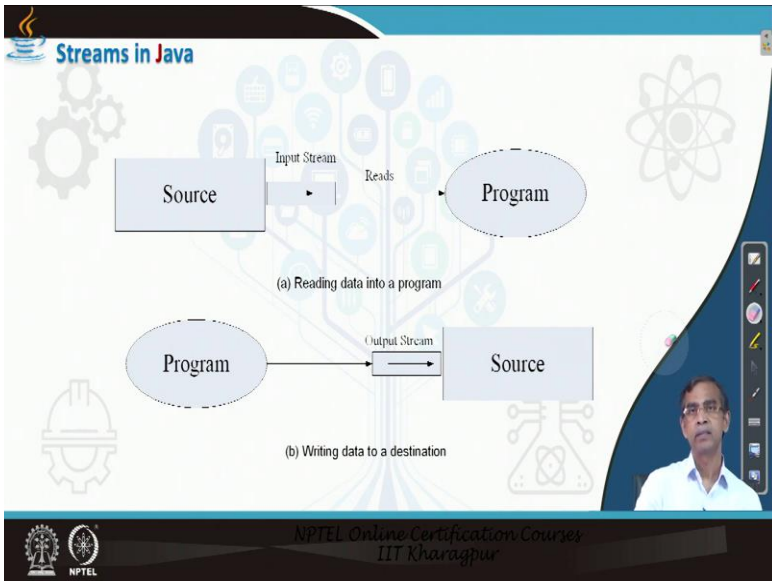
So, that is why there are certain interfaces are require by which program can deal with the source of data through this interface very comfortably, very conveniently, that the programmer can have it. So, this interface basically communicate the device and your program (())(13:48). So, this interfaces is basically by means of different what is called the classes that can basically handle the inputs. So, these classes are called inputs stream classes or characters stream classes we can say. Now just like these input sources, the data can be flow to different targets or destination.

The destination can be like simple consoler display monitor. It can be printer, it can be file again or it can be network line, whatever it is there. Now, for your program, if you want to send data to a network line, then you have to manipulate or implement many what is called the programming methods or programming codes for that purpose. And which is basically a tedious job because for different targets, the different manipulation techniques are to be programmed.

So, that is why again java developer proposed one interface here. This interface basically deals your output stream properly. That means from your program, these output stream classes or concept will consider your data to be float into the destination very comfortably or conveniently. So, there are two streams called input streams and output streams are to be there so that your program can read data and can write data. For input source to output source perfectly.

So, this java dot io basically provides this streams concept and this streams concept in the two form the byte stream and character stream to facilitate this concept, this idea actually or this mechanism. Now, so java dot io package, is therefore there.

(Refer Slide Time: 15:50)

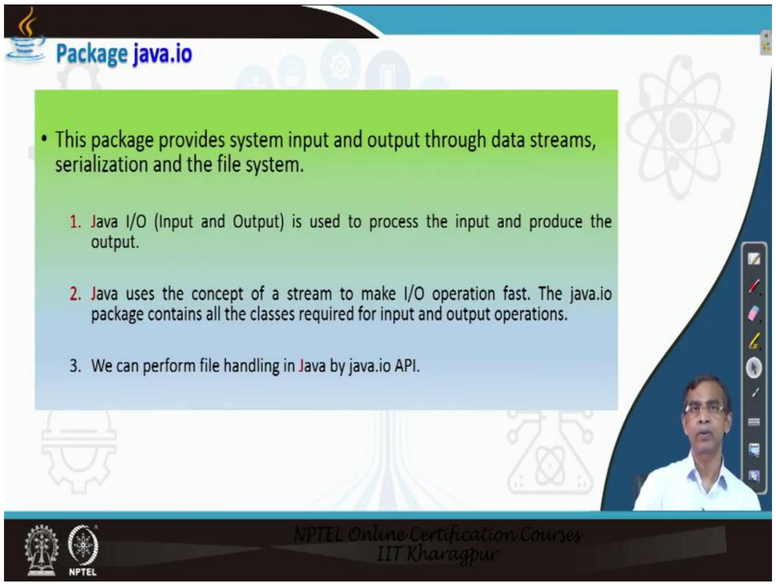

Streams in Java
Source
Input Stream
Reads
Program
(a) Reading data into a program
Program
Output Stream
Source
(b) Writing data to a destination
NPTEL Online Certification Courses
IIT Kharagpur

So this package is this package rather what you can say is that have the all sort of operations or methods rather we can say by which it can read data from any type of input source. And they are also method like it is reading from input source, it can also write data to any output source from your program. So, we will study about how your program can have this kind of sending data or receiving data from output source or to output source or from input source by means of input stream and output streams things it is there.

So, now let us consider the concept about this, as I told you, input stream and output stream. These are the two concepts and these two concepts are to be supported to the okay, to be supported. So, java developer has provided a package for this purpose. This package is, as I told you java dot io package. And from JDK 8 onwards the java developer has introduced another new package, it is called the new input output, so this package is called Java dot nio.

Java dot nio far, far faster than the existing java dot io package having more utility, more facilities, but in this course the java dot io package will not be covered.

(Refer Slide Time: 17:29)



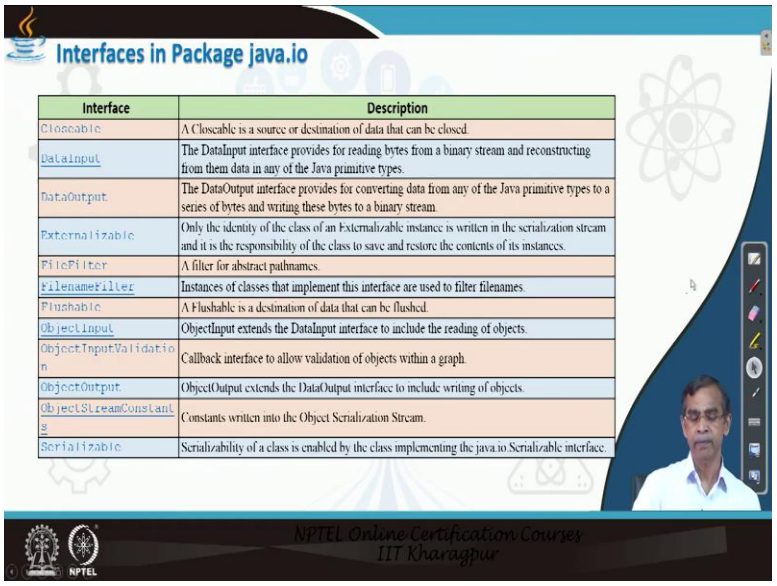
Package java.io

- This package provides system input and output through data streams, serialization and the file system.
  1. Java I/O (Input and Output) is used to process the input and produce the output.
  2. Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.
  3. We can perform file handling in Java by java.io API.

Anyway, so let us first limit to our what exactly java dot io package it is extensive as well as it is basically all square one package which you can fulfil whatever the needs you have. It is possible there. Now, there are many interfaces for this purpose has been pro planned in this package. First let us see what are the different interfaces are there.

I will just simply list all the interfaces. Details of the interfaces will be discussed while will discuss them individually in the context of the different type of stream that will cover.

(Refer Slide Time: 18:07)



Interfaces in Package java.io

Interface	Description
Closeable	A Closeable is a source or destination of data that can be closed.
DataInput	The DataInput interface provides for reading bytes from a binary stream and reconstructing from them data in any of the Java primitive types.
DataOutput	The DataOutput interface provides for converting data from any of the Java primitive types to a series of bytes and writing these bytes to a binary stream.
Externalizable	Only the identity of the class of an Externalizable instance is written in the serialization stream and it is the responsibility of the class to save and restore the contents of its instances.
FileFilter	A filter for abstract pathnames.
FilenameFilter	Instances of classes that implement this interface are used to filter filenames.
Flushable	A Flushable is a destination of data that can be flushed.
ObjectInput	ObjectInput extends the DataInput interface to include the reading of objects.
ObjectInputValidation	Callback interface to allow validation of objects within a graph.
ObjectOutput	ObjectOutput extends the DataOutput interface to include writing of objects.
ObjectStreamConstants	Constants written into the Object Serialization Stream.
Serializable	Serializability of a class is enabled by the class implementing the java.io.Serializable interface.

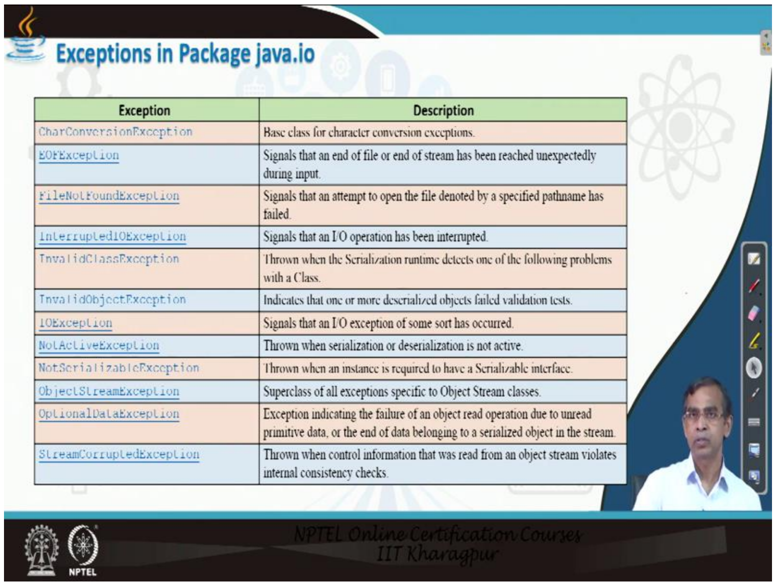
NPTEL Online Certification Courses
IIT Kharagpur

So, these are the interfaces which are defined here in java dot io package. As I have listed here like closable data input, data output, externalizable then file filter these are the different name actually. So, these interface as you know, the interface is basically has a design plan and all those interface defines the methods, their signatures and whatever it is.

But body is not defined there, so only declaration of all methods are listed there in this interface, in some interface, some fills or some constants are defined. Actually, all these interfaces are implemented by some classes, which we will discuss later on. So, these interfaces are basically targeting the different purpose, different functionalities that a programmer may require to manipulate their data those interfaces again can be categorized as a input related, output related.

Here we have listed all input output interfaces are there as from the table, you can see there are many interfaces. Now, while we are handling the different sources of input as well as output, you may face many errors or exceptions. So, to deal all these error and exceptions so that your program can be made more robust or reliable. There are many exception classes also can be planned.

(Refer Slide Time: 19:45)



Exceptions in Package java.io

Exception	Description
CharConversionException	Base class for character conversion exceptions.
EOFException	Signals that an end of file or end of stream has been reached unexpectedly during input.
FileNotFoundException	Signals that an attempt to open the file denoted by a specified pathname has failed.
InterruptedIOException	Signals that an I/O operation has been interrupted.
InvalidClassException	Thrown when the Serialization runtime detects one of the following problems with a Class.
InvalidObjectException	Indicates that one or more deserialized objects failed validation tests.
IOException	Signals that an I/O exception of some sort has occurred.
NotActiveException	Thrown when serialization or deserialization is not active.
NotSerializableException	Thrown when an instance is required to have a Serializable interface.
ObjectStreamException	Superclass of all exceptions specific to Object Stream classes.
OptionalDataException	Exception indicating the failure of an object read operation due to unread primitive data, or the end of data belonging to a serialized object in the stream.
StreamCorruptedException	Thrown when control information that was read from an object stream violates internal consistency checks.

NPTEL Online Certification Courses
IIT Kharagpur

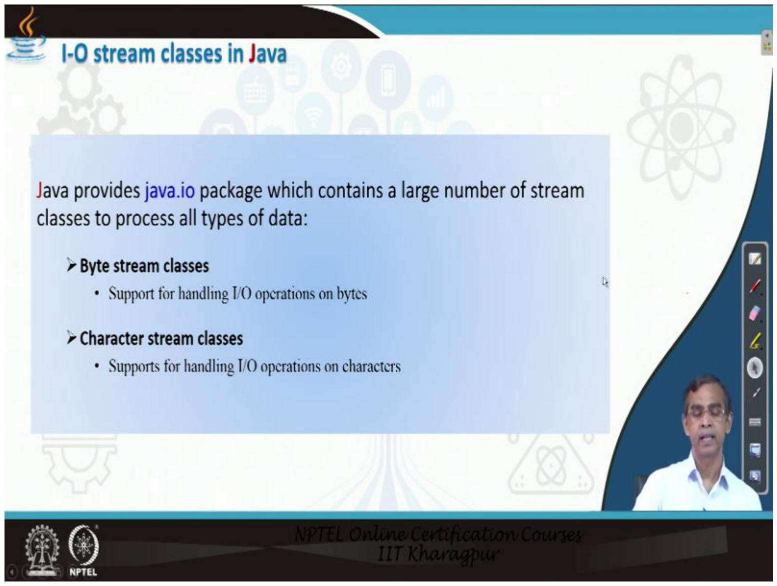
They are basically under the class called io exception. So, whenever you read the document, then you will say that this method will throw this exception, file not found exception, all these things. So, name of the exception given in such a way that from the name, you can understand that why this exception occurs and in which point it occurs those are the exception handle mechanism you can apply so that you can pinpoint the exception and you can wait and you can address those exception so that your program can be more reliable.

So, in this table I have listed those are the important exception it occurs, the name is a self-explanatory. For example, one exception, say EOA exception, it basically indicate that it related to the file whenever you want to read something where the file reaches only end of file. So, end of file exception, file not found exception and many more exception techniques are there. So, all exceptions are basically is an child children or subclass of the superclass. The super class is io exception.

So, io exception covers all the exceptions so that way we usually write while we write the program related to io handling only we just in order to try catch. We just simply try to catch io exception it catch basically all exception and accordingly we can check it. So, these are the exception basically to make the program reliable. And this is important because you are dealing with different source and targets of input and output.

Therefore, that error may occurs any time, so better to deal the errors and that is why this exception classes are defined there. This is these are the classes defined in that java dot io package. Now, let us come to a discussion about the Java classes for input output streams.

(Refer Slide Time: 22:01)



I-O stream classes in Java

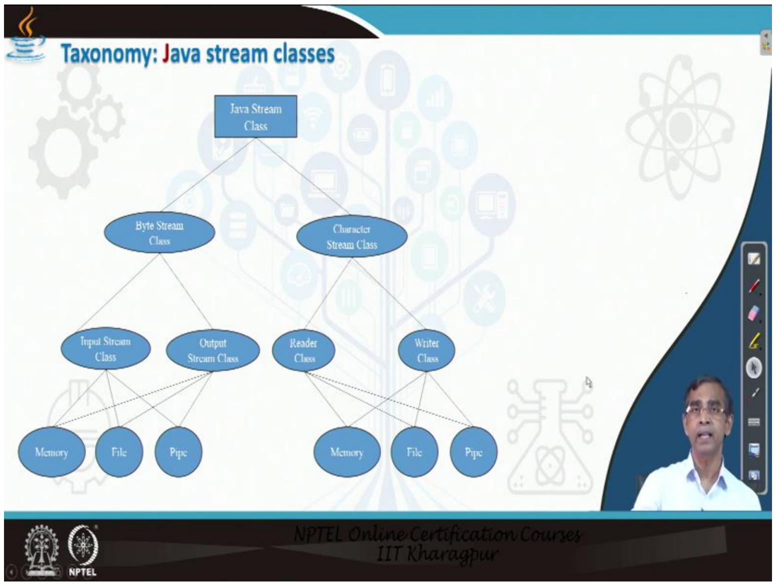
Java provides java.io package which contains a large number of stream classes to process all types of data:

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

Now, as I told you the streams weather input or output, whatever it is, so we can just simply consider either byte stream or character stream. So, byte stream means, the input flow (())(22:18) from the source to the program is in the form of raw data, raw bytes.

This is the byte stream input, similarly output also from your program. You can flow the data in the form of a bytes so it is a byte stream. On the other hand, character stream means it is a Unicode, it is in the form of a text we can say. The text can consist of any symbols that also can be any language Hindi or Chinese or French, whatever it is there possible. So, this basically is a text. So, these are the different streams are there and accordingly, the different classes are there. Now java to handle, so first things they proposed very good class hierarchy.

(Refer Slide Time: 23:10)



Taxonomy: Java stream classes

So, here basically, we can say all classes related to java io. We can say the java stream classes okay can be broadly categorized into two different classes, or different categories. They are called byte stream. If it deals with raw bytes like and then character stream classes, so basically deals with Unicode type like. Now, byte stream again of two type, they are input stream. It is more specifically call input byte stream classes.

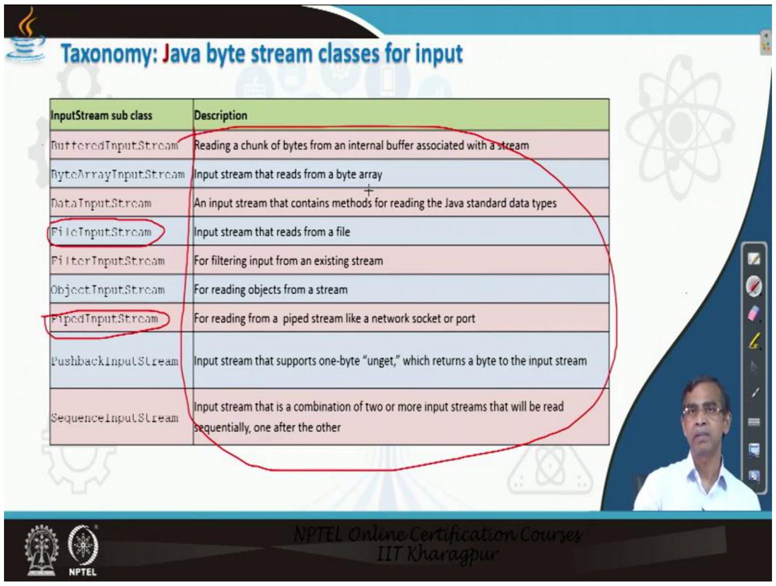
And that output byte stream classes that is called the output stream classes. So, input byte stream output byte stream they are input stream, output stream. And then those are basically may related to the different devices the different what is called the classes are there that is okay I will give you the summary of the different classes related to the different type of source that we can think about both for input stream as well as for output stream.

Now, again from the character stream classes, there are again two type reader class and writer class. Reader class is very similar to the input stream that is related to the byte stream and writer class is just like output stream related to the byte stream. So, this reader class is for input streaming and writer class is for output streaming. Here reader class for character inputting and the output writer class is for character outputting. And then input output it is just like in case of byte stream also, there are many targets or source.

The source may be hardware, any type of hardware, pipe, pipe is basically a flow that is basically related to our threading mechanism where pipe come into the picture. If you are familiar to the Unix program, Unix operating system, possibly you know the concept of pipe is there. So, it basically the source from one source, it will go to another destination. And this is through it is called the pipe. So, there are different types of input and output, source and destination we can say dealing with these things in the form of a character, character stream classes are there.

So, these are the different class hierarchy in a very (())(25:38) point of view I can say, it is very broad level classification I can say. Now let us have a little bit detail classification but detailed classification better we can start with input output byte stream classes, what are the different classes are there.

(Refer Slide Time: 26:00)



Taxonomy: Java byte stream classes for input

InputStream sub class	Description
BufferedInputStream	Reading a chunk of bytes from an internal buffer associated with a stream
ByteArrayInputStream	Input stream that reads from a byte array
DataInputStream	An input stream that contains methods for reading the Java standard data types
FileInputStream	Input stream that reads from a file
FilterInputStream	For filtering input from an existing stream
ObjectInputStream	For reading objects from a stream
PipedInputStream	For reading from a piped stream like a network socket or port
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

NPTEL Online Certification Courses
IIT Kharagpur

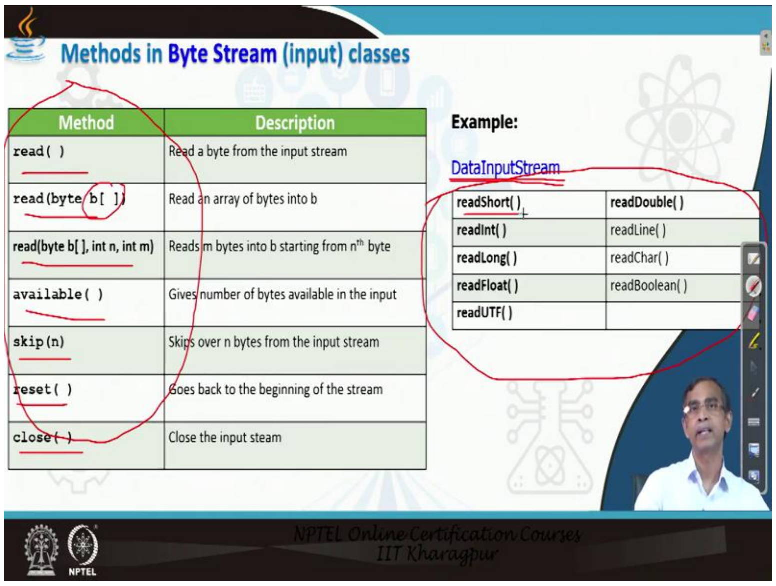
As I told you, it has two types input stream, rather we can say input byte stream. So, this table list whatever the classes are there for input byte stream related things are there. Now, all these classes, as I told you, it is basically interface between your program and the sources that is possible for input, and that is to in the form of bytes.

So, input bytes streams and here in this list I have listed many type of sources like say, buffer input stream, byte input stream, data inputs stream, then file input stream. Now file input stream, as the name implies, you can say here the source of data is file and we want to read data from a file in the form of a bytes.

Likewise, pipe input stream as you can say, it is basically we want to read from a pipe as a byte. And as an input, so there are many other things are there. So, we will try to will learn an actually so different byte stream classes related to inputs and with elastration better. So, that we can understand all those things and utility of there. And I have given a brief what is called the text about each classes. So, I should suggest to you that you should manage your time to check all these meanings are there.

At this moment you may not understand what exactly the things it is doing, but while I will use them and then write and illustrate them using a program, you will be able to understand many things very nicely. And further I have given enough effort to have a good documentation, giving lot of things in more details. Those link also I will give you so that you can follow this link and then you can follow the materials there that will makes your learning complete. I am sure that okay it will help you. So, these are the input stream classes like input stream classes, there are output stream classes.

(Refer Slide Time: 28:23)



Methods in Byte Stream (input) classes

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

Example:

DataInputStream

readShort( )	readDouble( )
readInt( )	readLine( )
readLong( )	readChar( )
readFloat( )	readBoolean( )
readUTF( )	

NPTEL Online Certification Courses
IIT Kharagpur

So, this table, okay I will come to the output stream classes later on. Now here, actually the input stream classes basically for reading and many methods are defined in all those classes. For example, file input stream classes. Now, all methods are basically of same, what is called the name but working of this method is different, which has been taken care in the corresponding class implementation.

So, as a programmer, you should not worried about how all these things are implemented or defining the program. You just simply use this method to solve your program, which I will again give the enough illustration and then demonstrate programming writing. So, the different methods, which are basically define all the classes that we have listed in the previous slides the table the methods are here. So, it is called the read method as you know, it is basically read as a byte. So, here read, write but stored in the form of a byte array write.

So, here read byte but within the set of offset like, available basically to indicate that how many bytes are still available to read from the input stream. These are skip means you can jump. Instead of sequentially reading, you can jump. Reset means you can whatever you are currently reading, you can reset into the starting point also. And close means this stream needs to be close always whenever you open a stream there is a necessity that you should close the stream.

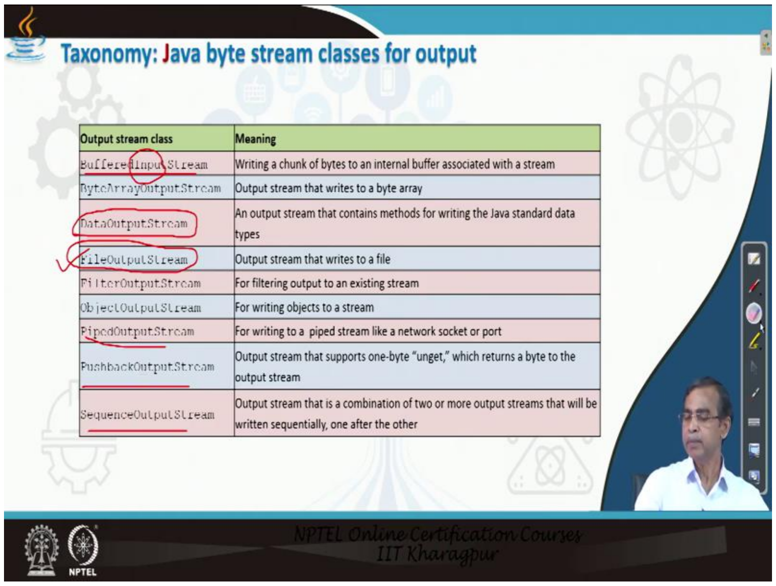
So, these are the different methods define all the classes which we have listed earlier in the table and all the methods are different purpose or different way reading actually are depending on the different targets actually, but from the programing point of view you do not have to bother about how the reading taking place either it from a filter inputs stream or from a pipe input stream or from a file input stream.

So, you just simply read method weather is file or it is printer irrespective of this thing, you will be able to (())(30:29) you will just simply create the stream from your program to or you can say create a line rather from your program to input stream and then reading in the form of a byte that is all. Now, as an example, here I can say this is a particular example data input stream is a another class which belong to the input stream class, actually.

And data input stream will allow you to read data the way the primitive data can be, primitive data means short, long, int and everything. So, in addition to these all methods, these data input stream also provide some method which are defined here. So, those things also we will be able to learn about it. So, it basically it is read short, but read short in the form of binary actually or bytes actually.

But to you, whenever this read short will written a written as a short integer form actually. So anyway, so these are the different methods are defined in different classes. And while we will demonstrate all those using a program, you will be able to follow much more then. But here I am just okay in the process of giving the summary of this package that it supports you. So, this is about the input stream package. Now, likewise input stream classes there is a output stream, different classes are there.

(Refer Slide Time: 31:52)



Taxonomy: Java byte stream classes for output

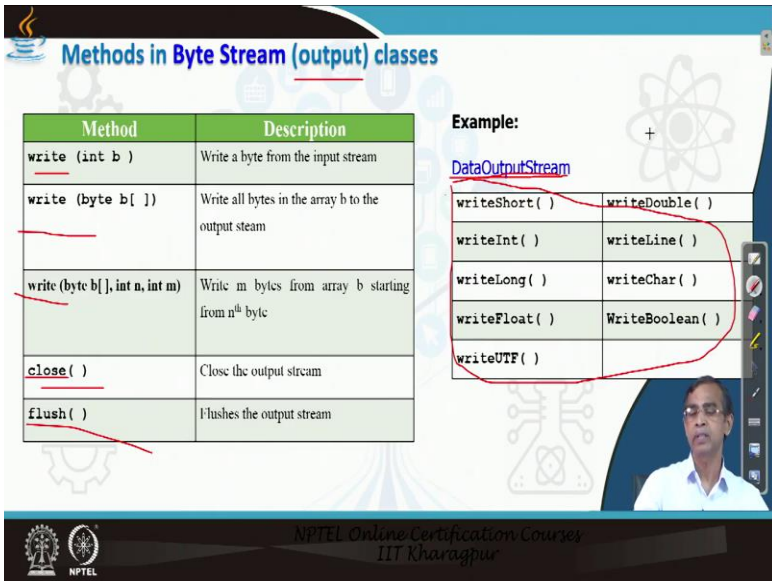
Output stream class	Meaning
BufferedInputStream	Writing a chunk of bytes to an internal buffer associated with a stream
ByteArrayOutputStream	Output stream that writes to a byte array
DataOutputStream	An output stream that contains methods for writing the Java standard data types
FileOutputStream	Output stream that writes to a file
FilterOutputStream	For filtering output to an existing stream
ObjectOutputStream	For writing objects to a stream
PipedOutputStream	For writing to a piped stream like a network socket or port
PushbackOutputStream	Output stream that supports one-byte "unget," which returns a byte to the output stream
SequenceOutputStream	Output stream that is a combination of two or more output streams that will be written sequentially, one after the other

NPTEL Online Certification Courses
IIT Kharagpur

Now here for an example file output stream, you can understand what it basically so that mean, destination is basically file, stream is output stream and from your program, if you want to write something into a file as a byte, you can follow this kind of class for your purpose. So, this will allow you to create a channel from your program to a file. Likewise, there are many. So, this is from the standard input standard output device like console or something like and then the buffer input is a, buffer output rather it is not input it is a output byte output push back.

And there are many other types of output, what is called the destination or output streams we can say it is available there.

(Refer Slide Time: 32:49)



Methods in Byte Stream (output) classes

Method	Description
`write (int b )`	Write a byte from the input stream
`write (byte b[ ])`	Write all bytes in the array b to the output steam
write (byte b[ ], int n, int m)	Write m bytes from array b starting from n^{th} byte
`close( )`	Close the output stream
`flush( )`	Flushes the output stream

Example:

DataOutputStream

`writeShort( )`	`writeDouble( )`
`writeInt( )`	`writeLine( )`
`writeLong( )`	`writeChar( )`
`writeFloat( )`	`WriteBoolean( )`
`writeUTF( )`	

NPTEL Online Certification Courses
IIT Kharagpur

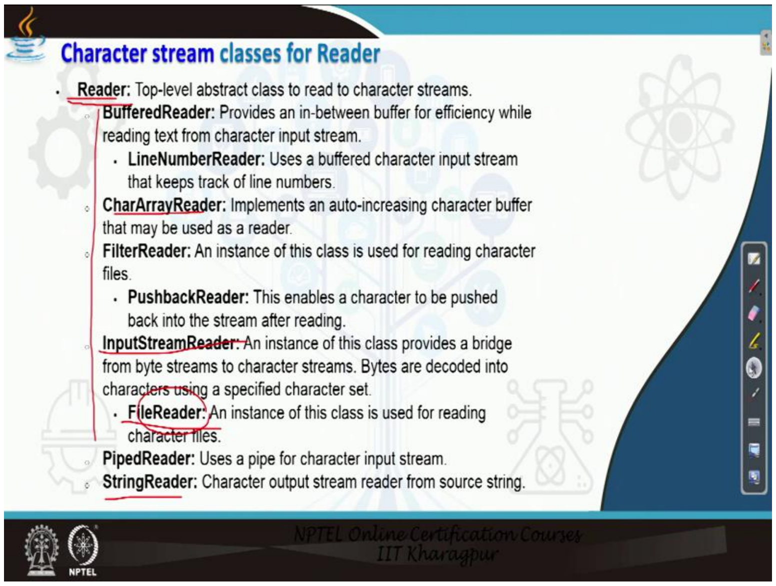
And like the input stream, there are many methods also defined and in the next slide you can see what are the different methods are there. So, earlier methods are related to reading, but this is basically output. So, it is related to writing. So, it is basically write, these are different write versions are there, close and flush, flush means whenever you encounter something, you want to clean the buffer or flush from your program to the destination output, actually so this one.

And close is basically closing a stream just like you have to close, like input stream to be close output stream if you open, it should be close like this one. Now, so these are the basically byte stream. Now there are some specialized method also available in some specialized classes. For example, data output stream, which is very more convenient. And here actually, if you want to write something of your own in a primitive form that means if you want to write an integer, write an character or string, then these are the method that you can consider, okay?

So, these basically allow you to write your primitive data, but ultimately all this primitive data will be converted into byte form and that will stored. But that will take care by the data output stream class and the method corresponding to that one. So, this write short basically take the short integer from your program and ultimately, it will write into the output say 5 in the form of a byte that is all.

And write UTF for example it will write in the text, stream format, like and java treat everything as a stream like actually, any way that things will be discussed while we will discuss about character stream later on, that okay? So, this basically the idea about handling the different output stream classes related to the byte stream like byte stream there is another class also there, it is called the character stream.

(Refer Slide Time: 34:46)



Character stream classes for Reader

- **Reader:** Top-level abstract class to read to character streams.
  - **BufferedReader:** Provides an in-between buffer for efficiency while reading text from character input stream.
    - **LineNumberReader:** Uses a buffered character input stream that keeps track of line numbers.
  - **CharArrayReader:** Implements an auto-increasing character buffer that may be used as a reader.
  - **FilterReader:** An instance of this class is used for reading character files.
    - **PushbackReader:** This enables a character to be pushed back into the stream after reading.
  - **InputStreamReader:** An instance of this class provides a bridge from byte streams to character streams. Bytes are decoded into characters using a specified character set.
    - **FileReader:** An instance of this class is used for reading character files.
  - **PipedReader:** Uses a pipe for character input stream.
  - **StringReader:** Character output stream reader from source string.

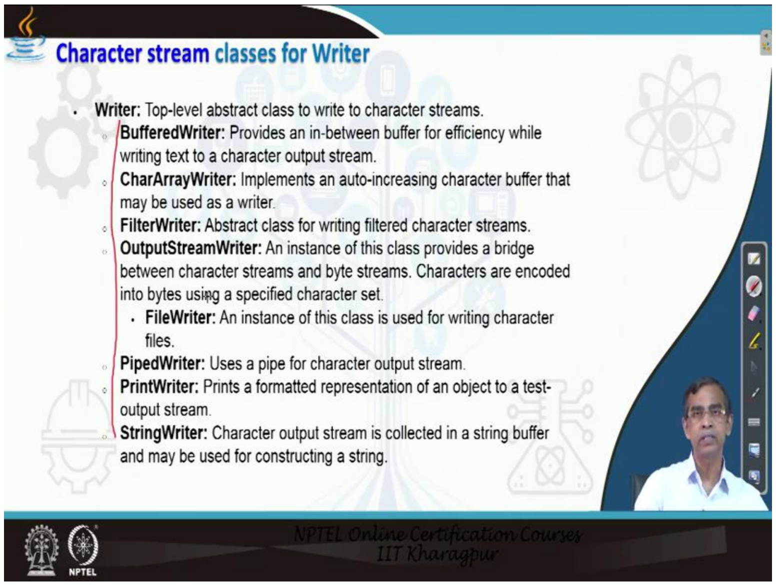
NPTEL Online Certification Courses
IIT Kharagpur

Let us see what are the different again classes are there, so for character stream is concern, here I listed the many classes as I told you, there this is just like input stream it is a reader basically input stream, but input flow of data in the form of a stream of characters I can say and reader take care about it.

There are different classes like in the byte stream here, also buffer reader, then character reader, then inputs stream reader, file reader, string reader. And you know, the file reader means it basically read reading from a file and reading from a file as a text or we can say as a character. So, there is a flow of data, is a character wise. So, in case of byte stream, maybe the flow of data in the of byte, one byte is like. Here the flow of data in the form of a Unicode symbols, which is 16 bits.

So, two bytes at the time like, so it is flow of bytes it is there. So, this is the reader for the input purpose.

(Refer Slide Time: 36:00)



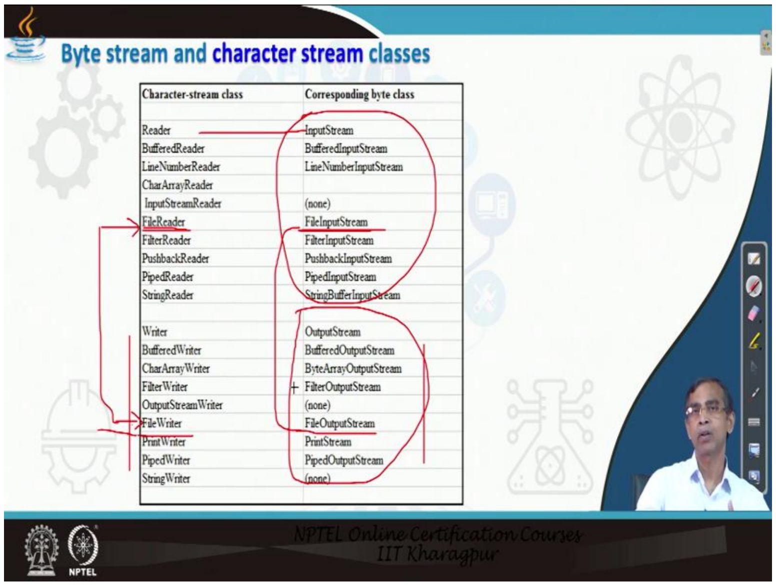
Character stream classes for Writer

- **Writer:** Top-level abstract class to write to character streams.
  - **BufferedWriter:** Provides an in-between buffer for efficiency while writing text to a character output stream.
  - **CharArrayWriter:** Implements an auto-increasing character buffer that may be used as a writer.
  - **FilterWriter:** Abstract class for writing filtered character streams.
  - **OutputStreamWriter:** An instance of this class provides a bridge between character streams and byte streams. Characters are encoded into bytes using a specified character set.
    - **FileWriter:** An instance of this class is used for writing character files.
  - **PipedWriter:** Uses a pipe for character output stream.
  - **PrintWriter:** Prints a formatted representation of an object to a test-output stream.
  - **StringWriter:** Character output stream is collected in a string buffer and may be used for constructing a string.

The likewise, there for output purpose also one different set of classes are there, which I have listed here. These are the write, writer related, so it is basically outputting, again from the character flow and so the different number of classes are there in java dot io package by which you can flow the data from your program, as a character to the output destinations.

So, these are the different types different classes are there belongs to the two different categories, like byte stream and character stream categories.

(Refer Slide Time: 36:35)



Byte stream and character stream classes

Character-stream class	Corresponding byte class
Reader	InputStream
BufferedReader	BufferedInputStream
LineNumberReader	LineNumberInputStream
CharArrayReader	
InputStreamReader	(none)
FileReader	FileInputStream
FilterReader	FilterInputStream
PushbackReader	PushbackInputStream
PipedReader	PipedInputStream
StringReader	StringBufferInputStream
Writer	OutputStream
BufferedWriter	BufferedOutputStream
CharArrayWriter	ByteArrayOutputStream
FilterWriter	FilterOutputStream
OutputStreamWriter	(none)
FileWriter	FileOutputStream
PrintWriter	PrintStream
PipedWriter	PipedOutputStream
StringWriter	(none)

And there is an analogy, actually, because the two stream are to serve same purpose, but different ways. So, byte stream is basically flow of data in the form of a byte. Whereas, character stream is in the flow of data as a characters, so there is analogy.

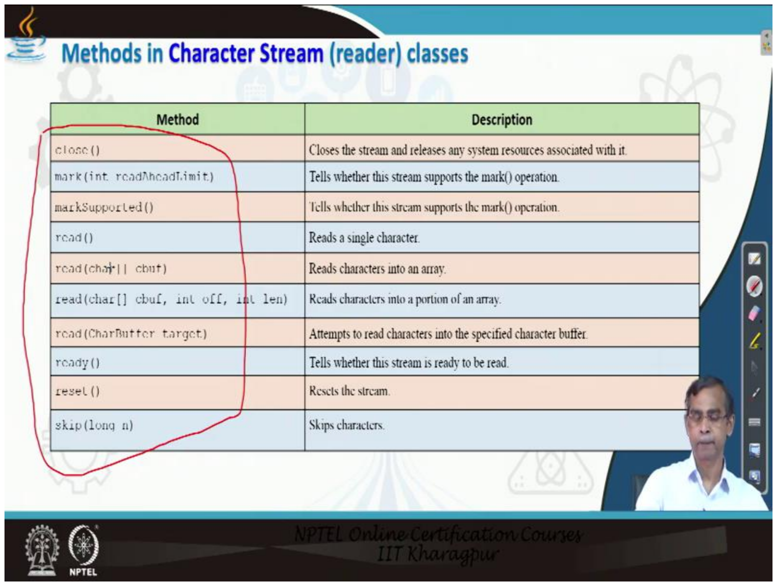
So, the reader, which is there in character stream, is basically input stream in the byte stream. For example, here so here, here is a file reader, which is in the character stream. This is file input stream basically in the byte stream like likewise in (())(37:16) also it is there. And another thing is that for this input stream and for output stream the (())(37:23) will read from an input, for example, the file and if we want to write into the same file, then how it can be done or from standard input or standard output, how it can be done.

So, there is a basically, for example, here if the file input stream is there, you can see file output stream is there. So, these two are basically complement each other like it is for input and one is for output. Here is also in case of character stream, their name is like file writer and here is a file reader. So, if it read a character from a file and these basically read character from a read, write something as a character into a file like. So, these also is a corresponding things it is there.

So, there are actually learning a particular type of input stream. Also help you to learn other category for example byte stream. If you learn successfully, you will be able to learn character stream also automatically because same concepts, same things are followed. Internal mechanism is different, internal mechanism absolutely up to the Java developer, how they have implemented. But from a programmer point of view, it is really is a boom, that is a basically is a lot of facilities that you can handle.

You can have to deal with any type of sources, any type of destinations for your data is concerned. So, this is a very great advantage having this programming language and you can have many things are there.

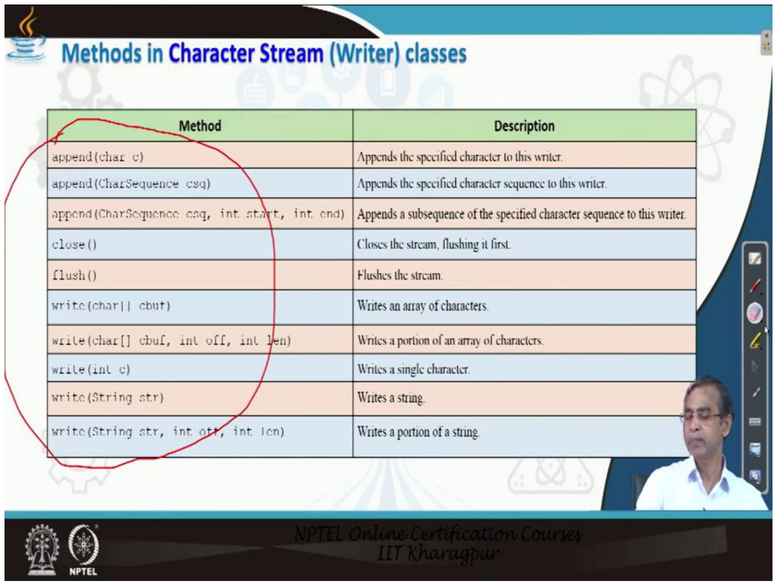
(Refer Slide Time: 38:57)



Methods in Character Stream (reader) classes

Method	Description
close()	Closes the stream and releases any system resources associated with it.
mark(int readAheadLimit)	Tells whether this stream supports the mark() operation.
markSupported()	Tells whether this stream supports the mark() operation.
read()	Reads a single character.
read(char[] cbuf)	Reads characters into an array.
read(char[] cbuf, int off, int len)	Reads characters into a portion of an array.
read(CharBuffer target)	Attempts to read characters into the specified character buffer.
ready()	Tells whether this stream is ready to be read.
reset()	Resets the stream.
skip(long n)	Skips characters.

NPTEL Online Certification Courses
IIT Kharagpur



Methods in Character Stream (Writer) classes

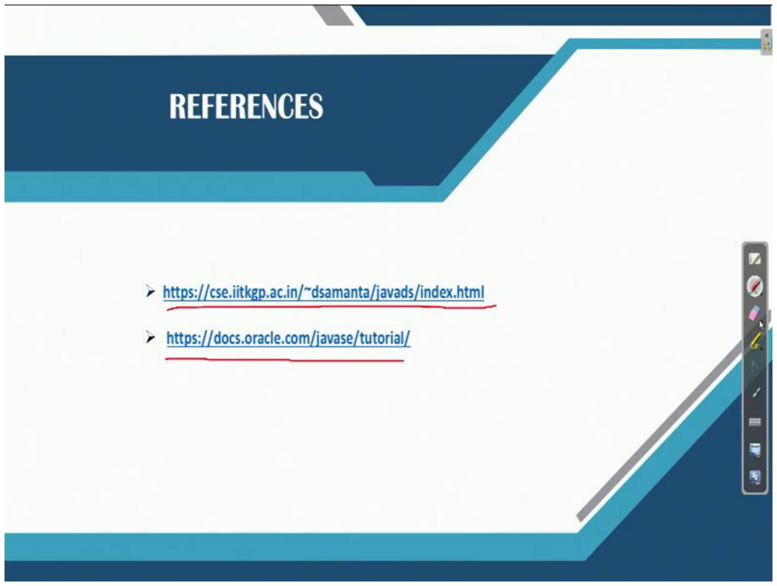
Method	Description
append(char c)	Appends the specified character to this writer.
append(CharSequence csq)	Appends the specified character sequence to this writer.
append(CharSequence csq, int start, int end)	Appends a subsequence of the specified character sequence to this writer.
close()	Closes the stream, flushing it first.
flush()	Flushes the stream.
write(char[] cbuf)	Writes an array of characters.
write(char[] cbuf, int off, int len)	Writes a portion of an array of characters.
write(int c)	Writes a single character.
write(String str)	Writes a string.
write(String str, int off, int len)	Writes a portion of a string.

NPTEL Online Certification Courses
IIT Kharagpur

Now, here the different methods, those are there belong to the different classes and character stream classes reader class. So, these are the different method, reader class is basically it will allow you to read related operations as a character stream into the different from the different sources of data.

So, these are the methods that declares there likewise there is a, there are some methods which are declared there, as a writer class. It is basically the methods related to writer I am just listing all these method. And finally, we will get will take some time where we will discuss all these methods, then utility and then application writing program. Then only many methods will be very clear to you so that you can understand, this is working like this way and that is the things are there.

(Refer Slide Time: 39:53)



REFERENCES

- https://cse.iitkgp.ac.in/~dsamanta/javads/index.html
- https://docs.oracle.com/javase/tutorial/

So, this is about just the introduction about java dot io package, I have tried to give the summary of this package actually. And half an hour discussion you can understand how much I can give it, it is there but if you want to have many more things, I should suggest you to check this link. This is a good link of course, you can check it where exhaustive material you can find it. And this is a link where I have write, manage all the discussion in the form of an HTML page.

So, that you can find, you can browse the page and then you can get many information, all programs and everything you can get it from here also, this is very important link for you. Okay fine we will discuss about programming with this java dot io shortly. And fine you just go through the things little bit and then when I will come to the programming you will be able to understand much more things in details. So, that will be followed from the next lecture onwards. Thank you very much.