

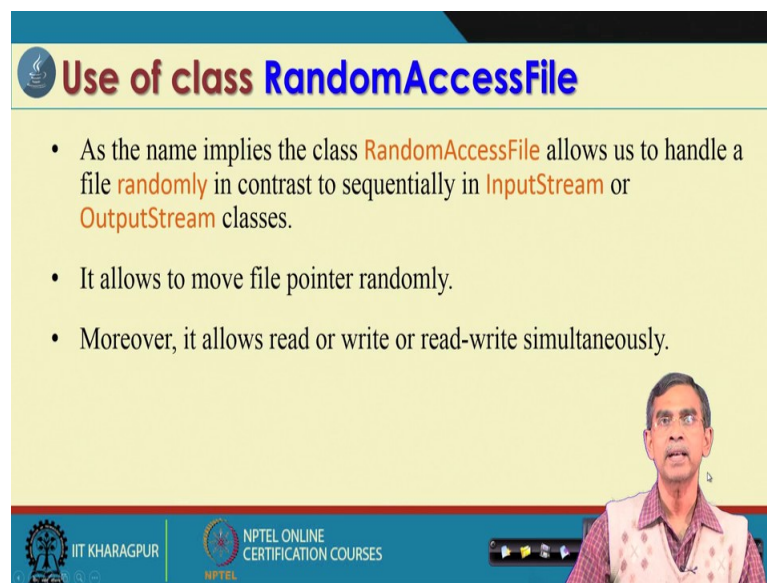
Programming in Java
Prof. Debasis Samantha
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 32
I-O Stream – III

So, we are discussing about input-output stream concept to handle the different ways of inputs and outputs to and fro from java programs. So, far we discussed mainly the two categories; one is called the bi stream class concept to handle the input-output and then character stream classes for handling the same. Apart from these two things, there is one more concept; it is called the Random Access File mechanism. So, by the name Random Access, we can understand that, in case of the other stream that we have learned so far, they are sequential in nature.

Sequential in nature means that you can read one item and then only the next contiguous item can be. It is not possible to change the file location or we can say more precisely, a pointer of the file from one point to another point randomly. So, this way, these are the two things are different. So, in many situations, the sequential stream access is what for us, but there are some situations where we need to access the file in a random way. So, our today's topic includes the Random Access File mechanism in a program.

(Refer Slide Time: 01:57).



Use of class RandomAccessFile

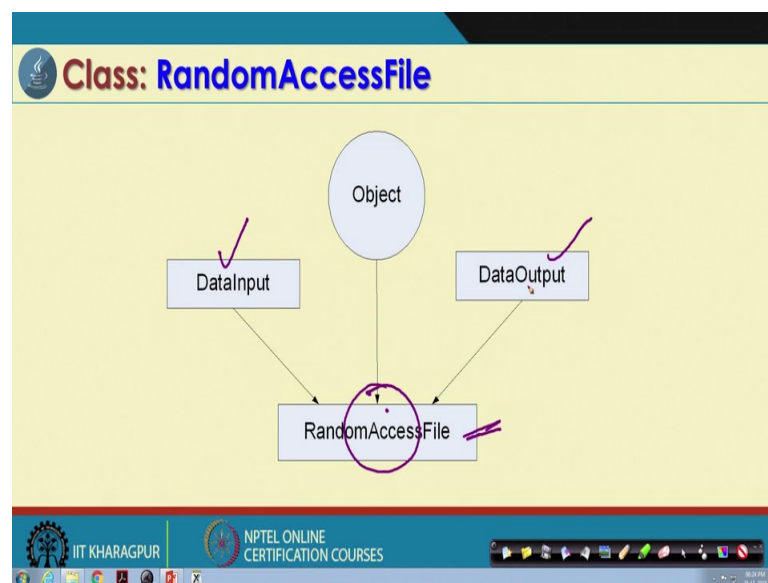
- As the name implies the class `RandomAccessFile` allows us to handle a file **randomly** in contrast to sequentially in `InputStream` or `OutputStream` classes.
- It allows to move file pointer randomly.
- Moreover, it allows read or write or read-write simultaneously.

The slide features a video inset of Prof. Debasis Samantha in the bottom right corner. The footer includes the IIT Kharagpur logo and the text 'NPTEL ONLINE CERTIFICATION COURSES'.

Now, as the name implies, so the random access implies that we can move the file pointer according to our own requirement. So, this is the main difference between the stream classes; like `InputStream` and `OutputStream`, that we have already learned then the Random Access mechanism. And another important difference between the sequential stream versus this random access is that, in case of all the classes that we have learned so far the sequential stream is concerned, they are basically, can be opened in only one mode; either the read mode or write mode.

So, in case of reading mode, it basically used as an input and in case of write mode it is used only for the output, but random access can provide us to do the things together, read and write simultaneously. So, we will see exactly how this random access file mechanism works for us in the java program.

(Refer Slide Time: 03:06).

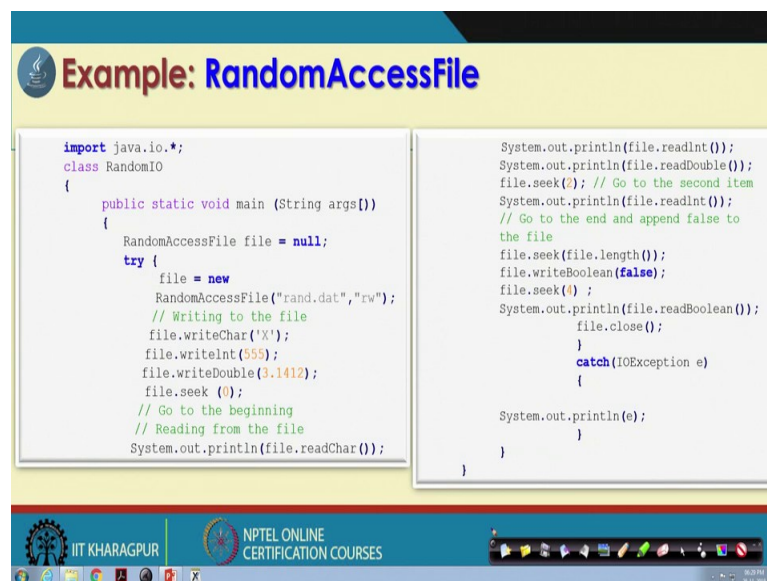


Now, so far the `RandomAccessFile` is concerned, the file system is very, the class mechanism is very simple. Here we see, this is the class called the `RandomAccessFile`, it has only two types `DataInput` and `DataOutput`. So, if we create an object of the types say `RandomAccessFile`, then these object can take the input and output. So, this is the concept there, so it is very simple. Now, we will see by means of an illustration only, how this file, this class works. Actually, the methods those are there so far reading and writing are concerned, is basically the same here, in this case also.

So, I do not want to repeat all the method. For example, read, readInt, readDouble, write, writeDouble, like this, all those methods are there and again I want to mention is that using this Random Access file, we can write or we can read in any form, whether it is a byte form or character form. So, basically we can say a Random Access File, concept is a general, overall file concept, it can be used either read or write mode. It can be used to access the items in a byte stream form or a character stream form like this and it is also simultaneous both input and output it is there. So, it is basically a more generalized form is there.

So, some people can work only if the Random Access File to do it, but so far other, I mean speed and all these things are concerned or another input mechanism is concerned, it is basically another input stream is better than Random Access File. It is usually good or easy to learn to programming, but so for the application point of view, other stream classes are more preferable than this one.

(Refer Slide Time: 05:09).



Anyways, so let us see, let us have an example and we have to watch this example a little bit carefully, because I did not mention about what are the methods are there. With this illustration only I want to point out what are the different methods are there. Now, this is the one program that I have planned for you to explain how the RandomAccessFile works or how you can use a Random Access File in your program.

So, here I first create one object of the type `RandomAccessFile`. So, here is basically if you see, this is the class and this is the object that I have created this is the object of the type `RandomAccessFile`. So, I have just declared an object, the name of the object is, in this case, is `file` and this `file`, this object is of type `RandomAccessFile`. I think this is clear ok.

Now, let us see I just establish a connection from this object `file` to one file, which is stored, there in the memory. Now, here you see our next statement is for this one. So, this statement `file equals new RandomAccessFile`, we pass this one. Now, this is the name of the file, which we want to use as a `RandomAccessFile` and here you can see, it has the two-parameter. So, for this constructor is concerned and this is called read-write. This means that we can use, access this as a read or write.

Now, instead of `r w`, if we write only `r`, then this file will be opened as a read mode or also we can access this file as a `w` mode, only then in that case this file will be opened as a write mode. So, in this case we opened the file as a read-write mode. Now, another one important thing is that this is the file we are assuming that already is available, if the file does not exist then; obviously, using these things a file will be created automatically for you. Again if a file already exists; this file can be created using any either data in, output stream or read buffer, write buffer concept character stream classes.

So, whatever the way, this file is there absolutely, this is not an issue. It is not that this file should be created using the Random Access File concept, whatever the mechanism if the file is created. Even this file also, can be created using your editor like; notepad++ or any other things also, even this file can be an image file, this file can be a video file also, absolutely this is not the issue there. Anyway, we assume that a file already exists and let us, let us give the name of the file as `rand.dat` ok.

So, with this statement we can create a connection, the connection to the actual file physically which is the file and name of this object, which basically connects from our program to this one are called `file`. Now, here we can see a few statements what we can do is that `writeChar`; this is the one method and then the argument is `X`, this basically the ASCII value of the character `X`. So, what is the idea it is that, this method `writeChar` will write this `X` into the file object.

Now, where it will write? It will write there, where the file position is currently located. In the beginning, whenever we create this file, it is that default, the file location is at the very beginning; that means, at the 0'th position what we can say. So, this means that this character X will be stored in the 0'th location of the file.

Now next, now we write another item called the writeInt. So, the writeInt basically, write these value as an integer and write into the file. So, where it will be written; after the X, exactly. So, if it is 0, then the position of this value will be in the oneth location. Next, we see, we again write into the same file objects and is a double. So, this is a double and it will write there. So, after this writing 3 elements into the file, this goes to the 0 element, this go to 1 and this go to 2. Now let us see, there is another method called the seek method. So, what the seek method is basically, this a very useful and very is important one method. This method basically allow a programmer to move the file pointer at any location where the programmer wants to do that. Now, so seek method is basically is seek or positioning the pointer in a location. Now, this parameter is very important. So, a parameter is basically, stated where we want to place the file pointer location right.

Now, here file seeks 0, this means that we want locate the file position at the 0'th one. So, in this case, 0 1 2; that means, file pointer will locate at this position, where the X is written there in the file, this file is rand.dat. So, it is basically, in other words, file so seek 0 we can say that file pointer will locate into the very beginning of the file. So, this way we can move the file pointer, we will see about positioning the file pointer more. Now, here again, see, so this is the 0th 1 2 and here System.out.println file readChar.

So, which character it will read because we have already positioned our file pointer at 0'th location and at the point file.read character, this actually this. So, file.read character means it will read the character at the 0'th location Now, so it will read basically X and here again once it is read automatically, file pointer then will move to the next location.

So, here again, if we issue the next, statement like file read.Int then it will basically read this one again, after reading this integer it will go to the next location; that means, it is where this value is stored, so it will go here again. So, this one will read to the next one. So, this will read the element which is the 0'th position of the file this will read the element, which is first position, then second position and so on. Now, so this way we

will be able to move the file pointer automatically. Now, again here if you want to move the file pointer explicitly then we can again issue the seek command. So, here seek 2, this means that file pointer will move 0 and 1. So, this location goes to the second item.

So this basically indicates that the file pointer will go to the first-second item; that means, these are the second item ok. So, this way we can move it and then if we readInt it will read the integer value there. Now, here again we can move our file pointer randomly and as we move it and from that pointer, we can read any value there. Now, whatever reading is concerned there are little bit formalities that you have to mention. Suppose, at a particular location we have stored an integer and you want to read as this is a double, then actually it will result in an error. So, this would not be correct actually. So, if it is an integer pointing it, so you should read that value as a read proper read method.

So, no other method is allowed if you do it then this will leads to an error that is; obviously, should be handle the way with exhibition handling mechanism Now, so we have learned about how to write something into a Random Access File, also we have learned about how to read something from the file and also we have learned about how the file pointer can be moved across the file. Now, here again, there is a many more so far the seek is concerned here if you say file seek file.length. So, this basically, whatever the size of the file at the movement, file.length will return this one and then it seek means; it basically says that the file position, file pointer will be positioned at the very end. So, this basically seek this one, it basically places the file located at the very end of the file.

And once this is again writeBoolean, so again we are writing something in the form of bBoolean and this is the value that we want to write, so false. So, it will basically write at the very end of the file and file seek 4 is basically position into the fourth item in the file. So, it is the file seek four and this is a usual method is that a close; that means, once every operations are over then you should close this file and then all things would be put as we have mentioned here, under the try-catch block to catch any exception, if any occurs while you are either opening the file or closing the file or you are reading or writing into the file or from the file. So, that is why the try-catch block is a must, if we do not put the try-catch block so the compiler will report an error, you will not be able to run the program actually. So, this is the mandatory thing that you should do.

So, this is the idea about the random access file. I hope you have learned that how the random access file to be managed and what are the methods are there very simple and whatever the way that the file can be opened and it can be closed is very simple. It is, as usual, the either, the earlier method and we have already understood about this one. Now, likewise also we have considered here reading integer format or double format or character format, write way also you can read or write into the other format like byte format uita format and many other formats absolutely it is more flexible. It will allow you to do whatever the way you want to do.

(Refer Slide Time: 15:34).

Example: Appending to a RAF

```
import java.io.*;
class RandomAccess
{
    static public void main(String args[])
    {
        RandomAccessFile rFile;
        try
        {
            rFile = new RandomAccessFile("city.txt", "rw");

            rFile.seek(rFile.length()); // Go to the end
            rFile.writeByte("MUMBAI\n"); //Append MUMBAI
            rFile.close();
        }
        catch (IOException ioe)
        {
            System.out.println(ioe);
        }
    }
}
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And here is another example that you can see, it is basically just example that ok, it can also be used as a byte stream mechanism like byte stream way and let us see the program here and this is the again, I am just creating one object rFile of the type RandomAccessFile in this program. And so, rFile is the object that is to be operated using the Random Access File concept.

Now, I am just creating this file object, first establish a connection, let the city.text is the file to be operated using random access file. We create this object using read-write method. So, you declare rFile and once the object is instantiated. So, this is the completion of the object rFile, in the mode of RandomAccessFile.

Now, here we can see rFile seek rFilelength, you understand that what it does mean; it means that whatever the location, it will go to the beginning give, go to the end of the

file. Now, in this case at the opening, so if there is no content, so it will be at the beginning is the same at the opening. Now, here if suppose city.text already has certain data. So, by means of this seek method `rFile.length`, it basically goes to the file pointer at the end of the file. So, that is why we have here, assuming that this file already has certain data and then `writeByte` as you know it is just like earlier; `writeInt`, `writeDouble`, `writeChar` like this one, it is basically `writeByte`. So, this is the value that it will write. So, it is in the byte form.

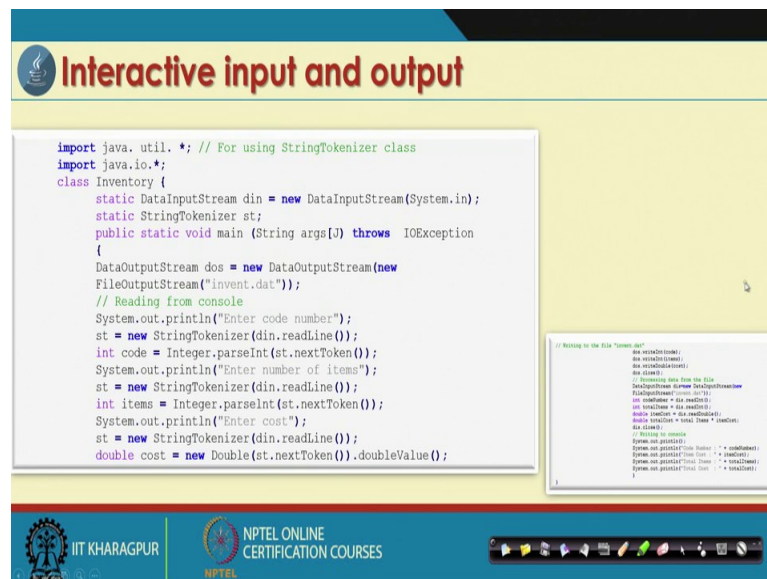
So, automatically java pro system, java run time manager will convert this string into the byte form and then that value will be appended at the end of the `rFile` and finally, we close the `rFile` and as I told you everything the operation whatever it is there should be put under try-catch. So, I just put under try-catch. So, in that case, the general exception that may occur is basically, input-output exception. So, it is called the `ioe`.

So, this the way that random access file can be used, actually it is very much similar to whatever the method other classes that we have already discussed for byte stream as well as for character stream classes. So, that is why I do not want to discuss much more about this one, more about this random access file will be able to learn when we will discuss about the demonstration giving some more programs in this regard. So, this is about the random access file concept.

Now, whatever the input-output stream we have learned so far, mainly byte stream character stream and then random access, whatever it is there you can use them and you also use them in an interactive input-output command; that means, you can do one thing, you can ask the common. The right way, the general program is that what is your input, you can give a prompt to the user, user will understand about what is the input to be, for example, enter your name. So, the user can understand that ok, in this case I should enter the name, you can then enter your name within 20 character. So, the user understands that the user has to enter a stream of the length of less than 20 characters like this one. So, this is basically called the prompt should be given, according to the prompt user will enter something and then accordingly the data will be collected, whether these data can be collected from the standard keyboard or can be from the file or can be from other network channels, whatever it is there, it is called the file.

So, this the idea about the interactive one, I am just going to give a very simple one illustration about interactive input-output using the concept of data output stream class. So, here the idea is that it will give a prompt, what to do; the user will accordingly work on that, given action reply to the action and then accordingly the program will write it. So, let us have a quick demo about this program here.

(Refer Slide Time: 19:44).



Interactive input and output

```
import java.util.*; // For using StringTokenizer class
import java.io.*;
class Inventory {
    static DataInputStream din = new DataInputStream(System.in);
    static StringTokenizer st;
    public static void main (String args[]) throws IOException
    {
        DataOutputStream dos = new DataOutputStream(new
        FileOutputStream("invent.dat"));
        // Reading from console
        System.out.println("Enter code number");
        st = new StringTokenizer(din.readLine());
        int code = Integer.parseInt(st.nextToken());
        System.out.println("Enter number of items");
        st = new StringTokenizer(din.readLine());
        int items = Integer.parseInt(st.nextToken());
        System.out.println("Enter cost");
        st = new StringTokenizer(din.readLine());
        double cost = new Double(st.nextToken()).doubleValue();
    }
}
```

```
// Reading from the file "invent.dat"
DataInputStream din = new DataInputStream(
    new FileInputStream("invent.dat"));
StringTokenizer st = new StringTokenizer(din.readLine());
int code = Integer.parseInt(st.nextToken());
int items = Integer.parseInt(st.nextToken());
double cost = Double.parseDouble(st.nextToken());
System.out.println("Total cost = " + cost);
}
```

Now, here is the program as we see in this program, so we have created one class called Inventory and next statement you can understand, what we have done is DataInputStream class.

I have created one DataInputStream class here. So, this the DataInputStream class. For this class we have created object din; that means, this is the name of the objects and DataInputStream System.in, this means what? This means that we have created an object of the class data input stream and this object is now ready to read something from the System.in, here the System.in means that we want to read from the standard input. So, standard input means a keyboard. Now, here we want to read a particular item from the keyboard that will read it as a stream.

So, for these things in the java.util package, there is a class called stream tokenizer, it will basically read a stream as a token. So, that is why it is called the stream tokenizer, whatever the input, you write it is a general form of giving input or treating input, not that integer, not that character not that double not that other thing. So, whatever you give

it, it can be considered as a stream and then stream tokenizer and then return it and then, later on, we can change into the format, whatever the way we want. Now, here see how we can do all these things here.

So, we first create an object called StringTokenizer and this is the name of the st. So, basically the temporary buffer, in the earlier example we use a singular kind of buffer to store a string value. So, similar same stream-like. So, it is like that. Now, this is the main method. Now, in this main method, we create another stream called DataOutputStream; that means, whatever the things will read from the standard keyboard, it will write into this file, so invent data. So, this is the file where we want to write something; that means, upon reading something from the standard input, we want to write into a file. It is not that we are writing into standard output or display.

So, in that case so, so, this the two objects; one DataInputStream objects and DataOutputStream objects. The two objects are created for input stream and output stream; that means, reading form and then writing into. Now, the next few statements is basically given a prompt here enter a code number. So, then once the code number is there then these a readLine; that means, whatever the data it is entered from the keyboard, the entered data will be read from the that source and so, this is the din is basically, the input object it is there; that means, it will read from the standard input and read a line and then StringTokenizer will take this value and then change into the string form and store into st.

So, basically it reads whatever you enter the code number, is basically may be digits and whatever it is there, but it will store at a string. Now, in the next statement as we see st.nextToken. So, it is basically read the next items, whatever it is there and then integer.parseInt; that means, it will change whatever the string that you have written into an integer and will store in code. So, basically code is now whatever the user enters, it stores as an integer into this form. Now, again enter a number of items. So, again another input the user will provide from the keyboard, it will read from the keyboard and store into st and again st.nextToken.

It will basically read this token, convert into integer and then store it into items. So, here we can see that code and items are read from the keyboard then again another input enter cost. So, again StringTokenizer din read, it will read and here we have to convert in the float. So cost, so in this, so far we have read three items; code as an integer, items as an integer and cost as a double. So, once these are the three values that are collected from the keyboard, our next is that how we can push it into the output object namely the dos; that means, how we can write these values into a file. So, here is the method of how we can do it.

(Refer Slide Time: 24:21).

The slide is titled "Interactive input and output" and features a Java code editor with two panels. The left panel shows the beginning of a program that uses a StringTokenizer to read input from the keyboard. The right panel shows the continuation of the program, where data is written to a file named "invent.dat" and then read back from the file. The code uses DataInputStream for file operations and System.out for console output. The bottom of the slide includes logos for IIT Kharagpur and NPTEL Online Certification Courses.

```
// For using StringTokenizer class
import java.util.*; // For using StringTokenizer class
class Shopping {
    static DataOutputStream dos = new DataOutputStream(System.out);
    static DataInputStream dis = new DataInputStream(System.in);
    static String code, items, cost;

    // Reading the code
    System.out.println("Enter the code:");
    code = dis.readLine();
    // Reading the items
    System.out.println("Enter the items:");
    items = dis.readLine();
    // Reading the cost
    System.out.println("Enter the cost:");
    cost = dis.readLine();
    // Writing to the file "invent.dat"
    dos.writeInt(Integer.parseInt(code));
    dos.writeInt(Integer.parseInt(items));
    dos.writeDouble(Double.parseDouble(cost));
    dos.close();

    // Processing data from the file
    DataInputStream dis2 = new DataInputStream(new
    FileInputStream("invent.dat"));
    int codeNumber = dis2.readInt();
    int totalItems = dis2.readInt();
    double itemCost = dis2.readDouble();
    double totalCost = totalItems * itemCost;
    dis2.close();

    // Writing to console
    System.out.println();
    System.out.println("Code Number : " + codeNumber);
    System.out.println("Item Cost : " + itemCost);
    System.out.println("Total Items : " + totalItems);
    System.out.println("Total Cost : " + totalCost);
}
```

So, you can see this is the next part of the program as you see in this program dos is basically, the objects for output stream and we just write all the values that we have written according to their format. For example, the first item, the code read as a writeInt, the second item also writeInt, the third item also writeDouble. Once all values are written, we close the output stream. Now, so, this is basically the way how the output-input, I mean taking using prompt from the keyboard collecting this one and then finally, writing into to there.

So, this kind of usually application is many more frequent and common, whenever you have to develop any application line. Now, again here what we are doing you see, see invent.data is already created by storing these on the file, we again create it; that means, we want to see we want to read it again, whether this. Now, this is again creating another

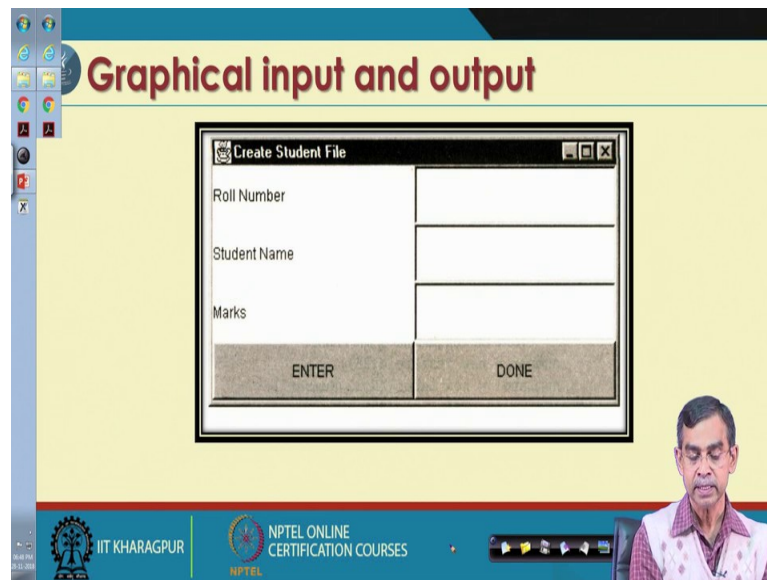
dis invent.dat. Now, here also there is another way that we can do it. So, code number we are reading as a readInt directly from this.

So, readInt and the readDouble and then we also calculate total cost and close this one and once read is complete we again complete it. So, this is basically another way. So, earlier is basically use the StringTokenizer, here we are showing the different ways how we can read directly from this one, no stream form, it is basically if we read, if we know the format directly, we can use it like this one. So, these are the common method that is there in this class DataInputStream.

So, these are the second method or alternative approach to reading the values interactively and then once the reader is there we just disclose means closing the stream and then again the same thing, we are doing here writing it. So, is basically whatever the things that we read it into the DataInputStream same is basically displaying into the standard output. Here, the standard output means the display. So, it basically displays all the value that we have read it there.

So, we have done about the interactive way of input and output that is possible in java program and now we are going to discuss one more interesting application of this is (Refer Time: 26:54) called graphical input and output concept. Now, what is the idea about that ok? This idea is very simple and regarding these things actually our next part of the lectures next few modules, actually we have planned to do it, but will give just a simple introduction to this concept. It is very simple to understand at this movement although, many more things we have to learn while we will progress further.

(Refer Slide Time: 27:17).



So, the idea about this graphical input-output is can be better explained with this example here. So, this is one a window and we can say it is basically the window for text entry things or we can say it is a frame or form. It is just like an applet actually, we have some idea about the applet only. Anyway, so this is also we can consider an applet here. Now, if we see this is an applet, it has many items in it. So, items mean all the applet items that we can classify into their; here this is the one item, we call it is a button and this is another button.

So, these applet has two buttons at the bottom, the buttons are label as entering and done. So, two buttons other then two buttons it has also another three elements; here this is the element, which is called the text entry field and this text entry field means it is the area, where the user can type something and this is basically a label for this text entry field. So, similarly their second text entry field and the label is this one and these are third text entry field, these are label ones. So, in this basically applet, there are actually, if we consider it as a grid so, there are 4 rows are there and 2 columns are there.

So, it is basically 4 cross 2 grid actually, we can see. Now, so the out of this 4 cross 2 grids right, there are 3 text entry fields and two buttons, this is very simple one idea about it there. Now, so this once these windows are displayed to the user, then the user can type the, his roll number can name and then marks one, all the values are entered

then if the user can click done. So, this means whatever the input that you have entered it will be collected and then once it is collected, it should be stored somewhere.

So, that can be stored in any output stream classes or in a random access file or in a character stream file whatever it is there. So, this our mechanism that I am going to explain to you that how graphically the user can enter some input and then some, the same input can be collected and finally, the input can be stored into the program. And this program can be repeated whatever the entries user wants to do and once the user wants to finish it so, this will select the done button. So, enter button means enter the record sorry and then done button means when the user wants to exit from the entry system. So, this is the concept, it is here.

Now, let us have some a program regarding these things how this program; this program also many things we should assume it and whenever we will follow applet and then many other concepts, then all those things will be clear clearly understandable. Anyway so, just for the sake of interesting point only, I just want to have included this discussion.

(Refer Slide Time: 30:25).

Graphical input and output

```
import java.io.*;
import java.awt.*;
class StudentFile extends Frame
{
    // Defining window components
    TextField number, name, marks;
    Button enter, done;
    Label numLabel, nameLabel, markLabel;
    DataOutputStream dos;

    // Initialize the Frame
    public StudentFile()
    {
        super("Create Student File");
    }

    // Setup the window
    public void setup()
    {
        resize(400, 200);
        setLayout(new GridLayout(4,2));

        // Create the components of the Frame

        number = new TextField(25);
        numLabel = new Label("Roll Number");
        name = new TextField(25);
        nameLabel = new Label("Student Name");
        marks = new TextField(25);
        markLabel = new Label("Marks");
        enter = new Button("ENTER");
        done = new Button("DONE");

        // Add the components to the Frame
        add(numLabel);
        add(number);
        add(nameLabel);
        add(name);
        add(markLabel);
        add(marks);
        add(enter);
        add(done);

        // Show the Frame
        show();
    }
}
```

So, this is the one program, I will briefly tell about what exactly the program is doing for us. So, in this program here we can see first, we declare one class called StudentFile Frame and is extend frame, this frame is basically is an AWT package which is not yet discussed, we will discuss of course. So, this basically one class, which basically allows

you to create a frame actually. So, the last window that I told you, it is basically is an example of a frame with 4 cross 2 grids.

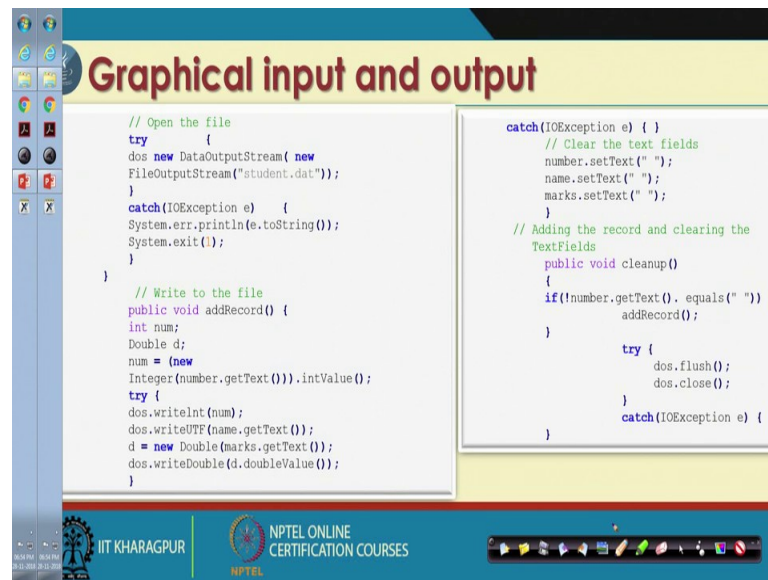
Now, so we create two things; first TextField, there are 3 TextField number, name and marks. They are basically student roll number, name and then marks are needed to be stored here and these are two buttons as I mean to enter and two-button that we have already told you and these are the label. So, numLabel is basically for the number and nameLabel is for the name and marks for the marksLabel. So, these are the different variable that is required in this frame and finally, data output stream dos is basically we declare the stream objects, which is basically for storing the data permanently in a file.

Now, here this one constructor as we have declared here, StudentFile is a constructor, this constructor and then these this basically create StudentFile. So, basically it just, it will basically give the idea about that this the create StudentFile is the menu there and then the setup is another method that we are going to discuss shortly. This is basically the setup method, we will create whatever the frame, it is there allow user to enter the values and then collect it and store it. Now, resize is basically sizing of the applet, how big you want to have it, setLayout as you see it is called the grid.

So, there is actually a mechanism using GridLayout 4 cross 2 grid; that means, a total of 8 total 8 elements can be placed in the applet this is the idea about. So this is the total way of creating the way applets in our program and once the applet is ready, then we have to place the different items here. So, these are the few steps as we see, this basically for the number TextField size is 25 and these are label is roll number. So, this basically the first TextField for displaying on in the applet and this is for the second TextField and this is for the third TextField. So, this is the idea about the three TextField and finally, there are two buttons having labels enter and done is to be placed there. One all these items are created this is called the different components in this applet created we have to add the component into the applet.

So, these are the common for adding all the components everything into that. So, they will automatically fix in the apple applets and finally, the applet will look the way as you have already shown it to you. So, this way and then finally, the show method is basically all once, all items are been placed into the applet it will basically display as an user. So, this is basically as a part of the unit method those are there in applet actually.

(Refer Slide Time: 33:50).



Now, once this applet is ready, therefore, once this applet is ready, what we have to do is that we have to add the records one by one. So, for these things what we have? So, DataOutputStreams already you have created and then dos are the object for that and let us see student.data is the file, where you want to store the data and then here is basically how we can, add the record into the file. So, add record basically, these are the temporary value numDouble d. So, num is basically collecting value from the TextField area as an integer value.

So, this is just users syntax is that int value, it will read from the getText and then a number. It basically converts in integer and finally, stored as an integer and then it basically dos writeInt. So, num will be read and then dos writeUTF name. So, it will read from the TextField area, name and is a writeUTF and then store it and then finally, it will read the double, the marks and read at the value and write this value into the objective dos and here is basically once a particular record is entered. So, this will basically cleanup. So, this is the method that cleanup will do that. So, add record, once the record is added so it will clean up and then finally, flush and then close.

(Refer Slide Time: 35:16).

Graphical input and output

```
// Processing the event
public boolean action(Event
event, Object o)
{
    if(event.target instanceof
    Button)
    if(event.arg.equals("ENTER")) {
        addRecord();
        return true;
    }
    return super.action(event, o);
}

public boolean handleEvent (Event
event)
{
    if(event.target instanceof Button)
    if(event.arg.equals("DONE")) {
        cleanup();
        System.exit(0);
        return true;
    }
    return super.handleEvent(event);
}

// Execute the program
public static void main (String args[])
{
    StudentFile student = new StudentFile(
    student.setup();
}
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, the output system will be closed. So, this is basically the skeleton of the program and finally, once it is done we have to call the method as a final method, but they're actually two events is required one action event object o, it is basically to deal with entering, whether it click DONE or click exit it will do there. So, it is basically two events for entering and another event for done the (Refer Time: 35:33) button event. So, regarding the button event we will discuss more in details whenever we will discuss about handling applets. So, this will be done and finally, it will come to the main method.

So, the main method will just simply create this object StudentFile and it call the setup method, which we have already discussed there; that means, this will display the total applet and then ready for your work a user will enter as long he wants, till he finish button, click the button DONE and then go on entering and for each entry, there will be entered click, it will go to the next cleaning their TextField area and so on. So, this is a very simple program which is simple, but yet effective to develop any, what is your text entry system or a data entry system in any application.


(Refer Slide Time: 36:24).

Another graphical Input/Output

```
import java.io.*;
import java.awt.*;
class ReadStudentFile extends Frame
{
    // Defining window components
    TextField number, name, marks;
    Button next, done;
    Label numLabel, nameLabel, markLabel;
    DataInputStream dis;
    boolean moreRecords = true;

    // Initialize the Frame
    public ReadStudentFile()
    {
        // Setup the window
        public void setup()
        {
            resize(400,200);
            setLayout(new
            GridLayout(4,2));
            // Create the components of the Frame

            number = new TextField(25);
            numLabel = new Label ("Roll Number");
            name = new TextField(25);
            nameLabel = new Label ("Student Name");
            marks = new TextField(25);
            markLabel = new Label("Marks");
            next = new Button("NEXT");
            done = new Button("DONE");
            // Add the components to the Frame
            add(numLabel);
            add(number);
            add(nameLabel);
            add(name);
            add(markLabel);
            add(marks);
            add(next);
            add(done);
            // Show the Frame
            show();
            // Open the file
        }
    }
}
```




So, this is about the graphical input-output concept I just do not want, do not want to discuss this one.

(Refer Slide Time: 36:31).

? Question to think...

- How Java helps programmers to develop GUIs?
- How one can develop programs like Google?



So, we have planned about the graphical user interface concept. There are many more sophisticated programming in this area of a graphical user interface, which we will discuss in our next module.

Thank you.