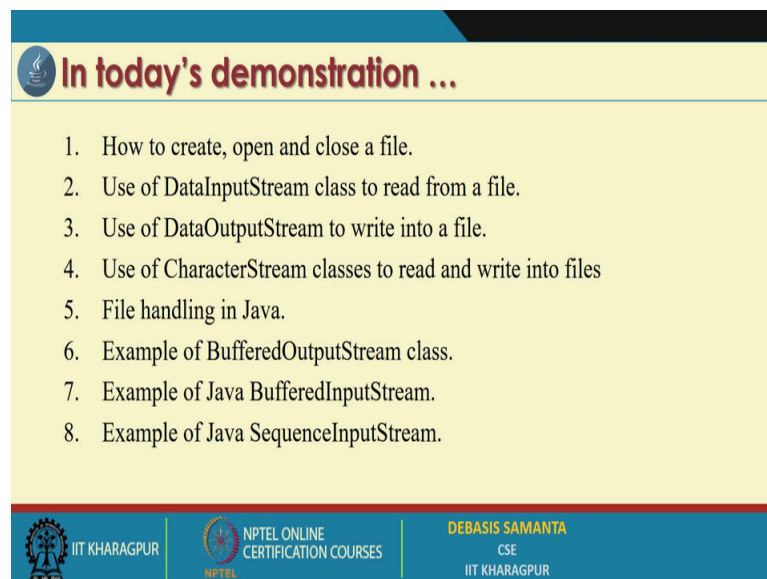


Programming in Java
Prof. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 33
Demonstration – XII

So, this demo is based on the topics that you have covered in the last three modules. That last three modules in the last three modules; we have cover on the topic Input-Output streams in java. And as you know the input and output is the very important, and what is called the activities is a very important activity in any program development. And in order to make this diversified input-output process that means input from the different sources, output to different sources, there are many many mechanism rather many ways has been devised in java system.

(Refer Slide Time: 00:53)



In today's demonstration ...

1. How to create, open and close a file.
2. Use of DataInputStream class to read from a file.
3. Use of DataOutputStream to write into a file.
4. Use of CharacterStream classes to read and write into files
5. File handling in Java.
6. Example of BufferedOutputStream class.
7. Example of Java BufferedInputStream.
8. Example of Java SequenceInputStream.

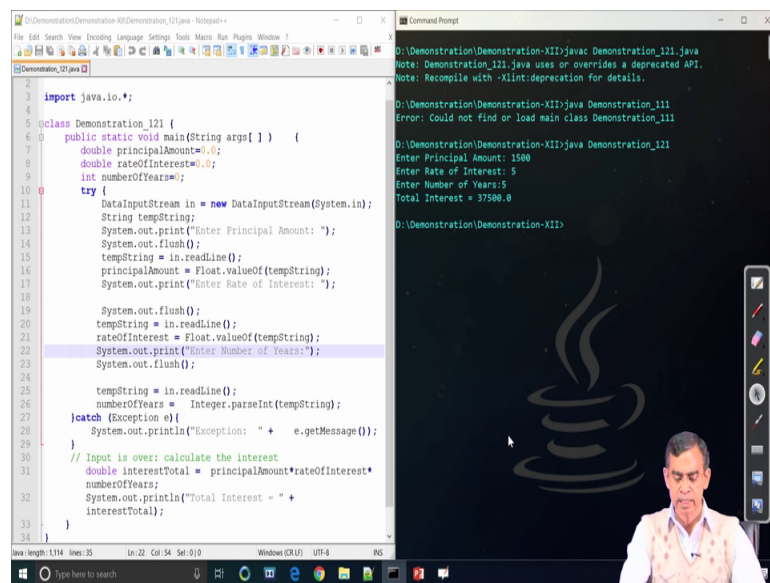
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA
CSE
IIT KHARAGPUR

Now, today we will discussed about how the different in way of input-output is possible. There are some trivial input output mechanism, those are related to the standard input, and then standard output namely keyboard, and then display unit we have already covered, we usually used it in our previous programs. But, here we will discussed about other than the standard input and output, how we can store some data into memory or we can receive something which is stored in hard disk like.

So, our demonstration includes how to create, how to open, and close a file, how we can create our own file. And there are again two different classes rather we can say the ways the byte stream classes, and then character stream classes to read and write into file from file by means of data input stream and the data output stream. So, we will discussed about the usage of these two classes to create files to access the data, all these things.

And then we will discussed again file handling java in java. So, file is basically secondary storage right, we can store some data in a permanent non-motile way into the secondary storage space. So, how we can create a file such a secondary storage, and also how we can open, how we can copy, how we can merge, so many other things are there. And few advanced methods, they are called buffered input stream, and sequence input stream to make our programming easy we will discuss in this demo.

(Refer Slide Time: 02:51)



```
import java.io.*;

class Demonstration_121 {
    public static void main(String args[] ) {
        double principalAmount=0;
        double rateOfInterest=0;
        int numberOfYears=0;
        try {
            DataInputStream in = new DataInputStream(System.in);
            String tempString;
            System.out.print("Enter Principal Amount: ");
            System.out.flush();
            tempString = in.readLine();
            principalAmount = Float.valueOf(tempString);
            System.out.print("Enter Rate of Interest: ");
            System.out.flush();
            tempString = in.readLine();
            rateOfInterest = Float.valueOf(tempString);
            System.out.print("Enter Number of Years:");
            System.out.flush();
            tempString = in.readLine();
            numberOfYears = Integer.parseInt(tempString);
        } catch (Exception e){
            System.out.println("Exception: " + e.getMessage());
        }
        // Input is over: calculate the interest
        double interestTotal = principalAmount*rateOfInterest*
        numberOfYears;
        System.out.println("Total Interest = " +
        interestTotal);
    }
}
```

```
D:\Demonstration\Demonstration-XII>java Demonstration_121.java
Note: Demonstration_121.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\Demonstration\Demonstration-XII>java Demonstration_121
Enter Principal Amount: 1500
Enter Rate of Interest: 5
Enter Number of Years:5
Total Interest = 37500.0

D:\Demonstration\Demonstration-XII>
```

So, let us have the demo for first thing, and this demo we can start about from very simple idea about using the data input stream class. We have already familiar to this class in our last in our earlier discussion we have used it. So, data input stream class as in the name in implies that it will basically for the input purpose, then we using this class, we can read something from some sources.

Now, here we will see exactly data input stream class can read from memory, can be from network channel, it can read from standard input device. This example illustrate how the data input stream class can be configured, so that we can read some data from

the standard input namely the keyboard. As we see the program here, so this is the main method as we see here. In this main method, we create two way fields namely the principal amount, and the rate of interest for which we want to read the value from the keyboard, and the number of years also. So, the three input needs to be collected from the user through keyboard.

Now, here basically we create an object called in, and this is object that object is created. And then this object wants that it is created by means of a constructor, where the input is basically system dot in. Now, system dot in implies that it is a standard input the system dot in this is already defined in the java dot line, so from there it will get the definitions. So, it will basically is a standard input.

Now, so basically what we do we here that we create and data input stream object, which basically connect your program to the keyboard. Now, let us see we given from to the user that entire principal amount, here basically we have discussed about that for seeing that means, keyboard has its own buffer. So, we have to clean the buffer, each time we are going to read from it.

So, system dot out dot flush is basically clean it is now here then ok, so it will read from the object in here that means which connect the keyboard. And the in that read line for this method read line means, it will read the entire content that is there is a buffer. So, the entire whatever the value we will enter, it is read it here, but as you know the value that will be there is a buffer java program read it as a stream. So, we temporarily store this in the stream format here, so it will basically read as a stream. Although these are may have entered some value say 5, 6, 5.25, so it is basically plotting point or double value.

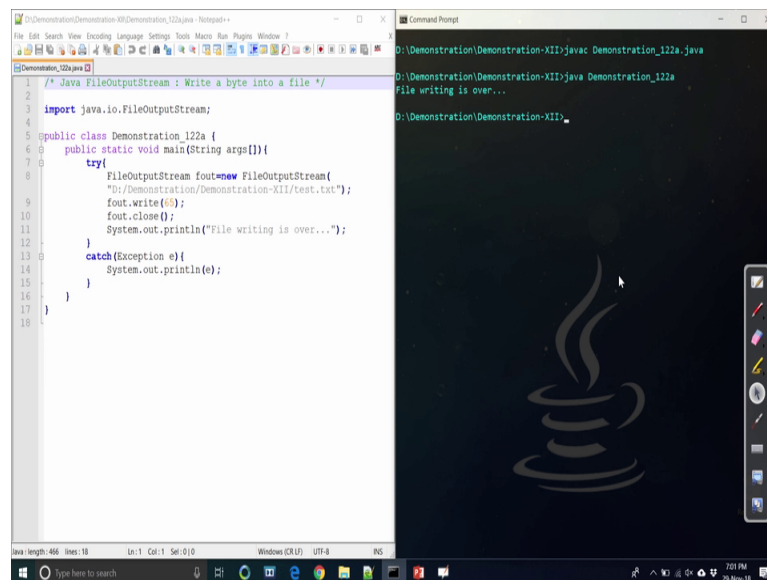
Now, here we convert this string value into our double value or float value. So, using the float method for this in the class float, which is defined in the java dot net (Refer Time: 06:02) package, and for this class there is a method called value of this is a static class actually. So, we just convert this string into the float value.

Now, again we read the another form enter interest again we flush it, and then read line, again read the buffer, and then again convert this string into the float value which stored in the rate of interest. Say again then we need the integer number from the read, we convert it into integer format as initially it is a string, and then store the value is a

number of years here. So, this is the way we can read the three inputs from the standard in standard input that is the keyboard here.

And then finally, we calculate these are trivial process. Now, main thing here we can see you can tell here is that how using this data input stream class, we can create a stream object which connect your program to your keyboard. So, the stream can be propagated from the input source to the program. Anyway, so this is the simple program we are already familiar to, and we have discussed it many times earlier. And other than this class also there are many way the input can be found the standard device like common line input, and then the using the scanner class which is defined in java dot util package, we can run all this things.

(Refer Slide Time: 07:43)



The image shows a screenshot of a Windows desktop environment. On the left, a Notepad++ window is open, displaying the source code for a Java class named 'Demonstration_122a'. The code imports 'java.io.FileOutputStream' and defines a 'main' method that creates a 'FileOutputStream' object pointing to 'test.txt', writes the character 'G' to the file, and prints 'File writing is over...'. On the right, a Command Prompt window shows the execution of the program. The prompt shows the compilation command 'javac Demonstration_122a.java' and the execution command 'java Demonstration_122a', with the output 'File writing is over...' displayed.

```
1 /* Java FileOutputStream : Write a byte into a file */
2
3 import java.io.FileOutputStream;
4
5 public class Demonstration_122a {
6     public static void main(String args[]){
7         try{
8             FileOutputStream fout=new FileOutputStream(
9                 "D:/Demonstration/Demonstration-XII/test.txt");
10            fout.write('G');
11            fout.close();
12            System.out.println("File writing is over...");
13        }
14        catch(Exception e){
15            System.out.println(e);
16        }
17    }
18 }
```

```
D:\Demonstration\Demonstration-XII>javac Demonstration_122a.java
D:\Demonstration\Demonstration-XII>java Demonstration_122a
File writing is over...
D:\Demonstration\Demonstration-XII>
```

Now, our next example that we are going to discuss about is basically as we have discussed about all the data input stream class is basically the two fold using the byte stream, and then character stream. Our next example this examples to illustrate how we can output to a file which is stored in a secondary storage, and we can propagate the output from the program to the file in the byte form. As you know byte is the smallest unit a chunk that the java program can handle it.

Now, here is a program as you see here ok. First you have to import java dot io dot file output stream, this is because we want to propagate the output to a file. And this a file output stream class in defined in the java dot io dot package, so the import is must. Now,

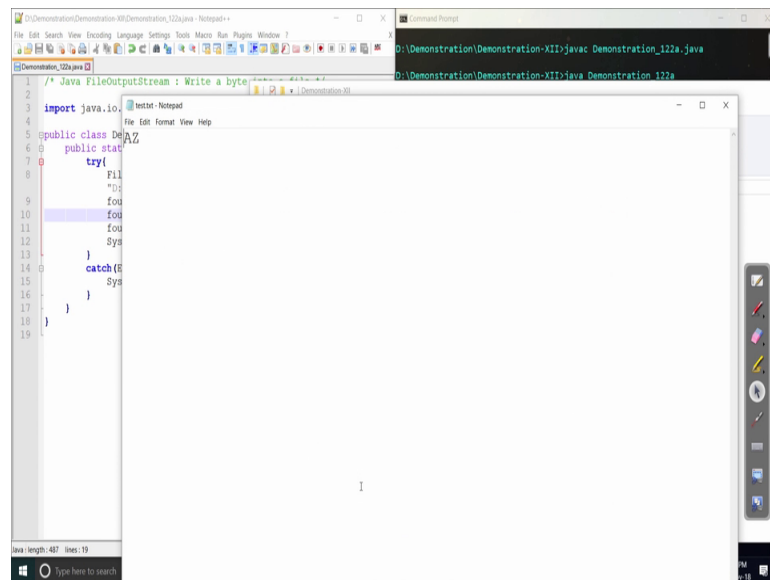
here is the main method as we see we create an object name of the object is a f out is basically of type file output stream. And here you see the constructor the file output stream has its own constructor, where an argument is required is basically the target. So, here actually the target we have mentioned explicitly, where we want to store this content.

So, as you see it is in the D drive under these D drive there is a directive demonstration, under this demonstration XII, and then this is basically the target file and test dot txt. So, you should explicitly mention path of the file, where you want to store your data. So, this is a way that we can do it. Now, here you see for this object f out, f out therefore object is basically the connection from your program to this file test dot txt.

Now, from the programme we can write 65. Now, 65 is basically here in integer form, but it will be converted to and byte form as you know the 65 byte in the byte form, it is basically capital A character. So, ultimately although we give the 65 your program the system will convert into it a byte that means, 65 in the byte is basically the ASCII code is a. So, a will be store actually in the file. And then finally, once the file is open, we have to close the file it is always, you should close your file always wants your task is over, and then finally it print the writing is over.

Now, this program as you see this program will write 65 in the byte code form into a file test dot txt. So, after the successful execution of this program, we will see the output file here test dot txt that 65 is told there or not. So, this program is run successfully, and then output file is here. The test dot text test or text is now displayed as we see it basically store a.

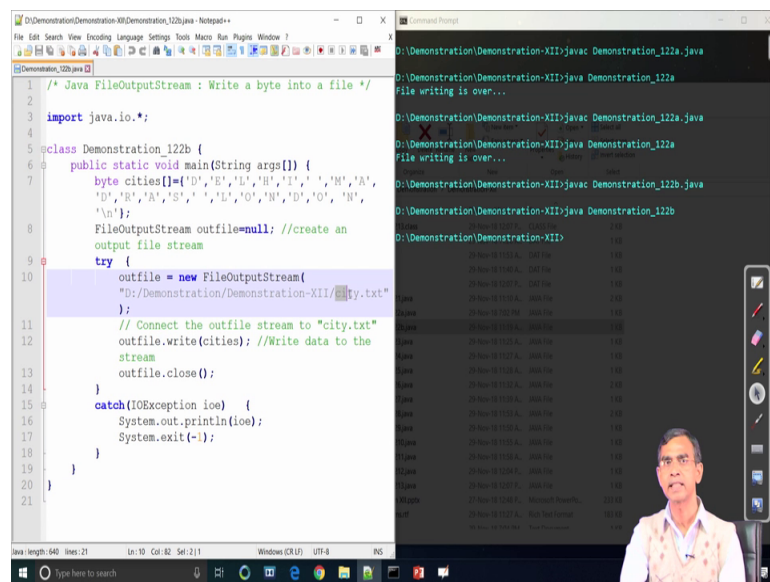
(Refer Slide Time: 10:41)



```
1 /* Java FileOutputStream : Write a byte
2
3 import java.io.*;
4
5 public class Demo {
6     public static void main(String args[]) {
7         try {
8             File f = new File("D:\\Demo\\city.txt");
9             FileOutputStream out = new FileOutputStream(f);
10            out.write('f');
11            out.write('f');
12            out.write('f');
13            out.write('f');
14            out.write('f');
15            out.write('f');
16            out.write('f');
17            out.write('f');
18            out.write('f');
19            out.write('f');
20            out.write('f');
21            out.write('f');
22            out.write('f');
23            out.write('f');
24            out.write('f');
25            out.write('f');
26            out.write('f');
27            out.write('f');
28            out.write('f');
29            out.write('f');
30            out.write('f');
31            out.write('f');
32            out.write('f');
33            out.write('f');
34            out.write('f');
35            out.write('f');
36            out.write('f');
37            out.write('f');
38            out.write('f');
39            out.write('f');
40            out.write('f');
41            out.write('f');
42            out.write('f');
43            out.write('f');
44            out.write('f');
45            out.write('f');
46            out.write('f');
47            out.write('f');
48            out.write('f');
49            out.write('f');
50            out.write('f');
51            out.write('f');
52            out.write('f');
53            out.write('f');
54            out.write('f');
55            out.write('f');
56            out.write('f');
57            out.write('f');
58            out.write('f');
59            out.write('f');
60            out.write('f');
61            out.write('f');
62            out.write('f');
63            out.write('f');
64            out.write('f');
65            out.write('f');
66            out.write('f');
67            out.write('f');
68            out.write('f');
69            out.write('f');
70            out.write('f');
71            out.write('f');
72            out.write('f');
73            out.write('f');
74            out.write('f');
75            out.write('f');
76            out.write('f');
77            out.write('f');
78            out.write('f');
79            out.write('f');
80            out.write('f');
81            out.write('f');
82            out.write('f');
83            out.write('f');
84            out.write('f');
85            out.write('f');
86            out.write('f');
87            out.write('f');
88            out.write('f');
89            out.write('f');
90            out.write('f');
91            out.close();
92        } catch (IOException ioe) {
93            System.out.println(ioe);
94        }
95    }
96 }
```

Now, in the same program if we again write 90 f out write 90 type it no another write, 65 is the here write yes in addition write.

(Refer Slide Time: 10:59)



```
1 /* Java FileOutputStream : Write a byte into a file */
2
3 import java.io.*;
4
5 class Demonstration_122b {
6     public static void main(String args[]) {
7         byte cities[] = {'D','E','L','H','I','M','A',
8             'D','E','A','S','I','O','N','D','O','N',
9             '\n'};
10        FileOutputStream outfile = null; //create an
11        // output file stream
12        try {
13            outfile = new FileOutputStream(
14                "D:\\Demonstration\\Demonstration-XII\\city.txt");
15            // Connect the outfile stream to "city.txt"
16            outfile.write(cities); //Write data to the
17            // stream
18            outfile.close();
19        } catch (IOException ioe) {
20            System.out.println(ioe);
21            System.exit(-1);
22        }
23    }
24 }
```

Now, we are writing another 90, as you see it will happened after the first 65 that mean A, and corresponding the 90 byte code, it will be stored there in the same file again. Now, here the same file can be AZ. So, Z is basically the ASCII code for the 90 ok. So, we are learned about how in the byte form, we learn more about in the byte. So, the next example, suppose you want to store an array of elements into a file. Here we can give an

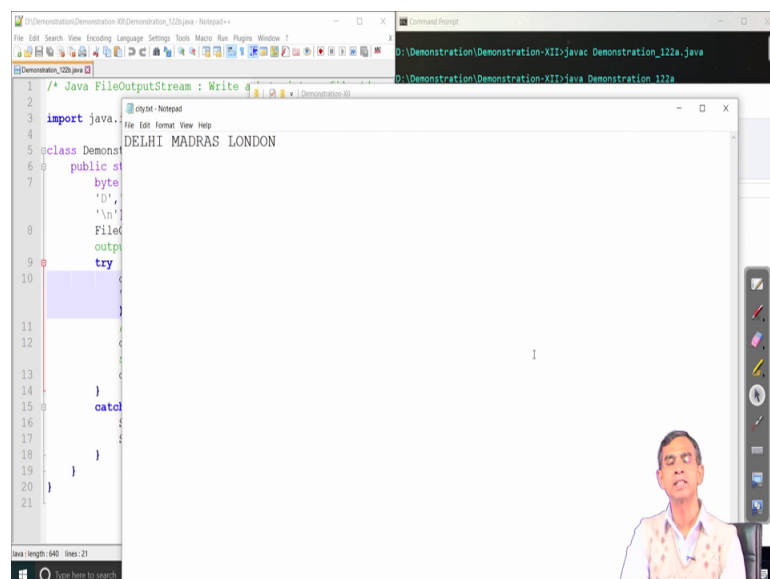
idea about how using the bytes code system, and array of elements can be stored into an into a file.

Now, this program as we see the main program we first created array of bytes, so here D, E all these are basically the ASCII value within the single code whatever we mention, it is basically the ASCII value of that character. So, capital D, E, L, H, I, so ultimately like the previous program as we see this is the byte so A it will store, but it will store now in the array. And this array is basically byte array, because it is declared as a byte type array. We can declare float, integer also that way it will store the integer from that way.

Anyway, so this basically now we create one output objects, so output file is a type of file output stream very similar to the previous one. And then we decide it is a create the object, and finally we make a connection that where this value will store. So, for this object output file, we establish a connection to this is the location. So, the location is as we see location is city dot txt that means, this is the output destination target, where the output the entire things whichever here will be there.

And then finally, you write the entire array here, so passing the array name here and finally close, these are the same as earlier now ok. So, this way the entire array elements will be stored into the file. Now, let us run the program, and final you will be able to see the file in this case city dot text txt, which store all the values that is there in the byte array ok. So, file has been written successfully.

(Refer Slide Time: 13:43)



The screenshot shows a Java IDE window with the following code:

```
1 /* Java FileOutputStream : Write a
2
3 import java
4
5 class Demonst
6 public st
7 byte
8 '\n'
9 File
10 outp
11 try
12
13
14 }
15 catch
16
17 }
18 }
19 }
20 }
21
```

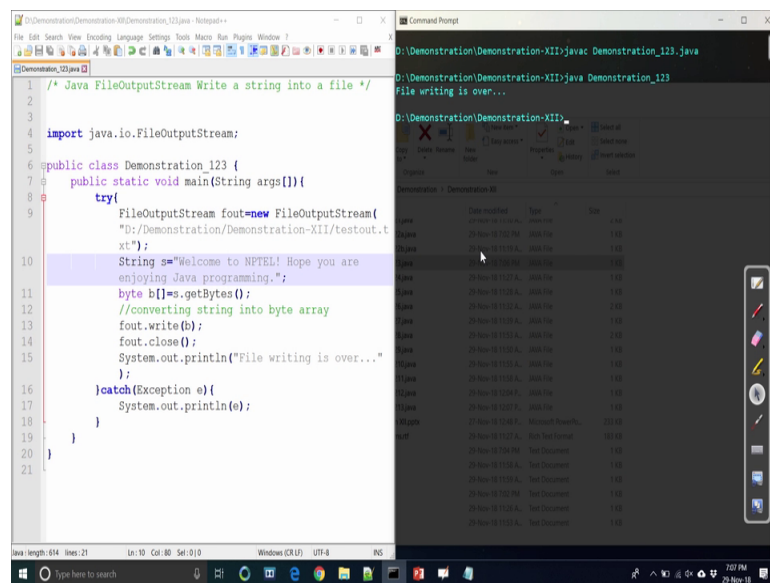
The Command Prompt window shows the following command and output:

```
D:\Demonstration\Demonstration-XII> javac Demonstration_122a.java
D:\Demonstration\Demonstration-XII> java Demonstration_122a
```

The IDE also shows a Notepad window with the text: DELHI MADRAS LONDON

And then we will be able to run the we are display city dot txt as we see here Delhi, Mumbai, London. These are the basically ASCII character as it is there in the byte array. So, it is basically print a store successfully. So, this program says how a an array, and the similar way this program can be an extended to store any array actually, whatever the you can use a byte array rather you can use the characteristic classes also whatever it is there.

(Refer Slide Time: 14:07)



The image shows a screenshot of a code editor and a command prompt. The code editor on the left displays the following Java code:

```
1  /* Java FileOutputStream Write a string into a file */
2
3
4  import java.io.FileOutputStream;
5
6  public class Demonstration 123 {
7      public static void main(String args[]){
8          try{
9              FileOutputStream fout=new FileOutputStream(
10                 "D:/Demonstration/Demonstration-XII/testout.txt");
11                 String s="Welcome to NPTEL! Hope you are
12                 enjoying Java programming.";
13                 byte b[]=s.getBytes();
14                 //converting string into byte array
15                 fout.write(b);
16                 fout.close();
17                 System.out.println("File writing is over...")
18             };
19         }catch(Exception e){
20             System.out.println(e);
21         }
22     }
23 }
```

The command prompt on the right shows the execution of the program:

```
D:\Demonstration>javac Demonstration_123.java
D:\Demonstration>java Demonstration_123
File writing is over...
```

So, an array of elements can be stored, a single element can be stored, and like this, here we are storing all these things in our memory. Now, our next example again using file output stream classes that we have discussed that we have to pay to copy some text using again byte array steam into the same file.

Now, here is a program as we see, it is the same file output stream class. Here file output stream class, create the object, and we just connect it to one target test out dot txt. And here we create a stream here welcome to nn NPTEL. Here is a small string we have considered, many other large string also we can include say, other things also you can input here.

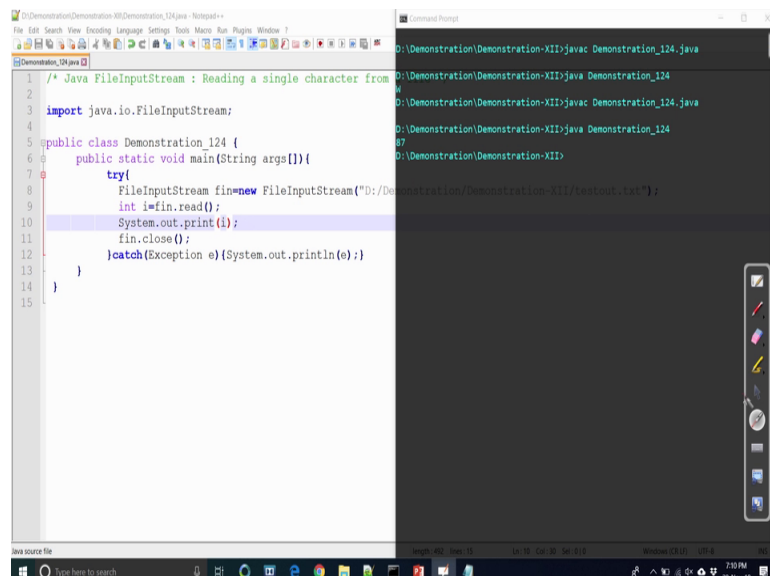
For example, type here welcome to NPTEL, hope you are enjoying java programming hope you are enjoying java programming ok. So, this is the text whatever the text may be here only few characters, very big I mean very large file also can be consider. I will discuss about how a file can be entire file can be copied here, anyway our objecting is

that these stray string, which is basically here in the program, I want to store into a file the name of the file is test out dot txt.

Now, we first store this string into the form of a bytes. So, here is a mechanism for a string object, how we can convert into the bytes. So, basically string this is the string and we converting into the entire string content is the byte. So, get bytes is a method for the string class, which is there in java dot (Refer Time: 15:57) package. So, we temporary stored into an array byte array, it is the similar to the city array we have as discuss there. Then write function write method for this a foul file output stream class, which basically write the entire array, it is also same as the earlier one and close. And finally, file writing is over (Refer Time: 16:16), and then it will there.

Now, we see after the successful running of this program. We will see one file test out dot text has been created, where we could store the stream which has mentioned there fine. As we see this the yes, so now we are displaying the file content as we see this is the content has been pushed to the file, which we have displaced there. Now, so this is the way that now so far what we have done is that we can create something we can store something from our program to file.

(Refer Slide Time: 16:57)



Now, we learn about how to read a content from a file. Now, this very simple program for this purpose for reading some content from the file, we have to have that create object for the file input stream class. So, file input stream class, because we have to go for input

process. And then `f` is the name of the object of this class we have created, and this is a usual process of creating the object, and giving the explicit mentioning where actually from which file we want to read.

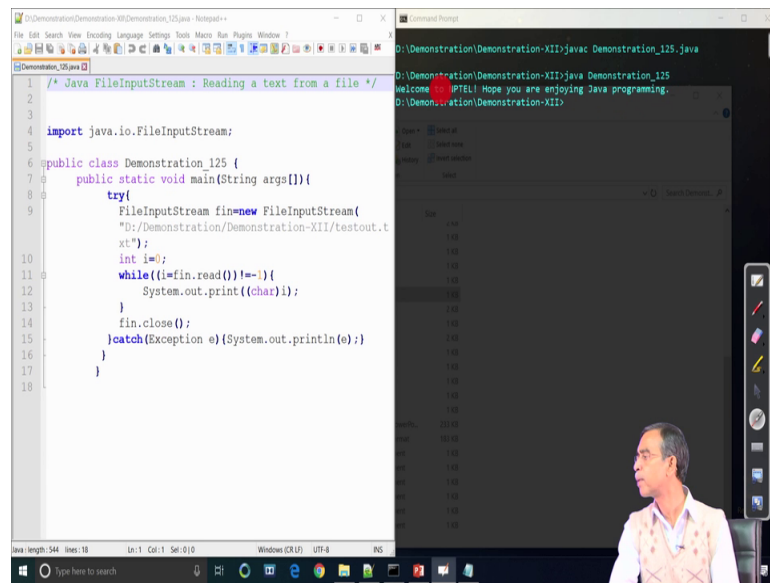
So, in this example we want to read something from `test out dot txt`, which file we have already created like `welcome to NPTEL` like. And here `int i = f.read()`, you little bit carefully observe for the `fin` object. A `fin` object means that is a connection from this program to this file, we inform the method `read`.

Now, `read` method always read one character at a time, so this means that the whenever we create the object `file` object here, it basically point to the first character in that file, and it will basically read that character. And this character actually written by this `read` method as an ASCII value of the character that means, it will read there. And finally, we can print this character by casting it that means, `i` its ASCII value will be converting the character corresponding to this, and it will be stored there.

So, you can note that in the previous example `test dot out`, we have store the `welcome to the NPTEL`. So, `W` has been store there, and `W` being the ASCII value, and then it is it will be access it, and then it will display from your program. Now, here you can see here you can see here, you can see `W` is being printed. Now, if little bit change this program without casting it, you can see here without casting it, we can run the same program again, but only the its integer value rather it is basically not ASCII code, it is the ASCII value.

Now, here we see we print here in case `87` and `87` is basically the ASCII value for the ASCII character `W`. So, this is basically the way here exactly we save this programme earlier in the byte code form, here we read in that ASCII form byte form, and then printing into its character. So, conversion is it possible. So, this means actually I want emphasize is that whatever the way in the byte or character, you can do it and you can read it also no issue it is there.

(Refer Slide Time: 19:55)



The screenshot shows a Java IDE with two windows. The left window displays the source code for a class named `Demonstration_125`. The code imports `java.io.FileInputStream` and defines a `main` method that reads a file named `testout.txt` from the directory `D:\Demonstration\Demonstration-XII`. The code uses a `while` loop to read characters from the file until it reaches the end of the file (indicated by `-1`). The output of the program is shown in the right window, which displays the text: `Welcome to NPTEL! Hope you are enjoying Java programming.`

```
1 /* Java FileInputStream : Reading a text from a file */
2
3
4 import java.io.FileInputStream;
5
6 public class Demonstration_125 {
7     public static void main(String args[]){
8         try{
9             FileInputStream fin=new FileInputStream(
10                "D:/Demonstration/Demonstration-XII/testout.t
11                xt");
12             int i=0;
13             while((i=fin.read())!=-1){
14                 System.out.print((char)i);
15             }
16             fin.close();
17             }catch(Exception e){System.out.println(e);}
18         }
19     }
```

```
D:\Demonstration\Demonstration-XII>javac Demonstration_125.java
D:\Demonstration\Demonstration-XII>java Demonstration_125
Welcome to NPTEL! Hope you are enjoying Java programming.
D:\Demonstration\Demonstration-XII>
```

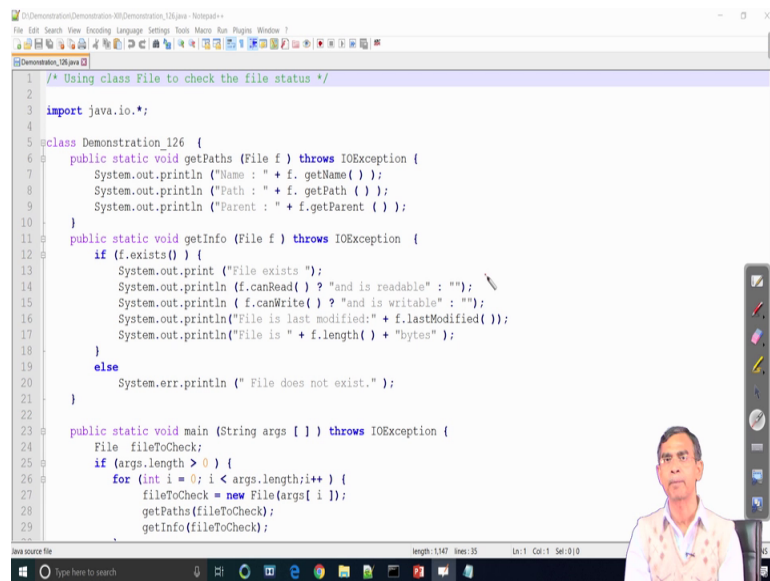
Now, we learn about more about regarding the charactering the stream classes. As we have seen in the last example, we could read only one character from the file. If you want to read the entire character set stream text from a file, what is the program should look like? In this case, again we have to create an object for the file input stream, because we want to read a file. And here is the object the same way, we created the object a `f in`. And this is the for example, `test out dot txt` is the target file from, where you want to read the entire text. As you know in this text, we have written `welcome to NPTEL hope you are enjoying java programming like`.

Now, here this basically is the loop instead of only single character, we want to read it all the characters which are stored there in this `test out dot file`. Now, a `f in dot read`, it is basically to be looped until here is the condition. This is very important not equals to minus 1, so not equals to minus 1 java implies that it is the end of file. At every end of file whenever you put `file close`; the java put a minus 1 there indicating that this is the end of the file, so minus 1 is a termination condition for a file object.

Now, it will basically read the file content. And here the content which will be read from their file input stream in the byte form, we convert the character and those things will be converted printed on the screen. And finally, wants the entire file is scanned, it will basically closed. And here you can see it will read the file sequentially the first, then second, and third and so on so on, until it will reach to the end of file.

So, these program again as you can understand anticipate. If we run, it is basically extract all the content, which is there in test out dot txt, and display on your terminal. So, I am running this program, and then will see after running. So, this program has we see here, it basically show the output entire which has been accessed from the text out dot. This is the basically content, which we have stored in the taste out file and this exercise and displayed it.

(Refer Slide Time: 22:03)



```
1 /* Using class File to check the file status */
2
3 import java.io.*;
4
5 class Demonstration_126 {
6     public static void getPaths (File f) throws IOException {
7         System.out.println ("Name : " + f.getName ());
8         System.out.println ("Path : " + f.getPath ());
9         System.out.println ("Parent : " + f.getParent ());
10    }
11    public static void getInfo (File f) throws IOException {
12        if (f.exists() ) {
13            System.out.print ("File exists");
14            System.out.println (f.canRead() ? "and is readable" : "");
15            System.out.println ( f.canWrite() ? "and is writable" : "");
16            System.out.println("File is last modified:" + f.lastModified());
17            System.out.println("File is " + f.length() + "bytes");
18        }
19        else
20            System.err.println (" File does not exist.");
21    }
22
23    public static void main (String args [] ) throws IOException {
24        File fileToCheck;
25        if (args.length > 0 ) {
26            for (int i = 0; i < args.length;i++) {
27                fileToCheck = new File(args[ i ]);
28                getPaths(fileToCheck);
29                getInfo(fileToCheck);
30            }
31        }
32    }
33 }
```

Now, so we are learned about how we can open a file. And then from that file we can read the content in the form of a byte. And our next example is basically regarding the file status checking. So, it is a 12.6 programme, we are going to see about it. Now, here you know so for the status of the file is concerned. Whenever we store a file or operating system maintain all about its file that mean in which directory what is the name of the file, what is the extension, whether the file is in which mode readable or writable, whether file is available or not available, whether file is corrupt, whatever the information it is there. So, it is basically the program, which all these information can be accessed from the program site itself.

Now, here we can see this is the program, and it basically two methods we have discussed here in this class, this is basically our main class. The method is get paths, and another method is gate info. Now, so for the gate path method is concerned for a object file object. So, file is basically one class, which is define in java dot io io package, so dot

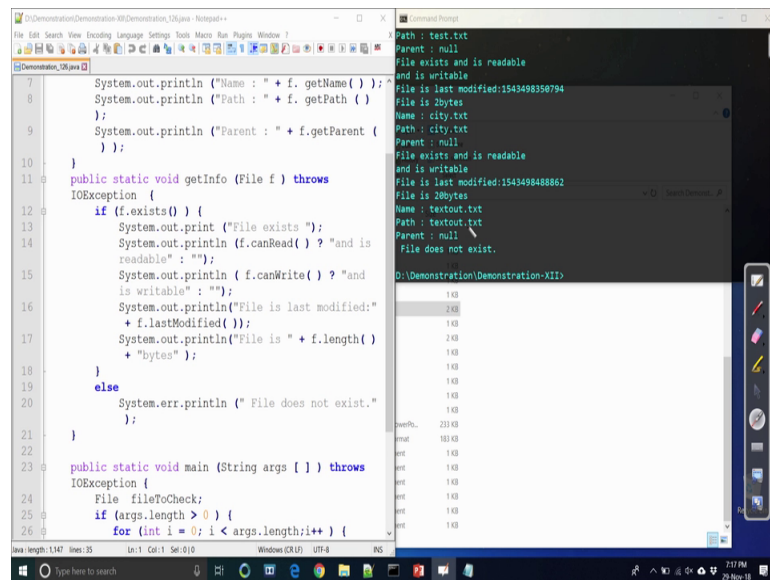
star include that file package the file class need to be accessed. So, file for this we will pass this file object, whenever you call this get path.

And now here you see this get path method input simple three statement namely, it will basically give the name of the file f dot get name, the name of the file f dot gate path, which path actually it is stored. And f dot get parent what is the parent, if it is there. And then so far the get info method is concerned, if the file exist that means, if file is not null, file exist means you can pass information which is no more exist. So, in that case if it is exist, it will just exception.

If it is exist it is true, then it will go for this printing f dot can read, f can write that means, it basically whether this file is readable or write writeable, read mode or write mode, read permission, write permission. And then it also say the last modified whenever you created file, system always store at the last modification of the date of this one so this one. And f dot length is basically how many bytes, basically how many byte basically are stored there in this file. So, they are the information.

Now, let us come to the main method here. So, this is the main method this is the main method. Here basically we create one file object name of the object is file to check. And then if args dot length, basically we can we can pass the common line input, so that we can run this meth program for as many file you want to check for it. So, it is basically args dot length args sense. And for each file argument we have passed it, it will basically call the get path and get info method that means, it will retrieve all the information regarding the file name path and status of it, and finally it will come to the end of this program.

(Refer Slide Time: 25:27)



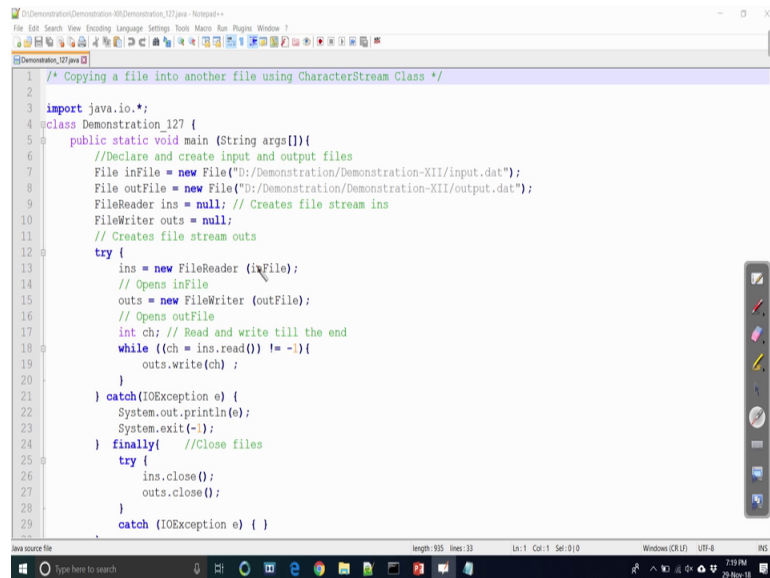
```
7      System.out.println ("Name : " + f. getName ( ) );
8      System.out.println ("Path : " + f. getPath ( )
9      );
10     System.out.println ("Parent : " + f.getParent (
11     ) );
12     }
13     public static void getInfo (File f ) throws
14     IOException {
15         if (f.exists() ) {
16             System.out.print ("File exists ");
17             System.out.println (f.canRead() ? "and is
18             readable" : "");
19             System.out.println ( f.canWrite ( ) ? "and
20             is writable" : "");
21             System.out.println("File is last modified:"
22             + f.lastModified( ));
23             System.out.println("File is " + f.length (
24             )
25             + "bytes" );
26         }
27     }
28     else
29         System.err.println (" File does not exist."
30         );
31     }
32     }
33     public static void main (String args [ ] ) throws
34     IOException {
35         File fileToCheck;
36         if (args.length > 0 ) {
37             for (int i = 0; i < args.length;i++ ) {
38                 fileToCheck = new File (args[i]);
39                 getInfo (fileToCheck);
40             }
41         }
42     }
43 }
```

```
Path : test.txt
Parent : null
File exists and is readable
and is writable
File is last modified:1543498350794
File is 2bytes
Name : city.txt
Path : city.txt
Parent : null
File exists and is readable
and is writable
File is last modified:1543498488862
File is 20bytes
Name : textout.txt
Path : textout.txt
Parent : null
File does not exist.
```

Now, will run this program passing three input as a common line, and say the input is input is say city dot text, we know we have created one file test text dot txt and test out dot txt. These are three files are already we have created. Now, so we can see ok, we are running this program using passing three input city txt, txt dot text, and text out dot txt fine.

Now, here you can see the output as we see path test dot txt parent there is no parent actually indicating null, and then and is writable. Last modified this is basically the system time it is in a some (Refer Time: 26:13) form, and then file size is two bytes. Now, name city text, path is city text, parent there is no parent for the, this file. Readable file it is also it is a writable, last modified this one 20 bytes, this also like this. And here you see text dot out text this basically file does not exist, because this file actually it is not there whatever it is here.

(Refer Slide Time: 26:47)



```
1  /* Copying a file into another file using CharacterStream Class */
2
3  import java.io.*;
4  class Demonstration_127 {
5      public static void main (String args[]){
6          //Declare and create input and output files
7          File inFile = new File("D:/Demonstration/Demonstration-XII/input.dat");
8          File outFile = new File("D:/Demonstration/Demonstration-XII/output.dat");
9          FileReader ins = null; // Creates file stream ins
10         FileWriter outs = null;
11         // Creates file stream outs
12         try {
13             ins = new FileReader (inFile);
14             // Opens inFile
15             outs = new FileWriter (outFile);
16             // Opens outFile
17             int ch; // Read and write till the end
18             while ((ch = ins.read()) != -1){
19                 outs.write(ch) ;
20             }
21         } catch(IOException e) {
22             System.out.println(e);
23             System.exit(-1);
24         } finally{ //Close files
25             try {
26                 ins.close();
27                 outs.close();
28             }
29         } catch (IOException e) { }
```

So, whatever file you have passed to the program, this program from side it can check the status, and can show you. Now, we have discussed about so far the byte stream classes for reading as well as writing. Now, it is our time to learn about other classes, so for the file handling is concerned. They are called file reader and file writer class.

Now, we are going to have a demo. Here in this demo, we will see how we can copy the content of one file into another, so basically making a duplicate, and using file reader and file writer class. So, this program again we make the program for illustration as simple as possible, we can see we fast create one object in file for the file. This basically creates a connection from this program to this target date input data. Assuming that input dot dat file is already there in the system. If it is not there, then it is throw an exception, because file opening will not be possible anyway.

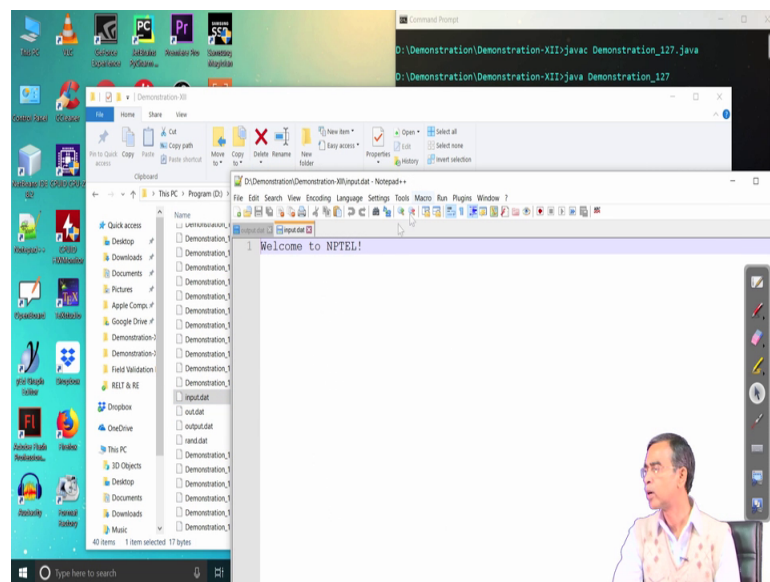
So, this is also another out file, and this basically where the content will store. So, here we have created two file, in file and out file for the two purpose; purpose is that reader and writer. So, I just create an object ins, so that in file can be used for the input purpose that means, from in file I will be able to read something. So, inside out are the two objects of type file reader and file writer for the input and output mechanism.

Now, here for ins we create the file reader object for the infile. So, in this case infile means, it is basically input data. So, in other words ins means, basically a connection from your program to infile, infile means input dot dat. Similarly, outs is the connection

from your program to out file that is the output dot dat file. Now, here basically we have to read each and every character those are there input dot dat that means, infile. And then write into the out display on the it is basically read the entire file, and then outs write means it will store into the output file.

Here we see the statement that we have ins read means read one character from the file at a time store as ch the temporary store. And then same ch, it write into the output file outs. So, reading one character from the ins, writing the same character into the outs that mean input date and output date. So, this program and also we have filled up with try and catch you have to always use try and catch block.

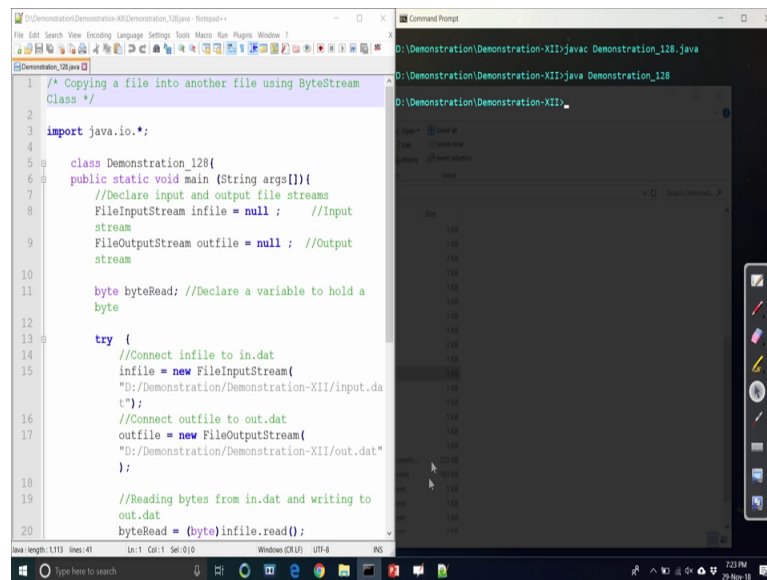
(Refer Slide Time: 29:27)



As there may be some situation unwanted situation, whenever a file is not accessible the file read permission is not there, write permission is there or file is no more memory space is available to avoid so many situations, we have to handle the exception. So, those things can be done by try catch mechanism.

So, you have to handle for every occurrences either creating file or reading or writing, whatever right so they should be put into the try catch flow there. Now, here is basically as we see the output welcome to NPTEL is the input dot file. And we copy this the same content, and also create the output file welcome to NPTEL. So, we created the duplicate of one file input dat the name of the duplicated file is output dat.

(Refer Slide Time: 30:29)



```
1  /* Copying a file into another file using ByteStream
2  Class */
3
4  import java.io.*;
5
6  class Demonstration_128{
7      //Declare input and output file streams
8      FileInputStream infile = null ; //Input
9      FileOutputStream outfile = null ; //Output
10     stream
11
12     byte byteRead; //Declare a variable to hold a
13     byte
14
15     try {
16         //Connect infile to in.dat
17         infile = new FileInputStream(
18             "D:/Demonstration/Demonstration-XII/input.da
19             t");
20         //Connect outfile to out.dat
21         outfile = new FileOutputStream(
22             "D:/Demonstration/Demonstration-XII/out.dat"
23         );
24
25         //Reading bytes from in.dat and writing to
26         out.dat
27         byteRead = (byte)infile.read();
28     }
29 }
```

So, we have learned about how a file copy is possible. Now, here another example in the last example, we have discuss that copying the file using characters, so it will basically character stream classes. We want to do the same thing again, but using byte stream classes. The difference is that in the last example, they will read as a character, but here they will read as a byte that is all.

Now, the mechanism is almost same, but there is again reason that when we have to read character, when we have read byte. Whenever the hydrogenate is concern, then you should use the byte stream classes. And homogeneity means in the same system whatever it is there, you have created the file in macro ways, and want to read from the windows, then obviously byte stream is the best procedure. But, if it is the same system (Refer Time: 31:09), then you can use whatever method you can. So, byte stream is always preferable.

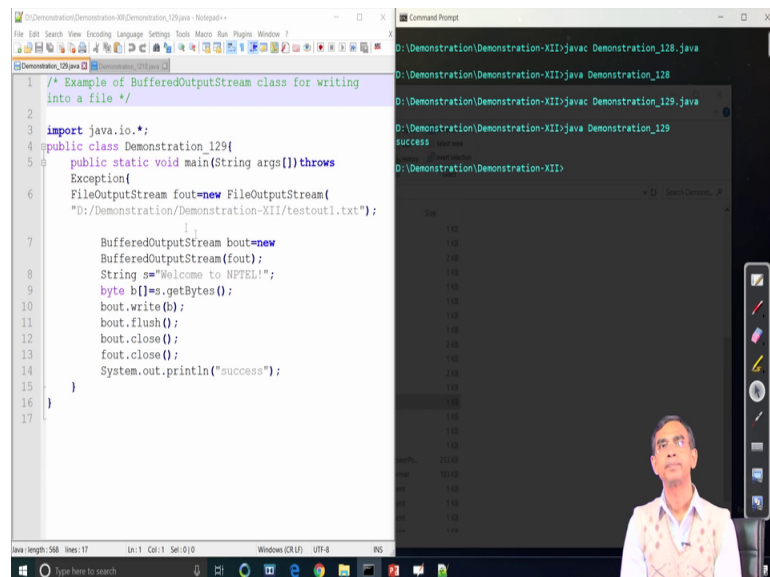
Now, here is a program as we see the same as a in file, and out file are the two file input stream and file output stream object. We create the connection in file, passing this is a target is the input source is the target that means, where the file will be stored there, now here is only mechanism it is difference then the previous one.

So, here byte read temporary store here as a byte in file read, it will read from the in file, store if the byte form, and it is the byte ok. And then by read is again while this is not equal to minus one mean, you have to copy scan the entire file. So, we have to go for

loop. And it read one byte at a time, and the same byte is write into the output file, and again go for reading the next, and then loop continue and until this one is there.

Now, input dat as you see welcome to NPTEL restored there. And here out dat file will be created now, if it is successfully copied from that file. So, mechanism is little it different, but the target objective is same object is that we are copying from file to another file yeah, file has been created out dat file we are going to display the out dat file yeah, this is the out dat file, you see this is the continue file that we have obtained it after successful compiling competition.

(Refer Slide Time: 32:37)



```
1 /* Example of BufferedOutputStream class for writing
2 into a file */
3 import java.io.*;
4 public class Demonstration_129{
5     public static void main(String args[]) throws
6     Exception{
7         FileOutputStream fout=new FileOutputStream(
8             "D:/Demonstration/Demonstration-XII/testout1.txt");
9
10        BufferedOutputStream bout=new
11        BufferedOutputStream(fout);
12        String s="Welcome to NPTEL!";
13        byte b[]=s.getBytes();
14        bout.write(b);
15        bout.flush();
16        bout.close();
17        fout.close();
18        System.out.println("success");
19    }
20 }
```

```
D:\Demonstration>javac Demonstration_128.java
D:\Demonstration>javac Demonstration_129.java
D:\Demonstration>javac Demonstration_129.java
success
```

Now, so this is the idea about, now before concluding our demonstration we want to have two more chance in programming illustration. This is the utility of buffered output stream class like also buffer output stream, buffer input stream. In our next two examples, we will demonstrate how the buffer output stream and buffer input stream is possible, it is basically same thing the same idea about byte stream actually, but you use another class that is there already java jdk call the buffered output stream class.

Now, let us have a quick look of this program here, we just create file output stream of this, because you want to read something to write into somewhere for the file output stream, we want to write something into test out one dot txt. This is our target file, where you want to store it from our programme, we want to put somewhere into this target.

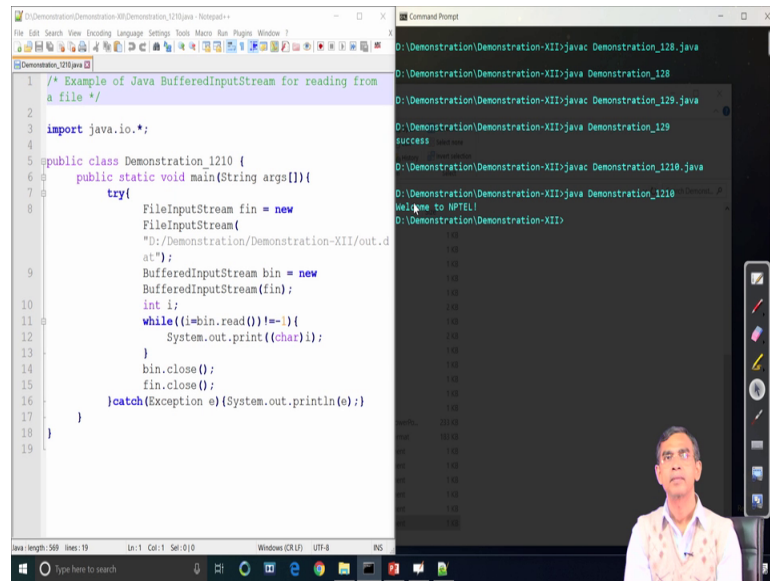
Now, let us see what is the program that you have to put string s welcome to NPTEL, this means that you want to write stray string, string into this one, but here we want to store it using this is the object the buffer output stream object call the b out. So, is basically b out is a buffer output stream is basically is a connection from the f out. So, the f out to this one that means, it will channelize the content. And then is stored in a buffered output stream form and that in this one.

So, here at this stream S will be from our program, and we convert in the string and here write b byte, and then we will store into the b out form. Now, this idea about that byte buffer output stream classes is there, and this kind of program although we are storing our programme. But, instead of this program if it is a network source at the moment, we do not have any network experience. So, we will not be able to do it with a networking example, but here this S can be forms a network channel.

Now, you will receive the buffer output stream from the network whatever the stream will come, we will buffer it first in our local buffer, and from that buffer we will push into the file target file. So, basically suppose you are downloading an image from the website, and then this image can to be a very large one, so we will buffer into. So, in that case we should use buffered output stream object, so that we can store and image as a buffer form instead of the entire image can be push there, because entire means too large maybe few MB, and you cannot put it there, so it is a buffer form. So, for this purpose, we will do it yes.

Likewise, buffered input basically if we read very large image from the channel network, and writing into a channel network like ok now, let us see here, how we can write this things into this one. And ok, so here the file is successfully content, here we can see directed our output to a file. But, here instead of directing this file, we can direct it into some network port, so that through network this can be transmitted. We will see about the network, whenever you will cover the networking in this module in this course.

(Refer Slide Time: 36:19)



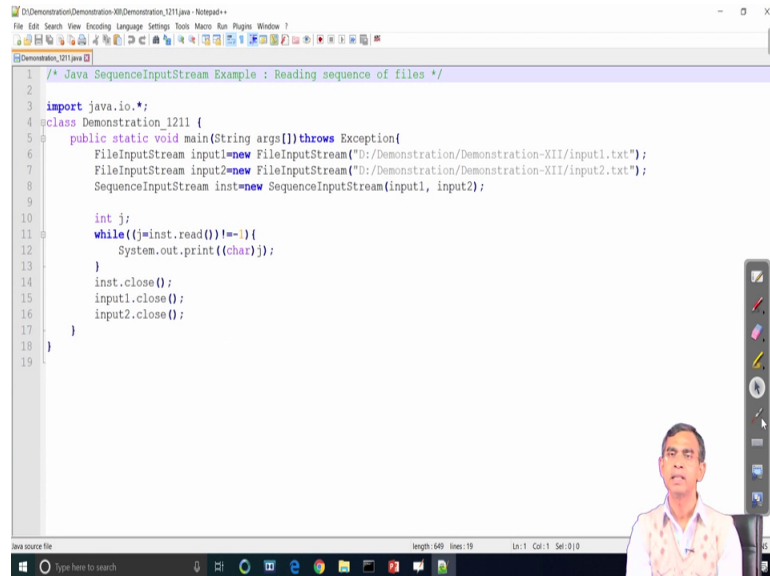
```
1 /* Example of Java BufferedInputStream for reading from
2 a file */
3 import java.io.*;
4
5 public class Demonstration_1210 {
6     public static void main(String args[]){
7         try{
8             FileInputStream fin = new
9             FileInputStream(
10                "D:/Demonstration/Demonstration-XII/out.d
11                at");
12             BufferedInputStream bin = new
13             BufferedInputStream(fin);
14             int i;
15             while((i=bin.read())!=-1){
16                 System.out.print(""+i);
17             }
18             bin.close();
19             fin.close();
20         }catch(Exception e){System.out.println(e);}
21     }
22 }
```

```
D:\Demonstration\Demonstration>javac Demonstration_128.java
D:\Demonstration\Demonstration>javac Demonstration_129.java
D:\Demonstration\Demonstration>javac Demonstration_120.java
D:\Demonstration\Demonstration>javac Demonstration_1210.java
D:\Demonstration\Demonstration>java Demonstration_1210
Welcome to NPTEL!
```

Now, the next example just opposite to this byte output stream buffer, here the byte buffer input stream class. Same problem here, basically here we just create the buffer input stream that means, we will read something. Here we will read again from this object out dot dot out dot dot is already stored there.

So, instead of out dot dot here, we can in this source we can maintain the network port number socket all these things. Here we will just read the output at in source of the input. And then it basically read the entire content, and then read in the form of a byte, and then print on the screen. So, it is basically the idea about buffer input stream, and then buffered output stream concept yeah this program as you see using the buffer input stream, we can access the file.

(Refer Slide Time: 37:19)



```
1 /* Java SequenceInputStream Example : Reading sequence of files */
2
3 import java.io.*;
4 class Demonstration_1211 {
5     public static void main(String args[]) throws Exception {
6         FileInputStream input1=new FileInputStream("D:/Demonstration/Demonstration-XII/input1.txt");
7         FileInputStream input2=new FileInputStream("D:/Demonstration/Demonstration-XII/input2.txt");
8         SequenceInputStream inst=new SequenceInputStream(input1, input2);
9
10        int j;
11        while((j=inst.read())!=-1){
12            System.out.print((char)j);
13        }
14        inst.close();
15        input1.close();
16        input2.close();
17    }
18 }
19
```

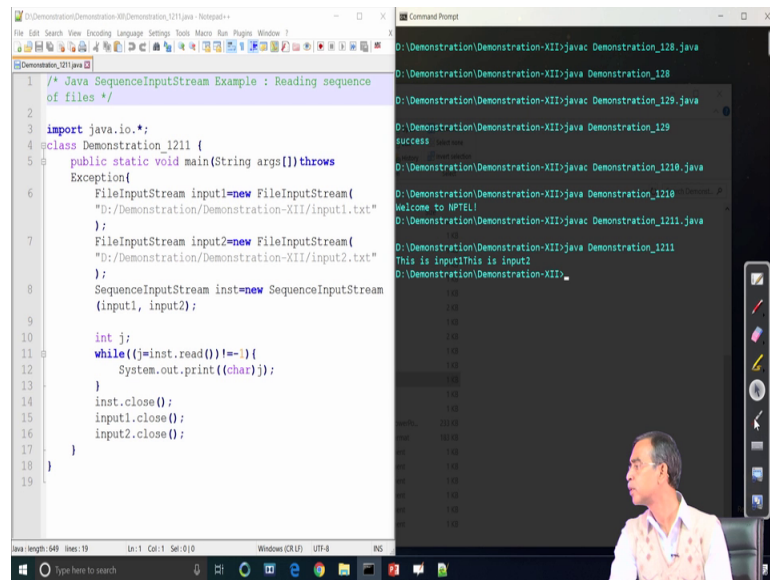
Now, if I ask you the same thing instead of buffer input stream using simple data input stream and output stream, you can do the same thing, you can do it actually you have done it already. Now, our next example to illustrate the usage of sequence input stream class. Now, so the sequence input stream class is basically can handle two more files together. And it will automatically manage it without any intervention within them.

Now, here is the program you see, first we create one input stream file input stream namely input-1 and another input stream-2. So, here input-1, and input-2 are the two input target source I can say. And these are two source namely input-1 dot txt, input-2 dot text assuming that there already present in the directory. Now, here you see we create an object call inst, which is of type sequence input stream, this class is again define in java dot io package.

So, we can create an object inst, now why we are creating objective you will see, how we can create object passing two input as a overloading constructor of this class input-1, input-2. Here also we can create input-1, input-2, input-3 also, it will take the overloading constructor will take it. Now, what it will do is basically input-1 and input-2 the two sources will be scanned one by one, and then the resultant total scanning output will be stored into the inst. And then that inst can be accessed I can read it, and then finally this inst can be display on the screen.

Now, here is basically inst, which is basically the result of sequence input stream, and then will be displayed on the screen. So, it will see the input 1 dot txt, and input 2 dot txt, the two content will be displayed on the screen. And finally, we close all the string in input-1, and input-2 from our system quick.

(Refer Slide Time: 39:31)

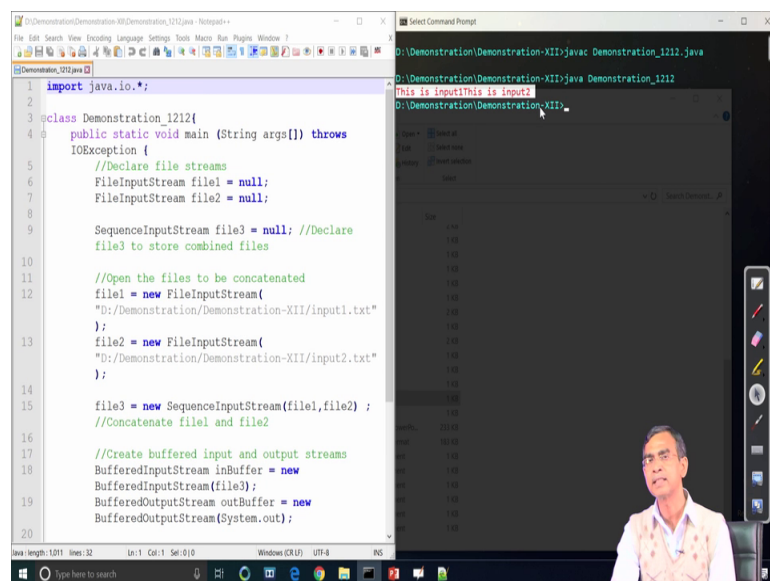


```
1 /* Java SequenceInputStream Example : Reading sequence
2 of files */
3 import java.io.*;
4 class Demonstration_1211 {
5     public static void main(String args[]) throws
6     Exception{
7         FileInputStream input1=new FileInputStream(
8         "D:/Demonstration/Demonstration-XII/input1.txt"
9         );
10        FileInputStream input2=new FileInputStream(
11        "D:/Demonstration/Demonstration-XII/input2.txt"
12        );
13        SequenceInputStream inst=new SequenceInputStream
14        (input1, input2);
15
16        int j;
17        while((j=inst.read())!=-1){
18            System.out.print((char)j);
19        }
20    }
21 }
```

```
D:\Demonstration\Demonstration-XII>javac Demonstration_128.java
D:\Demonstration\Demonstration-XII>java Demonstration_128
D:\Demonstration\Demonstration-XII>javac Demonstration_129.java
D:\Demonstration\Demonstration-XII>java Demonstration_129
success
D:\Demonstration\Demonstration-XII>javac Demonstration_1210.java
D:\Demonstration\Demonstration-XII>java Demonstration_1210
Welcome to NPTEL!
D:\Demonstration\Demonstration-XII>javac Demonstration_1211.java
D:\Demonstration\Demonstration-XII>java Demonstration_1211
This is input1This is input2
D:\Demonstration\Demonstration-XII>
```

As you see here the output as it is shown here, this is a input this is a input two one. So, actually this is the content in the input1 dot text and this is a contain input2 text. So, two files are sequential accessed one after another, and it is basically displayed on this one.

(Refer Slide Time: 39:49)



```
1 import java.io.*;
2
3 class Demonstration_1212{
4     public static void main (String args[]) throws
5     IOException {
6         //Declare file streams
7         FileInputStream file1 = null;
8         FileInputStream file2 = null;
9
10        SequenceInputStream file3 = null; //Declare
11        file3 to store combined files
12
13        //Open the files to be concatenated
14        file1 = new FileInputStream(
15        "D:/Demonstration/Demonstration-XII/input1.txt"
16        );
17        file2 = new FileInputStream(
18        "D:/Demonstration/Demonstration-XII/input2.txt"
19        );
20
21        file3 = new SequenceInputStream(file1,file2) ;
22        //Concatenate file1 and file2
23
24        //Create buffered input and output streams
25        BufferedInputStream inBuffer = new
26        BufferedInputStream(file3);
27        BufferedOutputStream outBuffer = new
28        BufferedOutputStream(System.out);
29    }
30 }
```

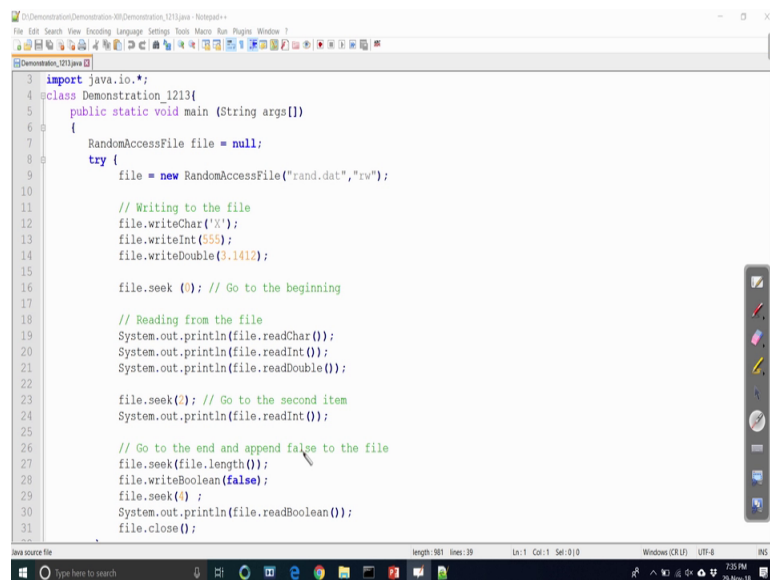
```
D:\Demonstration\Demonstration-XII>javac Demonstration_1212.java
D:\Demonstration\Demonstration-XII>java Demonstration_1212
This is input1This is input2
D:\Demonstration\Demonstration-XII>
```

So, this is the one usage of the sequence input stream class. Another one application of the sequence input stream class very same concreted margin. Here actually we (Refer Time: 39:58) not stored into the third file. When this program will basically tell you, how the merging result can be stored in the third file is the same as this one. And file1 and file2 just like input file1 input file2, and file 3 is basically the target third file is basically, we will store the concatenation of the two file.

We use the sequence input stream again file 1 and file 2 that means, file 3 will store the merging of the two content. Now, using the buffer input stream, I can read it and then display it. So, here basically in buffer, in out buffer is basically file3 and output is output buffer is basically displaying on the our standard output device.

So, it basically using the by buffer output in form and this program if it is data there, so the concatenation of the two file the concatenated march result will be displayed from the third file on the screen. So, this is a program that we have discuss about using the sequence input stream class example and here is the. So, this is the again program as you see this is a input this is the same thing that is concatenation of the two input file.

(Refer Slide Time: 41:17)



```
3 import java.io.*;
4 class Demonstration_1213{
5     public static void main (String args[])
6     {
7         RandomAccessFile file = null;
8         try {
9             file = new RandomAccessFile("rand.dat","rw");
10
11             // Writing to the file
12             file.writeChar('X');
13             file.writeInt(555);
14             file.writeDouble(3.1412);
15
16             file.seek(0); // Go to the beginning
17
18             // Reading from the file
19             System.out.println(file.readChar());
20             System.out.println(file.readInt());
21             System.out.println(file.readDouble());
22
23             file.seek(2); // Go to the second item
24             System.out.println(file.readInt());
25
26             // Go to the end and append false to the file
27             file.seek(file.length());
28             file.writeBoolean(false);
29             file.seek(0);
30             System.out.println(file.readBoolean());
31             file.close();
32         }
33     }
34 }
```

Now, we will discuss about random access, this is the last content to have the demonstration. Random access is just different mechanism the totally different, then the concept that you have learn so far. In some situation, we have to access the file in a random manner, whatever the things we have discuss that we discuss the file in a

sequence manner that means, one character the next character one by the next byte until the end of file likely, but here we can read at random.

So, this program can have a quick demo about it. And you can use the file class for this, so here we just random access file rather. There is a class random access file, which is defining java dot io file, we have to create an object for this. So, we create the file is that I random access file. And this file whenever we created, it can be open in any read write. So, other unlike sequence access, it can be read either read or write, but it can be open both mode.

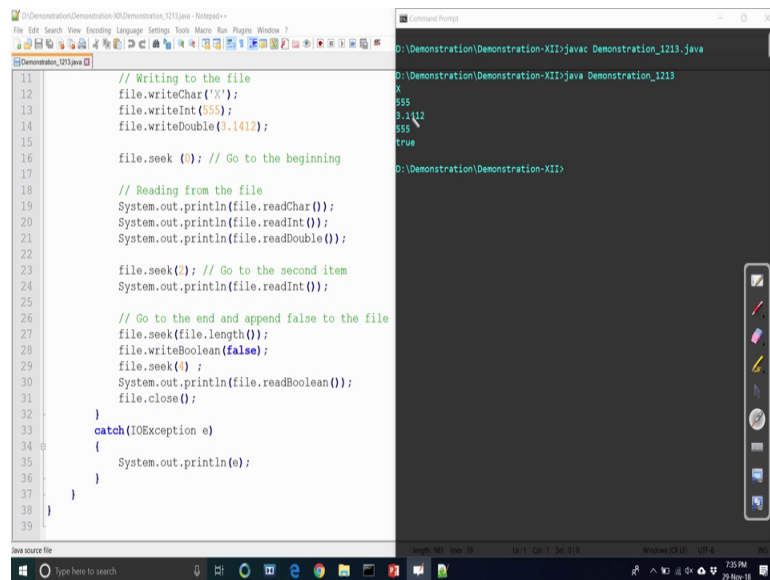
Now, here we create the file object and this is connected to the file. So, you want to randomly access rand data reading, writing on the same file at the same time like. So, here you see we first write something write character, write int, and write double, and these are the input that we have to write into this that means, if the file is blank or if the file is there if we write it from very beginning, it is basically overwriting. Then entire content will be deleted, and it will store this one.

And here is the file seek is basically positioning the file pointer, so it basically zero indicates that, it will position file pointer at the very beginning that means, where the x is now written there. Now, here system dot out dot println file read character that means, presently seek 0 that means start here. This statement will read x, and then again after reading x automatically file pointer move to the next one that means, read come here and read int, it will read this one. Go to the next one, read double, and it will read this one.

So, this is basically reading and writing from the same file as you can see there. Now, here again seek two as you see that file position will be move to the second location. So, 1, 2, then it will go there. And then again we read int that means, you can read the 555 here. And then file seek file dot length, so file dot length as you know, it will give you the size of the file.

So, if the seek is go there means, it will move the file pointer to the end of the file. And at the end of the file write Boolean false that means, we write a Boolean value and the false. And files seek 4, again we go to the forth position 1, 2, 3, 4 means, it will come here. Then we can file read Boolean, so read Boolean means it will false and then finally file close. So, this is basically shows a very quickly shows that how the random access mechanism can be applied to a file object.

(Refer Slide Time: 44:13)



```
11 // Writing to the file
12 file.writeChar('X');
13 file.writeInt(555);
14 file.writeDouble(3.1412);
15
16 file.seek(0); // Go to the beginning
17
18 // Reading from the file
19 System.out.println(file.readChar());
20 System.out.println(file.readInt());
21 System.out.println(file.readDouble());
22
23 file.seek(0); // Go to the second item
24 System.out.println(file.readInt());
25
26 // Go to the end and append false to the file
27 file.seek(file.length());
28 file.writeBoolean(false);
29 file.seek(0);
30 System.out.println(file.readBoolean());
31 file.close();
32
33 catch(IOException e)
34 {
35     System.out.println(e);
36 }
37
38
39 }
```

```
D:\Demonstration\Demonstration>javac Demonstration_1213.java
D:\Demonstration\Demonstration>java Demonstration_1213
X
555
3.1412
555
true
D:\Demonstration\Demonstration>
```

Now, here is a quick execution of the program, so that we can learn about it. And then you can see the difference state that it will show here as you see here, this is the first read integer write, read character, write integer, and then write double, and then again read, and then again finally go to the file position at the end, where the true is store and we read tribute and then get it. So, this way we will be able to access the file in a random way. This is a very small example, we have discussed about it.

Now, I hope you have understood the concept of file input output mechanism in java. Input output mechanism, it just started here. The more input output mechanism will be discussed, whenever we will discussed. The graphical user interface concept there is a lot of input output in a different fashion, the different style we have to follow. But, all those lesson that we have learned, we will be utilize there. In addition to this also few more in those context, we discussed ok, we should wait for the next topic that we are going to cover, it is basically graphical user interface programming our next topics to be covered.

Thank you very much, thanks for your attention.