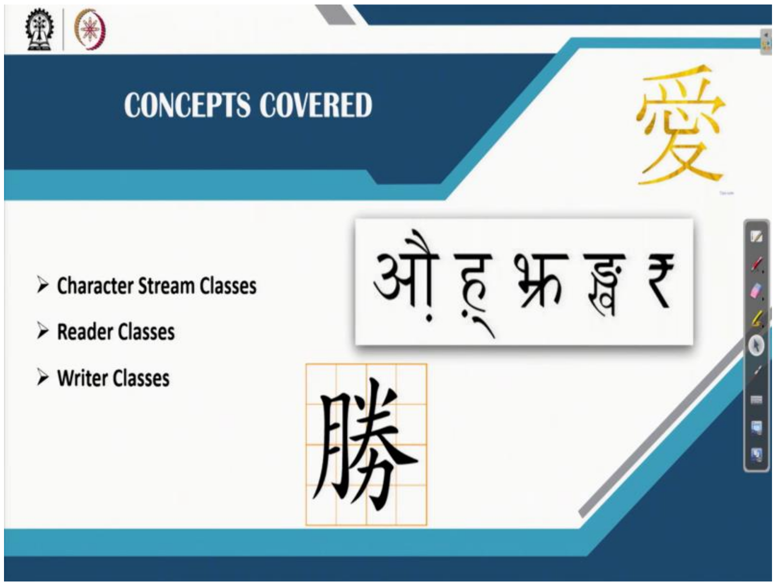
Data Structures and Algorithms using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 44
IO with Character Streams

So, let us have a quick tour to another streams, it is character streams. So, today our topic is input output with character streams which is there is Java.IO package.

(Refer Slide Time: 0:41)



CONCEPTS COVERED

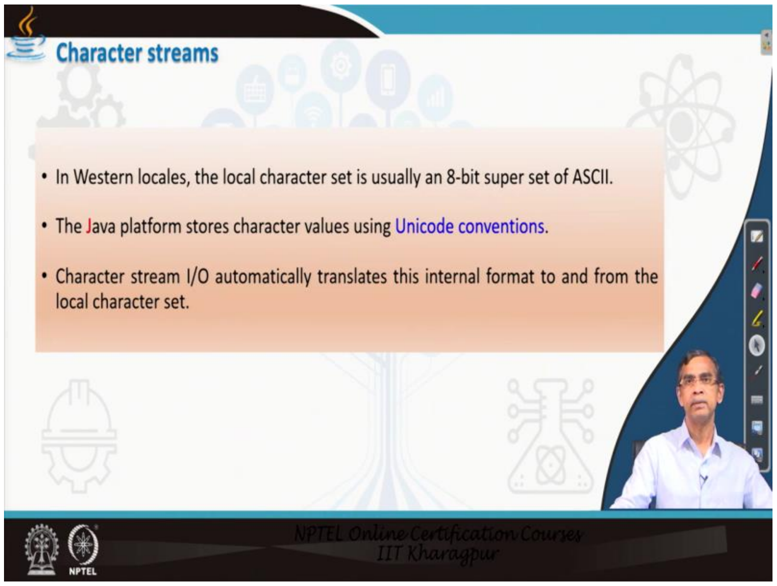
- Character Stream Classes
- Reader Classes
- Writer Classes

So, in today's discussion, will discuss about character stream classes and there are two different types of sub classes basically; reader and writer. So, reader is mainly for reading characters from an input source and writer classes are the classes basically helps the programmer or to write some data from the program to a target source or destination output actually. Now, unlike write stream, it basically read as a character and you know in case of Java a character is basically represented by a Unicode format.

And Unicode can represents the any what is called the character that is displayable into some locale, for example here you can display some information in in Hindi or in Japan, in Chinese language whatever it is there. So, there are many different locale that it can be support. It is a basically you can specify that which Unicode character you want to display. Anyways so different forms are basically defined as a Unicode characteristics actually.

So, there are huge number of characteristics and support almost all language those are recognised worldwide. Anyway, so these basically read as a character as you can see from this figure you can read Sanskrit or English or French whatever the character actually, anyway because it is read as a Unicode.

(Refer Slide Time: 2:22)



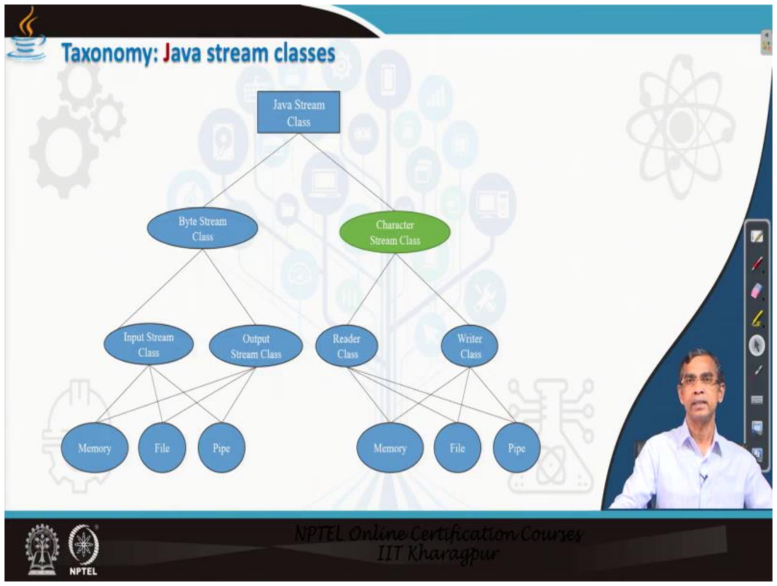
Character streams

- In Western locales, the local character set is usually an 8-bit super set of ASCII.
- The Java platform stores character values using Unicode conventions.
- Character stream I/O automatically translates this internal format to and from the local character set.

Now let us see the character stream classes. What are the different character stream classes that is there? So, there are convention of old computer. It is called the western style that is called western locale which basically read the character in the form of ASCII value which is actually 8 bit character sets. Now in contrast as I told you Java stores character set as an Unicode which is the 16 bit character set. So, different codes, values are there to represent different phones actually. So, here are you do not have to bother about how internally it can change the format and can paint a particular font corresponding to different character actually.

It is internally Java actual machine will do for you. So, this way it basically gives or enables you to display any locale character set which is very important in any OPS or development or in web documents or writing some OPS like PHP or it is in Java script or whatever it is there. Because we can represent any symbol that is basically corresponding some local set.

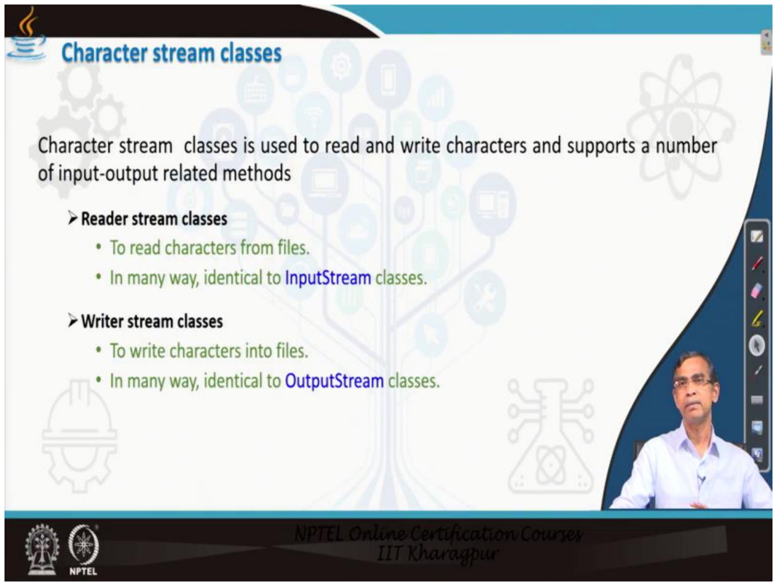
(Refer Slide Time: 3:39)


Taxonomy: Java stream classes

Now, for these character stream classes representation, there are again two type of write, one for input and another for output. Related to input streaming, there is the classes called reader classes and belong to reader classes, again several stream classes are there. Similarly, belongs to writer classes. Writer classes is mainly for outputting purpose that means if you write something from your program to some destination output. Destination output may be console also, may be monitor or it can be a printer, it can be a file, it can be a another pipe. Pipe is basically connected to another thread whatever it is there.

So, this kind of communication if you want to do from your program to any other target output, it basically you can cons write writer classes. So, there are several sub classes for the writer classes are there.

(Refer Slide Time: 4:32)



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.

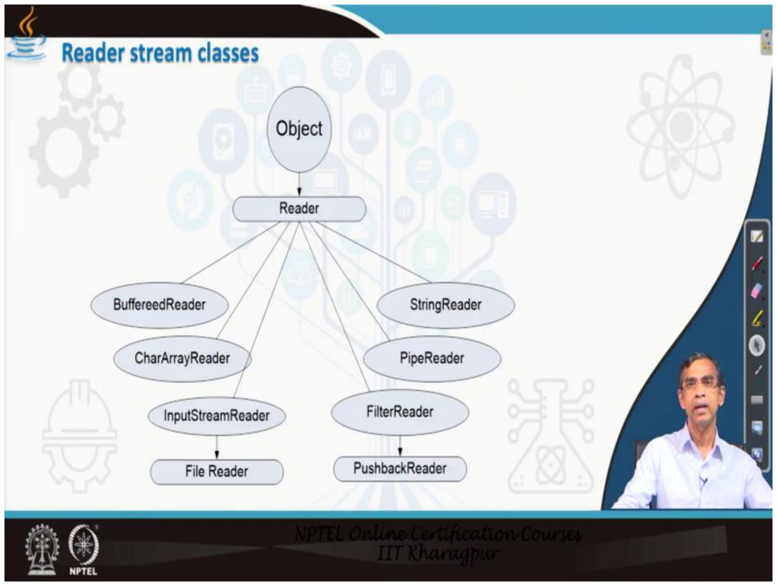
NPTEL Online Certification Courses
IIT Kharagpur
NPTEL

So, these basically you can see, all the reader classes are very similar or resembles you the input stream classes which you have already studied. So, like 5 input stream actually this is one input stream related to reading from file corresponding to this and it is basically if you want to read as a character flow, character as a flow or character as a stream, then that file is called the file reader. Similarly, output stream which you have already studied for the write stream in in much many ways resembles with the reader writer classes also.

And for example, one sub class of output stream class is file output stream and corresponding to this file output stream. File output stream basically read byte byte wise whereas, the character wise if you want to write something, then the file is file writer. So, the way we can create the connection from a program to an input source. Similarly, we can create a connection from a program to output source creating a particular object or particular type that means for the particular class. So, a file reader can be created a connection can be created if you want to write from your program to a file file writer.

Similarly, file reader is there if you want to read something from the file character by character, then you should create a file reader as a class. So, this concept is very similar, the method and everything is very same same actually.

(Refer Slide Time: 6:06)



Reader stream classes

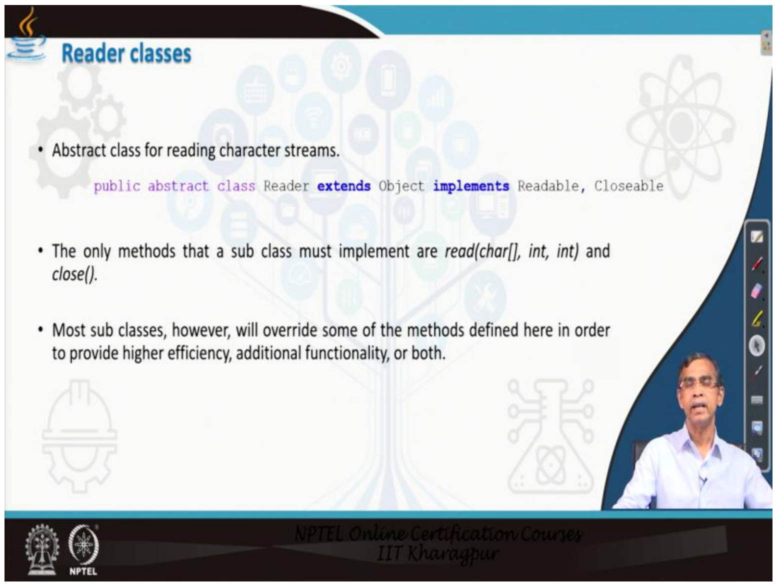
So, let us see how different classes are there. Now so far the reader classes are concern there are many sub classes those are defined in Java dot IO package. These are buffered reader. So, it basically read a buffered data. This is for the faster access. It is read from the character we already familiar to buffered reader as a write stream. So, from an input source maybe keyboard whatever you type it goes to the buffer. Buffer size can be limited which is certain size, you can specify while you create the object and then that buffer will be used to store the data temporarily. And your program will read from the buffer.

So, your input source may be slow compared to your program that you are running because CPU is far much faster than input device like in that case buffered reader will help you to do the working faster and then the character Array reader is basically, if you want to read which is stored in the array of characters. You can see it as a collection like. And then input stream reader it is similar to the input stream that is there, it is basically a bit from input stream as a from byte to a character stream.

So, this is one example that is called the brazing that can help you to byte to character version automatically. File reader is basically reading from a file where this interface is basically connects from your program to your file. There is a string reader, this basically there are many utility programs are there, many methods are there which basically related to string. Basically it will read the string and is particularly very important whenever you want to read something from your local. That means different fonts that you want to feel like.

Pipe reader it is basically reading from a pipe, pipe is basically a thread actually you can say. Filter reader is basically convert the data and filter it into a primitive form so it can convert it and then you can read it. Pushback reader is basically a reading and after reading the data will be send back to the input source again. So, these are different classes those are there. And it is not possible again to discuss all classes. We will try to choose some selective classes but these can be considered as a representative examples. All other classes are in similar nature, the details of this the other classes, you can get it link from the other I give you at the end of these lecture slides.

(Refer Slide Time: 8:36)



Reader classes

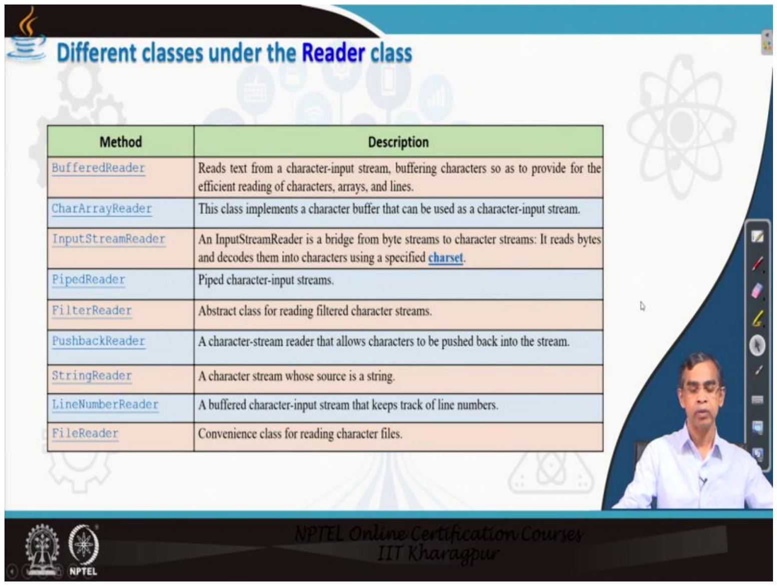
- Abstract class for reading character streams.

```
public abstract class Reader extends Object implements Readable, Closeable
```

- The only methods that a sub class must implement are *read(char[], int, int)* and *close()*.
- Most sub classes, however, will override some of the methods defined here in order to provide higher efficiency, additional functionality, or both.

Now, so reader class, it is very similar to input stream class basically. It basically read character by character. And it has many methods.

(Refer Slide Time: 8:48)



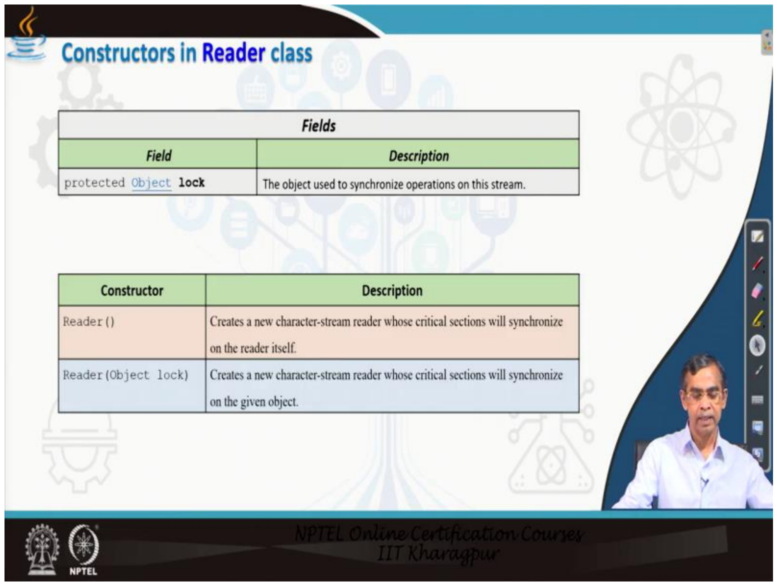
Different classes under the Reader class

Method	Description
BufferedReader	Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.
CharArrayReader	This class implements a character buffer that can be used as a character-input stream.
InputStreamReader	An InputStreamReader is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified charset.
PipedReader	Piped character-input streams.
FilterReader	Abstract class for reading filtered character streams.
PushbackReader	A character-stream reader that allows characters to be pushed back into the stream.
StringReader	A character stream whose source is a string.
LineNumberReader	A buffered character-input stream that keeps track of line numbers.
FileReader	Convenience class for reading character files.

NPTEL Online Certification Courses
IIT Kharagpur

Those are basically useful for reading purpose only reading character by character. These are different class we have already discussed about, it is basically listed in a tabular form. You can check it, but mainly we will discuss about, we will discuss about some particular classes that can be more important to understand about.

(Refer Slide Time: 9:15)



Constructors in Reader class

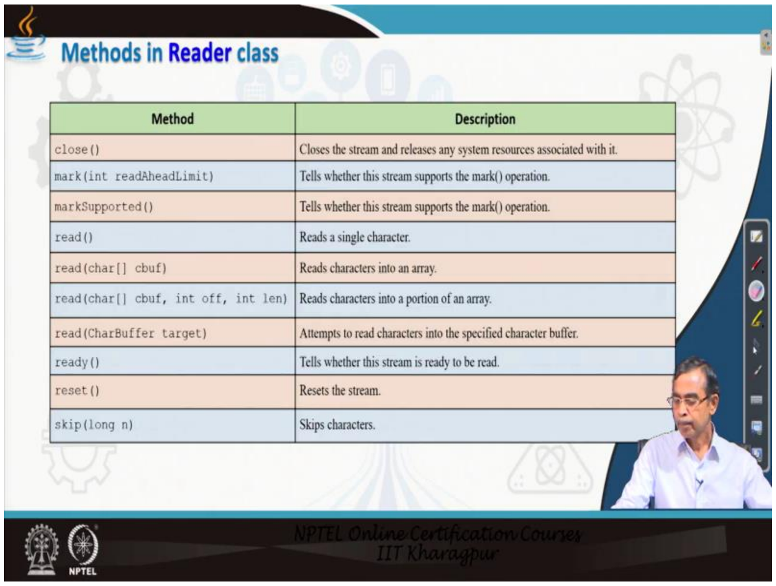
Fields	
Field	**Description**
protected Object **lock**	The object used to synchronize operations on this stream.

Constructor	Description
Reader()	Creates a new character-stream reader whose critical sections will synchronize on the reader itself.
Reader(Object lock)	Creates a new character-stream reader whose critical sections will synchronize on the given object.

NPTEL Online Certification Courses
IIT Kharagpur

Now, so reader class object can be created there is a constructor of course.

(Refer Slide Time: 9:21)



Methods in Reader class

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.

Now here actually this slide is more important. Let us see what we have listed here, few methods. Those methods are very useful to read character from a character stream class as an input. So, close basically is an essential method because I told you that whenever you open any stream, input stream or output stream, at the end of the program, it is the customize to close it so that the input source or output source will be will not be I mean set from the destruction or damage.

And marking is basically you can mark that which point you have you are in reading like but this marking is possible only mark supported. It is very similar to the input stream classes that you have considered. Now, there are the few read method that you can see, how the different way you can read. So, these basically read method is written a character and this is basically read read read something I mean set of character which can be initially stored into the buffer. So, buffer is an array of characters. And you note that array of character string is different so this is an array of characters.

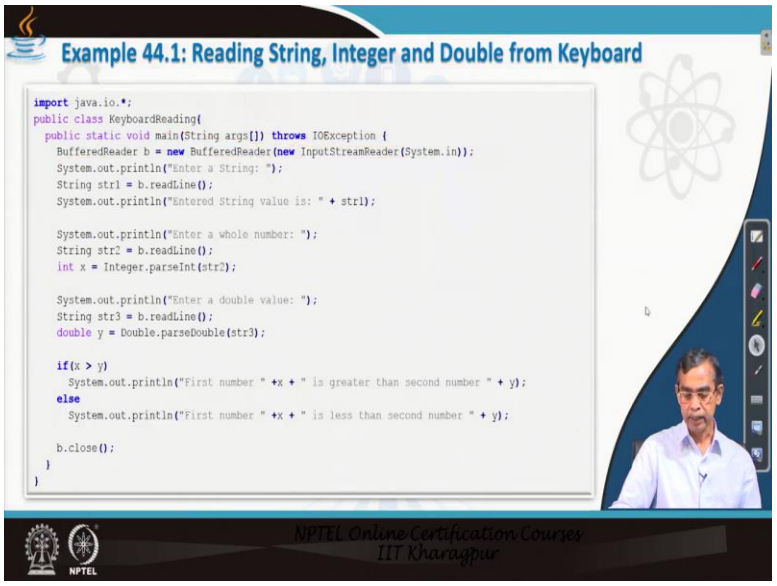
Now, here also reading a characters and storing into a buffer, the name of the buffer is this one. And starting from the off and length, actually we can specify that how much part in the buffer where you want to store the data. So, these are basically reading as a bulk, these two methods and store into a temporary array, array of characters. And you can note all these method are very similar to the method which we have already studied during the discussion of input stream classes but only the difference is that the read method in input stream written you byte whereas it is written as a character.

It written as a array of characters, in case of input bytes stream it was it was read as a array of bytes and like this one. Now, ready is basically to test whether the input source is ready, it is for synchronization purpose actually to make that thing sync so ready method can be called for if the if the input source is busy or not able to produce data, then it basically down falls like. And reset is basically mark you have mark something, if you want to reset into starting 0 like so it reset will be there. And skip basically, you can skip some characters from reading actually. So, this is basically skip method.

And you can note that all this methods basically same which is also available for the input byte stream. So, this means if you learn input, input stream classes you are by the process you learn the character stream classes. Only difference is that whether reading as a byte flow as a stream as a byte or stream as a character. That is that only matters, so internally they their mechanism is different but from the programmer point of view, basically same thing. So, you can read either as a byte or character depending on your choice.

Now, obviously question that arises that which you should consider. There are certain pros and cons. I will come to the discussion at the end of this class. We can discuss about whether byte stream or character stream.

(Refer Slide Time: 12:55)



Example 44.1: Reading String, Integer and Double from Keyboard

```
import java.io.*;
public class KeyboardReading{
  public static void main(String args[]) throws IOException {
    BufferedReader b = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter a String: ");
    String str1 = b.readLine();
    System.out.println("Entered String value is: " + str1);

    System.out.println("Enter a whole number: ");
    String str2 = b.readLine();
    int x = Integer.parseInt(str2);

    System.out.println("Enter a double value: ");
    String str3 = b.readLine();
    double y = Double.parseDouble(str3);

    if(x > y)
      System.out.println("First number " +x + " is greater than second number " + y);
    else
      System.out.println("First number " +x + " is less than second number " + y);

    b.close();
  }
}
```

Now, let us have some illustration, first we will see exactly how we can read from a keyboard. We have already familiar we have already experience on idea about how reading from keyboard using byte where you use the data input stream class. But here we will use the same thing but it is the not data input stream class but we will use one one another class that is basically to read from the keyboard.

We can term it a buffer reader actually for that is suitable that is basically is a counter part of the data input stream like. So, this program basically explain same thing which we have done reading the keyboard using data input stream but here we will read using another class called the buffer reader. Now, this is the main program so using keyboard reading using character stream, now through the IO exception because all the codes should be enclosed within the IO is a try catch block.

So, otherwise this program cannot be compiled actually compile or can ignore to compile this program either you can write through IO exception here or you can write try and then catch block here. So, all these code can be enclosed which is a try catch block. Anyway so first we have to create a connection from your program to an input source. Again, we consider reading from the keyboard our input source will be system dot in that is the keyboard actually. And here buffer reader is basically your reader from which you want to read it.

Now, here input stream reader is basically big because ultimately at the end it is basically byte. So, basically whatever you type from the keyboard it basically goes as a byte by byte. But we want to read as a character so is input stream reader is a bris here. So, basically it is read byte from the keyboard and written this as a character. So, this concept it internally input stream reader will take care for you. So, this basically makes a connection between your program to the keyboard actually.

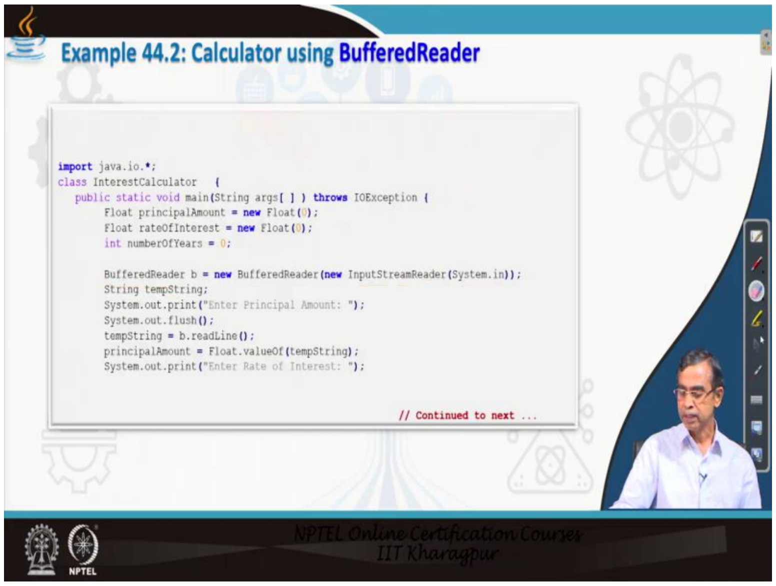
Now, here few things are very similar to the earlier one. Enter a string a promt can be given from your program. So, you read the text from, so it can be read as a string because everything that written is a set of character actually whatever it is basically are the string and then this string needs to be parsed in order to get the right value for you. So, here is basically enter string and then you so you pin the string that you have read it.

Next, you just give a prompt whole number. It read as a string and then pass the string, enter a double value, read the value, parse the value, get the value. So, x and y are the two value that you could read it from the keyboard. And then this is the simple logic that you have compare whether x is greater than the y or not accordingly print whatever it is there. Now, you note everything is very similar up to this part everything is very similar whatever we are already familiar in the previous example while we are discussing input stream.

So, this part is very similar to the previous one. Only the part which is very different that whether you are reading as a character stream or byte stream, here is the code actually. You can recall the previous program where you use the data input stream to read something from the keyboard. So, we create a data input stream object. But while we are reading as a character we read as a buffer reader and for this purpose one (())(16:49) is required, this is right and then you can write it. So, either buffer reader directly system dot in may come may leads to a total error because it will read as a byte by byte.

So, it is required to collate the byte from the character font actually. That is why we need the input stream reader that is the only difference that we can see, otherwise rest of the program remain same actually. So, this this point is important to note actually.

(Refer Slide Time: 17:19)

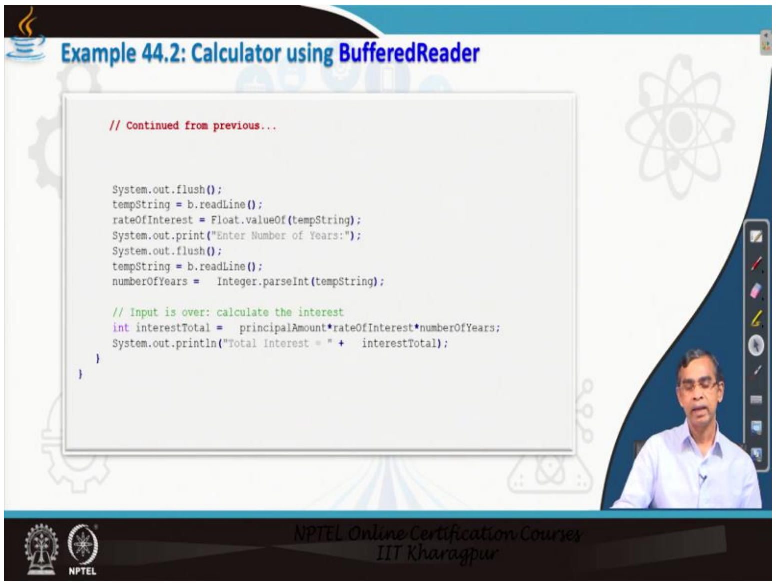


Example 44.2: Calculator using BufferedReader

```
import java.io.*;
class InterestCalculator   {
   public static void main(String args[ ] ) throws IOException {
        Float principalAmount = new Float(0);
        Float rateOfInterest = new Float(0);
        int numberOfYears = 0;

        BufferedReader b = new BufferedReader(new InputStreamReader(System.in));
        String tempString;
        System.out.print("Enter Principal Amount: ");
        System.out.flush();
        tempString = b.readLine();
        principalAmount = Float.valueOf(tempString);
        System.out.print("Enter Rate of Interest: ");

                                                   // Continued to next ...
```

And now let us come to the another program which you have already done using data input stream. But in that case we will do the same thing using the character stream. So, it is a buffer reader again here so input stream reader, this is the keyboard and this is the part of the rest already we are familiar to that basically in the same as the things we have discussed in the last program that we have.

(Refer Slide Time: 17:41)



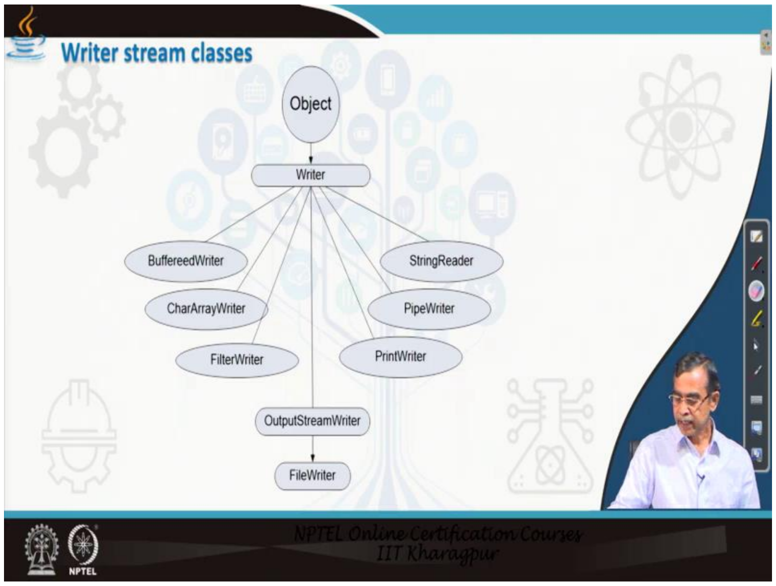
Example 44.2: Calculator using BufferedReader

```
    // Continued from previous...



    System.out.flush();
    tempString = b.readLine();
    rateOfInterest = Float.valueOf(tempString);
    System.out.print("Enter Number of Years:");
    System.out.flush();
    tempString = b.readLine();
    numberOfYears =   Integer.parseInt(tempString);

    // Input is over: calculate the interest
    int interestTotal =   principalAmount*rateOfInterest*numberOfYears;
    System.out.println("Total Interest = " +   interestTotal);
  }
}
```

So it basically read the different values and then process some calculation it is there. Now so reading either from input stream as a byte and reading from a reader write as a character essentially same. From the programming point of view, only the point is that which type of data you want to read that you have to think and then accordingly you can make a connection from your program to the input stream that is matter, otherwise everything remain same.

So, programming is so actually Java developer has made the programming so easy actually this way it is very easy. Now, let us consider few more example to understand the concept. I will mention the example in the same line to draw an analogy between the byte stream and then character stream. That is the I am quoting the same example but ok doing the same thing, it is a using character stream whereas in the last video we have done in the byte stream actually.

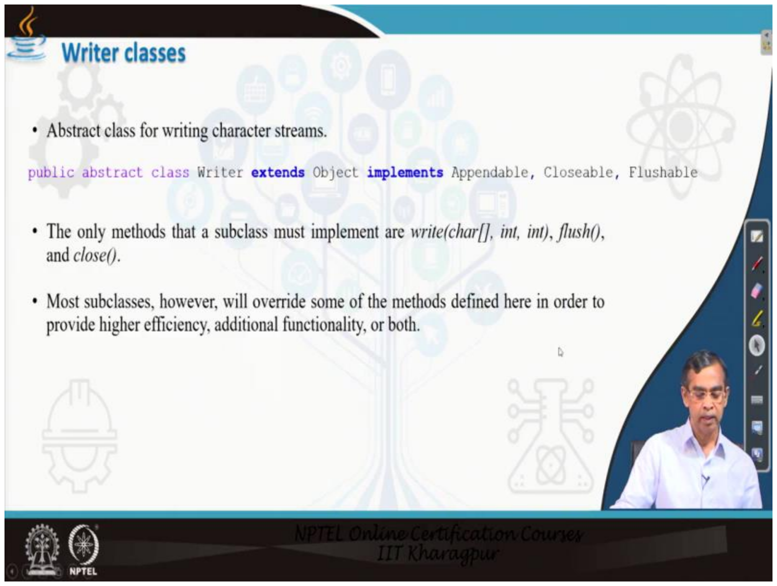
(Refer Slide Time: 18:45)



Now, let us consider another example. So, this is about write writing some data from you program to output, output source output what is called the destination. Now, for this purpose there are many classes are known. These are the sub classes of the writer class which is an extra class. So, is a buffered (read) writer, character array writer. If you want to write into an array that is the correction you can say. Filter writer, write in a filter form. String writer, it is a writer actually. Then pipe writer, print writer, print writer is basically printing something into may this is basically printing writing the standard what is called the output.

Standard output means it is a consoler monitor. You can use it also to send the data to your file also using. Then output stream writer is basically just opposite to the input stream writer is basically bridge between the byte to character actually. And is a file writer if you want to write something into file directly then you can use this one. So, these are the different classes are there so for writing data into an output stream.

(Refer Slide Time: 20:14)



Writer classes

- Abstract class for writing character streams.

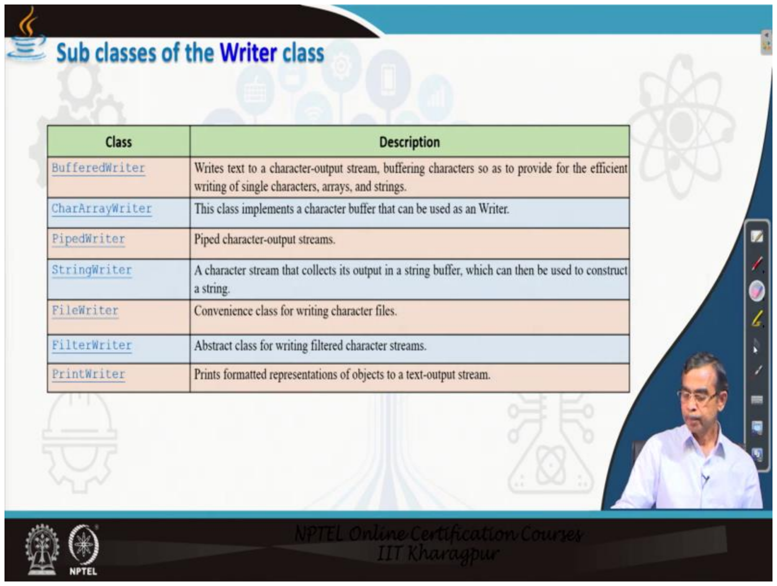
```
public abstract class Writer extends Object implements Appendable, Closeable, Flushable
```

- The only methods that a subclass must implement are *write(char[], int, int)*, *flush()*, and *close()*.
- Most subclasses, however, will override some of the methods defined here in order to provide higher efficiency, additional functionality, or both.

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us consider few example using this one. And here the write method is the most important method.

(Refer Slide Time: 20:20)



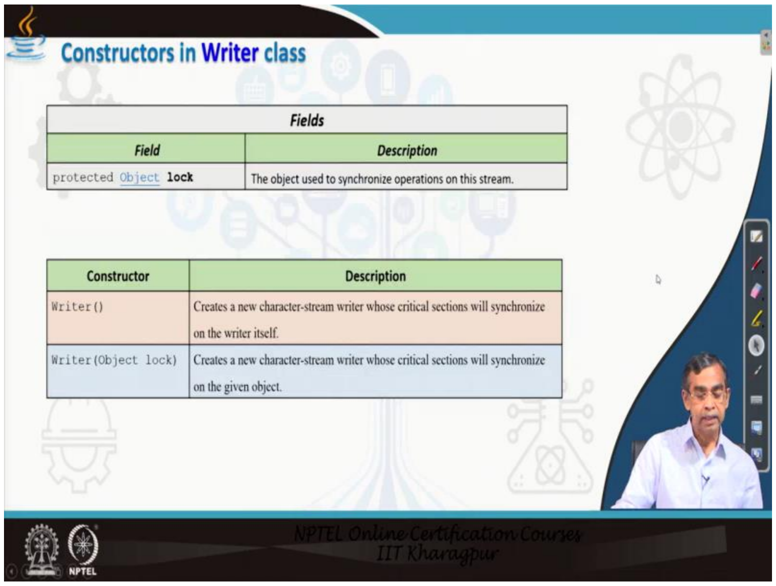
Sub classes of the Writer class

Class	Description
BufferedWriter	Writes text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings.
CharArrayWriter	This class implements a character buffer that can be used as an Writer.
PipedWriter	Piped character-output streams.
StringWriter	A character stream that collects its output in a string buffer, which can then be used to construct a string.
FileWriter	Convenience class for writing character files.
FilterWriter	Abstract class for writing filtered character streams.
PrintWriter	Prints formatted representations of objects to a text-output stream.

NPTEL Online Certification Courses
IIT Kharagpur

Write and close are the two methods are there. These are the different classes which you have already mention. Discuss in details here and more details you can obtain from the link that I will give you later on.

(Refer Slide Time: 20:35)



Constructors in Writer class

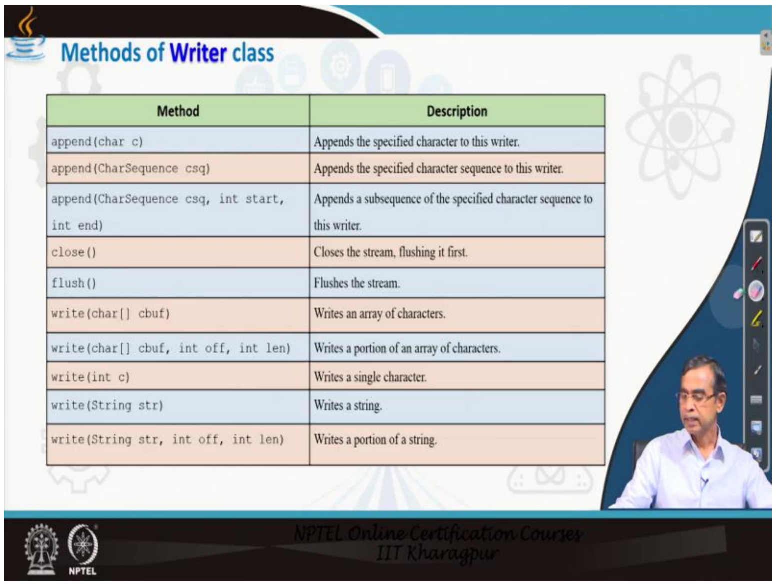
Fields	
Field	**Description**
protected Object lock	The object used to synchronize operations on this stream.

Constructor	Description
Writer()	Creates a new character-stream writer whose critical sections will synchronize on the writer itself.
Writer(Object lock)	Creates a new character-stream writer whose critical sections will synchronize on the given object.

NPTEL Online Certification Courses
IIT Kharagpur

Now, these are the, these are the way that you can create an object.

(Refer Slide Time: 20:40)



Methods of Writer class

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

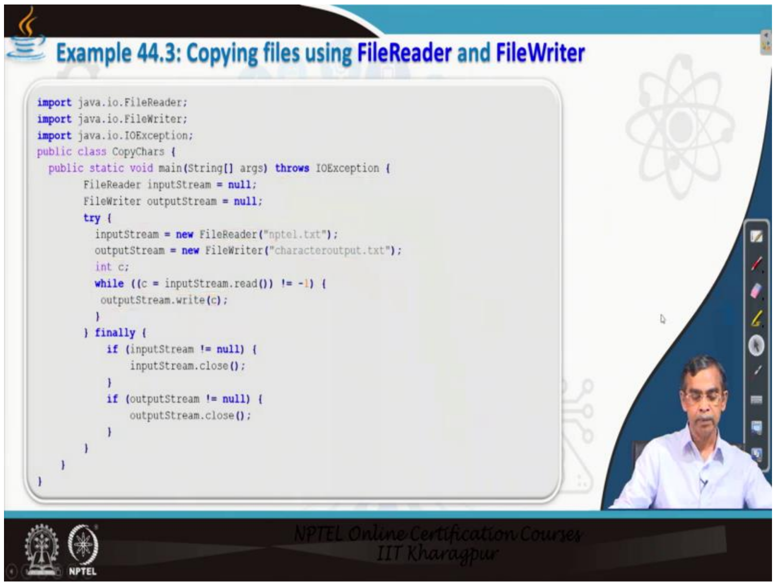
Object can be created I will give an example. But before going to the example let us see what are the different methods are define there in writer class. This are the method, append method if you want to, now usually the data or character if you want to write into the stream, it start from the very beginning. For example, a file, if the file contains some data already stored then if you write it. It basically non-appending means it is over write the data. Even append it will go to the end of the file actually.

So, append is there, then append as a character, append as a sequence of character, append as a character sequence with certain limitation or range of characters. Close is basically to close a output stream. And then flush is basically complete the writing actually so that means from your program every data will go force that no more writing is there so flush it. And these are the few write method as you see write has the different version.

Here write as a character as a array of characters you can write. Write as a character also, it is basically you can pass an integer it basically convert in the character because of the Unicode format. In the unicode like AASCI code every characters are represented by a integer value and all 0 to 250 same as a AASCI value, that you can note. And this is the also another writing as a array of characters is a writing as a string also it possible. Then writing string but starting with some off and length so that mean so for a part of the string also you can write into the buffer.

So, these are the basically different way, you can write into your output destination. Now, let us consider few program so that you can understand this concept better.

(Refer Slide Time: 22:34)



Example 44.3: Copying files using FileReader and FileWriter

```java
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class CopyChars {
  public static void main(String[] args) throws IOException {
        FileReader inputStream = null;
        FileWriter outputStream = null;
        try {
          inputStream = new FileReader("nptel.txt");
          outputStream = new FileWriter("characteroutput.txt");
          int c;
          while ((c = inputStream.read()) != -1) {
           outputStream.write(c);
          }
        } finally {
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
        }
    }
}
```

So, writing to an output stream we will consider again file as an output stream. Here is the program you can consider. This program is CopyChar. It is basically equivalent to copy bytes which you have already learned. So, this program what it will do, it is basically copy character by character from a input file and it also write into an output file. So, it is basically making a duplicate file actually.

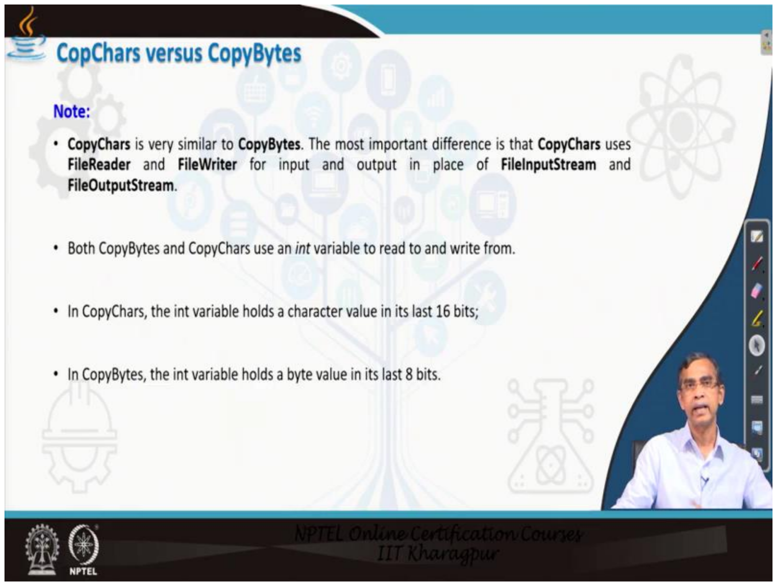
Now, let us see input file. So, file reader is basically the connection, let the name of the file form which you want to read, it is the nptel dot text and output file here we want to copy the content from nptel text to another output file say it is character output dot text. Assume that nptel dot text file is available in your working directory. If this file does not exist then it is an error, so this error can be caught by this exception or you can write try catch block enclose within this.

Now, here again you see within we have already used the try catch block as for this particular code. Whenever you create the two streams, one is input stream and another is output stream. Now, here is a basically we have to read continuously from input and reading from the nptel text and writing into the character output dot text. So, it is basically read from the input stream so read as a character so this is the c, c basically it is read as a character but character will be stored in according to the Unicode form. So, it is c integer that is a declare integer c and this read will continue until end of file. So, end of file will be mark with minus 1.

C is equals to minus 1 means no character actually is a nil we can say. Then we can write, write c means that whatever we read we can send back to a target file, it is called the write c write c output stream. This is output stream actually. So, this way we can see how character by character can be copied and store into an file. And you can again remember the same program which we have done for copy byte byte by byte also.

Say difference only that you declare as a byte and then we read as a byte, write as a byte. So, only this part is a different and this part is different because we are creating the character stream flow for there. So, these are the two things are important otherwise everything remains same reading as a byte or character.

(Refer Slide Time: 25:25)



CopChars versus CopyBytes

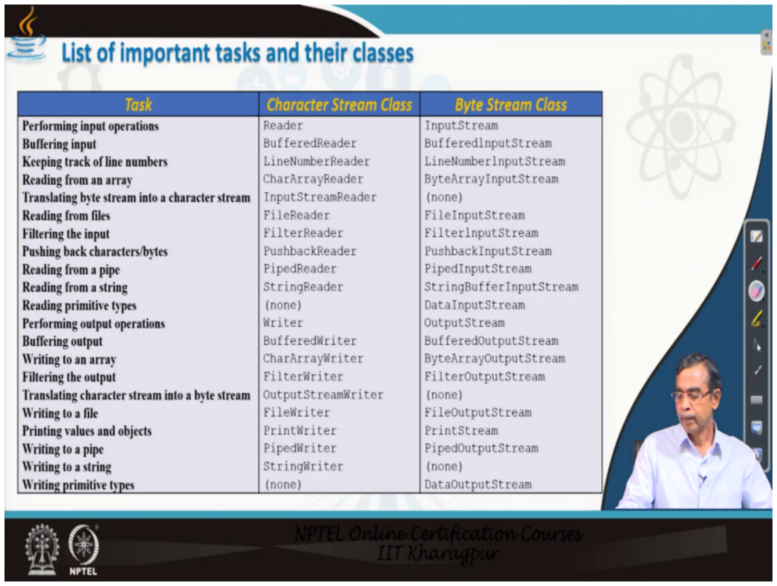
Note:

- **CopyChars** is very similar to **CopyBytes**. The most important difference is that **CopyChars** uses **FileReader** and **FileWriter** for input and output in place of **FileInputStream** and **FileOutputStream**.
- Both CopyBytes and CopyChars use an *int* variable to read to and write from.
- In CopyChars, the int variable holds a character value in its last 16 bits;
- In CopyBytes, the int variable holds a byte value in its last 8 bits.

NPTEL Online Certification Courses
IIT Kharagpur

From the programming point of view absolutely it is nothing as declare there. So, what you can say is that both CopyChar and CopyBytes. Logically, the same programming structure but internally they are different mechanism they will follow because one will read as a stream read stream as a flow or bytes and another will read stream as a character sets. So, this is the only thing that it matters.

(Refer Slide Time: 25:55)



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

NPTEL Online Certification Courses
IIT Kharagpur

And otherwise you have to create the connection that you can have it. Now, so let us see have a quick summary of different Java stream that is there. And so we have two different stream classes, one is called the byte stream and another is called the character stream. Now, I just want to give a analogy between the two. So, the input stream which is there it is basically reader. So, it is basically for input operation. I mean if you want to read something from the input data. Under this, there many other which is buffer input stream that means reading as a byte. The buffered reader is reading as a character.

So, line number input stream is basically reading as a set of characters. Here it is line number reader is basically. There reading as a set of bytes actually. It is basically set of numbers there. This is basically duplicated in the current feature. It is no more available but this is also going to duplicated but still it is there. Now, character array reader this is basically if you want to read from an array of characters. Here byte array input stream if you want to read as a bytes from an array of bytes.

The input stream reader is basically is a bridge that can read as a byte can written as character. Corresponding to there is no conversion, it is not good conversion that is read as a character and convert to the byte, it is not there somehow. So, file reader is basically analogical to the file input stream. That means if your input stream is file so file input stream. Then filter reader is basically as a filter form. You read one form and then convert into your technical specification of locale so that you can get the data according to your filter style and this corresponding to filter input stream and byte stream.

Push back reader is basically read and then you can written as a character here then as a byte also you can do the same thing in the byte stream. Then piped reader that means two threading if we want to send from one thread to another thread communication as a pipe rather. So, it is it is a piped input stream as a byte by byte communication here character by character communication. String reader if it is related to character string buffer input stream is related to the byte stream. Now, Data input stream is basically converting data from primitive to byte. But corresponding to this it is not there because all are basically primitive here in the character form.

Now, writer is basically is a character writing character into an output stream like. Here output stream is basically writing as a byte. And there is again this is related to the output stream different classes for character. And this is basically analogical to the byte stream. So, these are the different what is called analogy that we can. This analogy I mention because thus two things are same, you can have the same flavor actually but only internally whether you are reading as a byte or reading as a character. That matters otherwise both the things are more or less same.

And so this is basically idea about there and then I just want to mention one thing that in this situation two things are same. So, byte stream and character stream but which which mechanism you have to follow while you are writing program. So, there are obviously pros or cons. Byte stream if you see, it basically deals raw data directly from the input source whereas character stream data reads from the text.

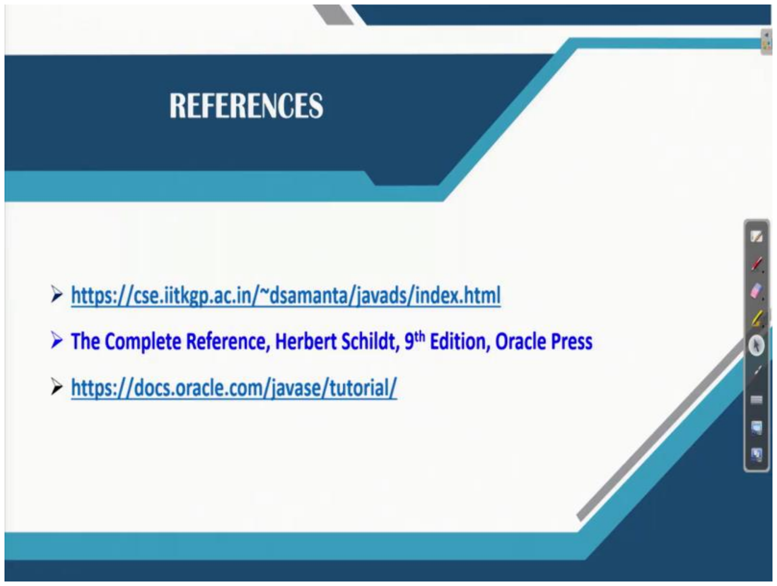
Now, reading raw by which is advantage here in some situation like if you want to read data from a image file, audio file or video file then raw data matters. So, in that case it is preferable to use the byte stream mechanism. Similarly, if you want to write something as a form of a byte, to an image or whatever it is there so byte stream is good. On the other hand, character stream is byte stream is in the sense is very fast because byte by byte is always very fast compare to the character by character which basically require the internal (confor) confirmation.

It essentially read from a byte because missing can understand only byte but you have to somehow some encoding decoding needs to be carried out before coming to the program. So, it is there in order to do that processing so character stream classes comparatively slower than the byte stream. Another important thing where the character stream is more recommendable. This is because the Java is mainly thing for the platform dependency.

If you write the program following the character stream classes, it is basically better. Better in the sense that because it basically independent or it basically manageable to any sort of platform so that is why unicode set has been planned and then it is adapted in Java system. And character stream class is basically is good to make the program or application platform independent. So, it is there so now you have to decide about whenever your program is related to input output where the text text internship data that mean all data related to some either character or something then you should think about character stream class.

So, mainly text processing (data) purpose character stream classes is good. Whereas, for data processing purpose if you see and data means multimedia data if you say then byte stream classes more preferable. So, this is the concept that is basically so far IO stream is concerned we have learned about character stream class in this lecture and so this is very important.

(Refer Slide Time: 32:07)



REFERENCES

- https://cse.iitkgp.ac.in/~dsamanta/javads/index.html
- The Complete Reference, Herbert Schildt, 9th Edition, Oracle Press
- https://docs.oracle.com/javase/tutorial/

And this are the reference that you can think for study further. This is the first link that you can think for running many more things in details and Herbert Schildt book is good to have many good discussion and there with illustration and then official documentation regarding the java edition release and then development and then many more discussion related to the many class in details that you can get it from the oracle website. Thank you very much!