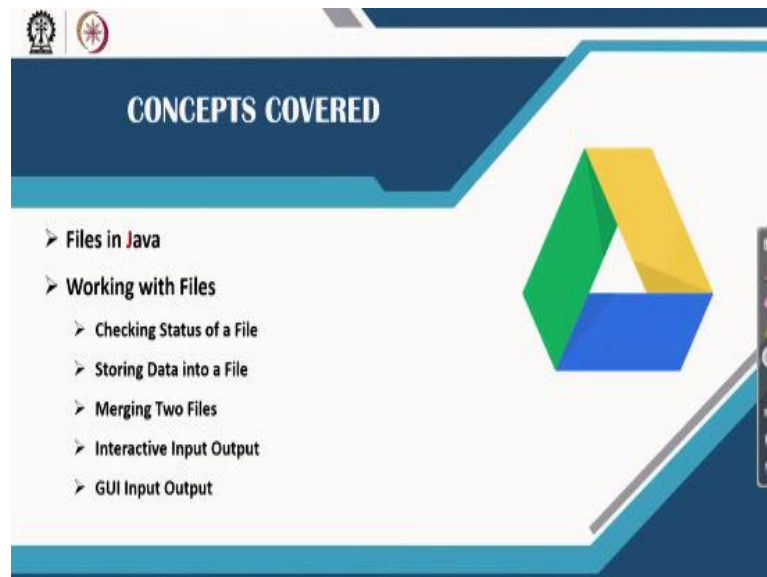


Data Structures and Algorithms Using JAVA
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 45
File Input-Output

In this video, we will discuss one very common things which many programming language usually provides. JAVA is now a exception, JAVA also support this facilities it is called the file handling. So, in this program we will discuss about file input-output. Now we are some way already familiar to handling files while we are discussing about that JAVA IO in the JAVA dot IO package and they are basically different way the input stream and output stream can be handled. And out of the different sources of input data and then output destination file was considered one important there.

In that context we have already studied about the file, but file needs to be studied more details because it has some many other things to be considered which is not possible to consider there in the stream handling actually or JAVA IO stream as a JAVA IO stream. Now for this purpose, in JAVA.IO package there is one class has been defined, this class is called the file, but there are file two types. One is called the sequential access file and another is called the random access file. So, first we will discuss about the sequential access file and then random access file will be discussed in another discussions.

(Refer Slide Time: 01:54)



So, today's topic which we want to include here basically the file concept in JAVA. File is for the permanent storage of data actually if you want to store some data in a permanent manner that usually use the file. Now regarding working the file obviously file cannot be created automatically, file cannot be created manually rather it can be created automatically whenever you just open a file if it is not available, then it will basically create the file actually it is the concept, but whenever open a file in the write mode it is possible.

Reading a file it does not that file will open automatically so that is not right. Now, so, file creating is a one concept, so it will automatically open when you want to write first time into the file starting from the null file actually otherwise, there are some some operations or management is required in order to have the different different different facilities of the file. I have listed few here so suppose you want to see the status of the file whether file is readable or writable or file is exist or not, so these are the difference or in which directory the file is available, what is the path name all those things is basically status.

Then next thing is the retrieving data either storing data into file or reading data from a file. Next is that merging the two contents, the contents from two files into a single another file so third file can be created taking the data from two or more file sources. So, it is there and then finally we will discuss about interactive input output, you read some data from the keyboard, store into file or you read some data from the file, do some processing, write back into another file or display the data after processing into computer.

It is called the interactive input output and GUI input-output is very common in many application where the graphical user interface will be there where you can just suppose you are booking a ticket online so you have to enter the data so the graphical user interface, the interface will be given to you from where you can enter the data and then you can submit the data will go where it will go? It will automatically store in a file. So, this is the GUI input-output we will see exactly how it is possible.

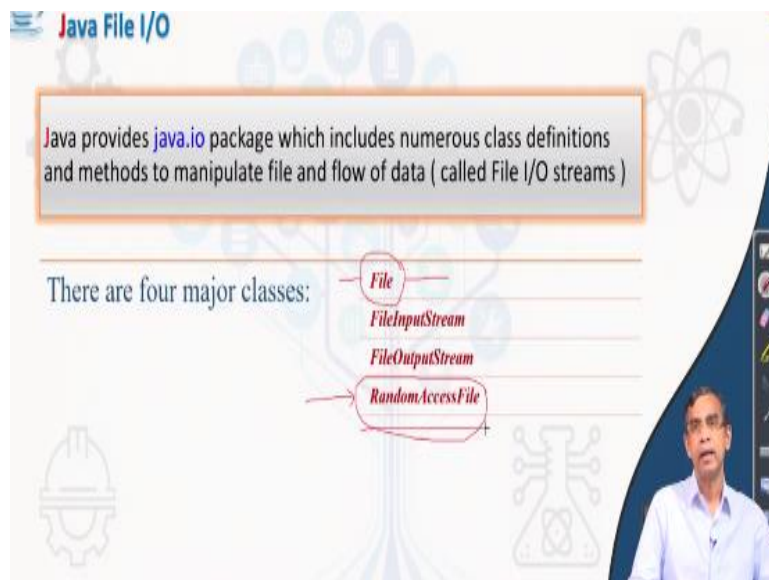
I can give a simple program for your practice so that you can learn many thing and then following the program that I will demonstrate, you can write your own program, but it require obviously graphical user interface namely say AWT or Swing. So, if you are not familiar to Swing then you can face a little bit problem otherwise it is fine. I assume that you are already familiar to JAVA Swing or JAVA Apex or JAVA AWT. So, this is basically our plan for today.

(Refer Slide Time: 04:57)



Let us proceed regarding discussions of the files in JAVA. So, file is basically an object, like other object in object oriented programming as you know everything is an object. So, file is an object and this file object is basically for permanent storing of data.

(Refer Slide Time: 05:17)



So, this file as I told you we are already familiar to handling files using as a byte stream like file input stream or file output stream that is for byte stream or also file reader or file writer as a character stream or whatever it is there, but in this lecture we will discuss one is called file another is called random access file. These are the two classes which are very important related to the file handling.

This file is basically sequential access file and this is the file is called the random access that means you can read from any position, write into any position that is why it is random access, but here file reading or writing can take place sequentially and another thing is that this particular type of file can allow you to work in one mode only. So, there are several mode read mode, write mode or append mode.

But random access file it can be opened both read or write mode together at one time, whereas this file can be opened at one mode only either read mode or write mode or it is append mode is basically one version of write mode. So, this is the idea about there. So, all those write these are basically JAVA supports available in the package JAVA dot IO and there are many classes already defined which we have already studied in the last discussions.

And today we will discuss about these files in details and then random access file will be covered in the other class. So, let us first discuss about the file structure actually what we can do using file concept it is there.

(Refer Slide Time: 07:12)

The slide is titled "Using class File" and has a sub-header "Opening a File object". It lists three ways to create a File object:

- There are three constructors
- Way 1:
 - File myFile;
 - myFile = new File(fileName); // Constructor 1
- Way 2:
 - File myFile;
 - myFile = new File(pathName, filename); // Constructor 2
- Way 3:
 - File myFile;
 - File myFile = new File(myDir, fileName); // Constructor 3

The slide also features a "WHAT WE DO?" graphic with a question mark and a small video inset of a man in a light blue shirt speaking.

So, here are basically file I told you is an object so if it is an object then how this file object can be created. A file object may be already available how we can make it active or instantiated. So, there are many constructor file there. Now, here I have given some syntax how you can do that. First of all, you have to create a file object and whenever object needs to be created you have to give a unique reference to this one. So, name of the file should be given here my file and naming of the file whenever you give the name of the file it should be compatible to your system.

Now many systems like windows and everything can allow you to create the file with blank space in between or underscore or some characters permissible characters, but not all. So, whatever the naming convention that basically allowed in your operating system you have to follow it here. So, basic convention that is basically followed is the same as how you declare an identifier. So, naming and identifier or variable is same as giving the name of a file as an object so that procedure we will follow it.

Now as the file is a class so you can create an object of this class by giving the name of the object here then this file can be created. So, a constructor is there, there are many constructor I have listed 3 constructors. This is the one constructors, this is the another simple constructors, this is another constructor and this is the third constructor. These constructor are for different purpose this constructor basically take the initial file name this is the input file name and then this file constructor will create a file object and this basically reference to the file object. So, this is the file object.

So here, you have to supply the name of the file and here this is another way. This is basically you can give the name of the file and you can direct that in which path or in which directory the file will be stored. So, you can specify the name of the path where the file will be stored. Now this actually this will create an instantiation that the file will be created in the same directory where your program is running so is a working directory, but here from the working directory to any other directory you can create a file.

And this is the third way where you can mention the directory name because in your working directory there may be sub directory or some other like. So, you can give the name of the directory and the file name. Now if the directory is not available at the moment, then it will create a directory under the working directory and this file name will be created there in this one.

So, this is basically the way the file object can be defined or object of the file can be created and once the file object is created then our rest of the things is that how you can store the data, how we can read the data from there either using file input stream or file output stream or file reader file generators those are the usual concept we are already familiar to can be followed here.

Now let us proceed further. So, basically what you can do is that first of all you have to create the file object and I have mentioned the 3 different way the file object can be created.

(Refer Slide Time: 10:41)

Using class File

Dealing with file names

- String getName()
- String getPath()
- String getAbsolutePath()
- String getParent()
- boolean renameTo(File newFilename)

WHAT WE DO?

+

Speaker

Once the file is created then we can have the different information relate to the file can be obtained. So, in one object is created so this object is basically give a connection to a file. File is having some name so if you want to know the name of the file so you can call the method `getName`, all these methods are defined in the file class. `getName`, `getPath` in which path the file is located.

`getAbsolutePath` path with respect to the with respect to the root or home and then `getParent` that means which is the parent directory of this path and then `rename` that means if you want to rename the file name then you can do also. So, these are the few already methods are supplied in the file class and you can use it for your managing files. So, these are the file related operation that you can do. Let us have some illustration so that you can understand about it.

(Refer Slide Time: 11:34)

Using class File

Testing a file

- boolean exists()
- boolean canWrite()
- boolean canRead()
- boolean isFile()
- boolean isDirectory()
- boolean isAbsolute()

WHAT WE DO?

Now there are again few more methods are there in the class file. So, this is a class file there are many methods are there whether file exists or not. You can create a file object, but file may not exist that is what you can write that means whether file is write mode or read mode or read permission or write permission is available or not so that you can test. So, it is basically to check whether read permission is there.

Whether it is a file or it is something else because something which is not as a stored as a file it is it is some some sort of what is called the internal or hidden things so you can check it and then directory whether it is a directory or not is absolute path or not. So, these are the few methods related with the file management that you can consider while you are working with file.

(Refer Slide Time: 12:30)

Using class File

Getting file information

- long lastModified()
- long length()
- boolean delete()

Directory utilities

- boolean mkdir(File newDir)
- boolean mkdirs(File newDir)
- String [] list()

WHAT WE DO?

Speaker: [Video feed of a man in a light blue shirt]

Now so there are many other information also you can have it. So, when you create a file the data of creation of the file, if you modify the file, date of modification of the file, length of the file also can be available. If you want to delete a file also you can have the method for that and if you want to make a directory, if you want to make a directories list of directories, if you want to make all the list of files under a current directory those are the iterative program which is defined in the class file.

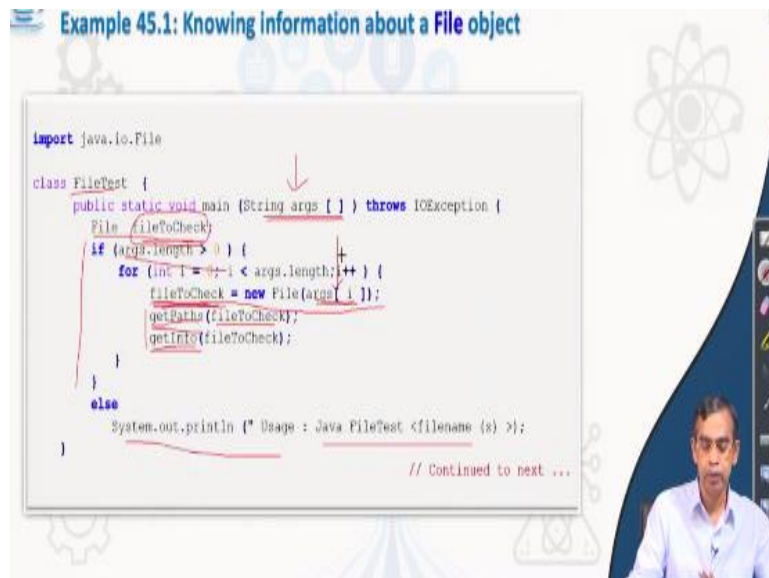
That means many more things related with the file management also can be obtained from the class file and that is a great advantageous for the program because from the program point of view we can create something from the program itself instead of doing this things manually rather. So, automatically many more things can be done related to the file handling actually. So, these are the file handling utilities that JAVA supports to the programmer.

(Refer Slide Time: 13:42)



Now let us have certain programming understanding because we have discussed many utilities that is there so better it can be understood if we follow certain program.

(Refer Slide Time: 13:56)



So, the first illustration it is basically explain you to know the different information about the file. So, this is very simple program you can follow. Let us look at this program this program is basically related to test a file. Test means the name of the file, path of the file, size of the file like. So, here is the program actually. So, this is basically the file objects to be created so this is basically reference to a file objects.

So, whenever file that can be referred to this file to check. So, this is the name of the object. Now here, so file to check you see this object can be connected to a file. Now here, args I so basically this is a common line argument you can test one or more file. So accordingly, while you run this program you should pass the name of the file or files 2 files, 3 files, 1 file if you do not pass no file it will not do anything so because it is there.

So, common line argument should include the name of the file. So, whatever the name of the file you pass through the common line here it basically here from there then argument so it basically create a connection from your file. This is the name of the file that you have passed as a common line right and it basically connect to this. So, now it basically is the file object. So, this file object connect to the name of the file that you have a passed as a first argument, second argument and so on so on it basically run the loop.

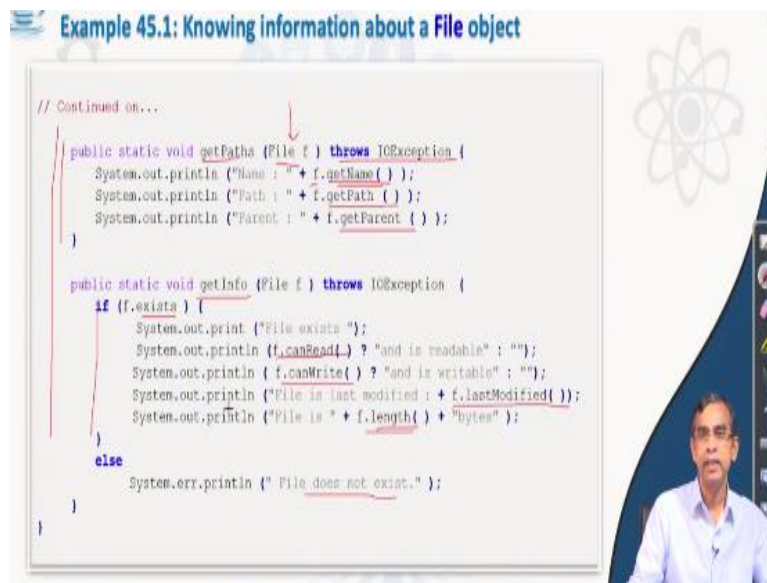
So, letter is the first name is abc dot text so it basically file check is basically refer to abc dot text means this is the file object. Then you pass this file object as an argument to this method which method is basically used at different method we will discuss this method right now as a continuation of this program. So, what this program we will do is that it basically get file and get information regarding the file that you are that you are interested.

Now if args dot length is not is equals to 0 or like then it basically error that you should give the name of the file to run this problem. So, this is basically first part of the program. Now here you see how we can define these two method getPaths and getInfo regarding the status of the file actually. So, in the next part as a continuation of this program we define this two methods get path and get info.

(Refer Slide Time: 16:41)

Example 45.1: Knowing information about a File object

```
// Continued on...  
  
public static void getPath (File f) throws IOException {  
    System.out.println ("Name : " + f.getName ( ) );  
    System.out.println ("Path : " + f.getPath ( ) );  
    System.out.println ("Parent : " + f.getParent ( ) );  
}  
  
public static void getInfo (File f) throws IOException {  
    if (f.exists) {  
        System.out.print ("File exists ");  
        System.out.println (f.canRead() ? "and is readable" : "");  
        System.out.println (f.canWrite() ? "and is writable" : "");  
        System.out.println ("File is last modified : " + f.lastModified ( ));  
        System.out.println ("File is " + f.length ( ) + "bytes" );  
    }  
    else  
        System.err.println (" File does not exist." );  
}
```



So here is a definition the first is a `getPath` method it takes as a file object that you pass it and it should throw IO exception in case some error occurs either file is damaged, file is not available, file is not readable or these kind of things are there. Then you see the `getName` method we have already mentioned that for this file object the `getName` that means it will return what is the name of the file.

`getPath` it will basically in which directory, in which location the file is available `getParent` it basically say that what is the directory that file is available. So, this basically regarding the `getPath`. Similarly, `getInfo` regarding the file many information can be available. So, this is basically if `f` dot exists that means if file exists, then only it will do otherwise it say that file does not exist error.

Now, so file exists then `f` dot can read whether this file is readable then it will basically print is a Boolean whether it is write mode that means you can write on the file or not it can give it and on when is the last date this file is modified it will print the date and what is the size of the file in bytes length. So, these are the different information that you can obtain for a file which you passed as an argument and then you can read it.

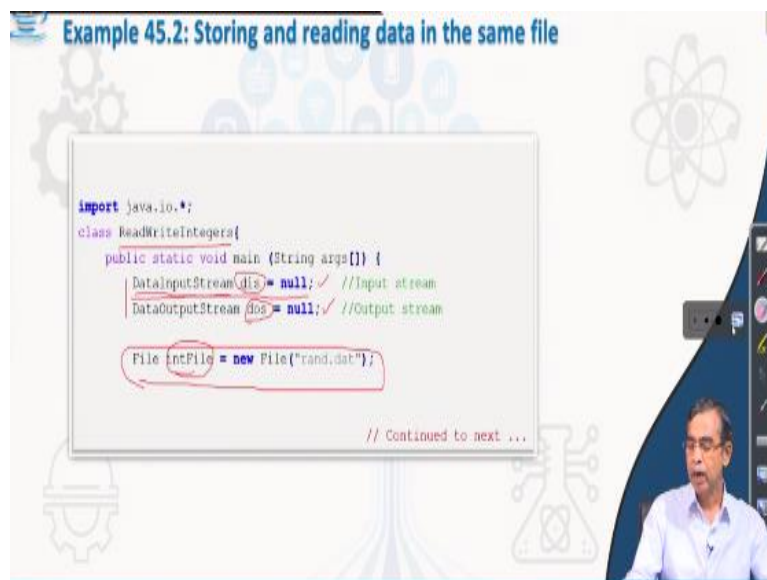
So, you can run this program with many test cases so that you can check that your program is working and you can obtain the different information of the path. So, this is a good program to understand about learning about the file handling utilities in little bit.

(Refer Slide Time: 18:21)



So, there are many more those methods are there you can use it, but this is a representative examples so that you can follow and you can use it. Now so this is basically regarding file handling utility that we have learned now let us see how we can store data into a file. So, storing data into a file we have already familiar to this kind of activities while we are discussing about stream IO byte stream or IO character stream. There also we use that how the data can be stored either reading from the keyboard or copying from one file whatever it is there. So, we will do the same thing again here, but in a different manner of course so that we can have few essence about it.

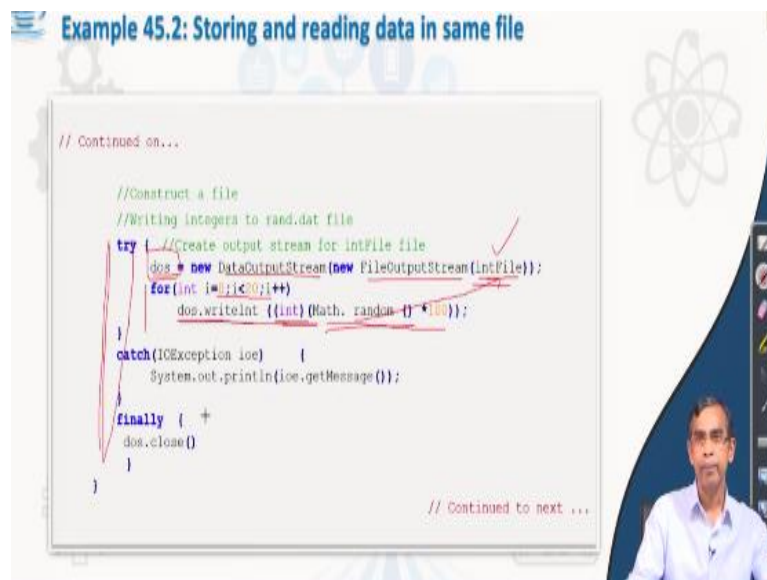
(Refer Slide Time: 19:04)



So here read, write integer so this program basically we want to exercise reading and writing into files. Some integers or some tabule you can write into the file. Now for this purpose we can create any object file input stream or data input stream or file buffer or file depending on whether you want to read as a byte mode or character mode this one. Here in this example we want to read we want to read a file for in the data input stream that means read the file as a byte form so, for which we create the input stream object and then output stream basically to write it.

So, two stream objects are created and so initially they are null because they are not connected to any input source file. Now here we just create a file object this is the name of the file. In file now this file this is basically our file object now we can create an input stream connected to this file in the next part of the program we will give you how this file can be connected to a impaired stream so, you can read the file from there as a byte or it can be read from a primitive data also. Data input stream is for the primitive data reading actually as a primitive data.

(Refer Slide Time: 20:31)



```
// Continued on...

//Construct a file
//Writing integers to rand.dat file
try { //create output stream for intFile file
    dos = new DataOutputStream(new FileOutputStream(intFile));
    for(int i=1;i<=10;i++)
        dos.writeInt((int)(Math.random()*10));
}
catch(IOException ioe) {
    System.out.println(ioe.getMessage());
}
finally { +
    dos.close()
}

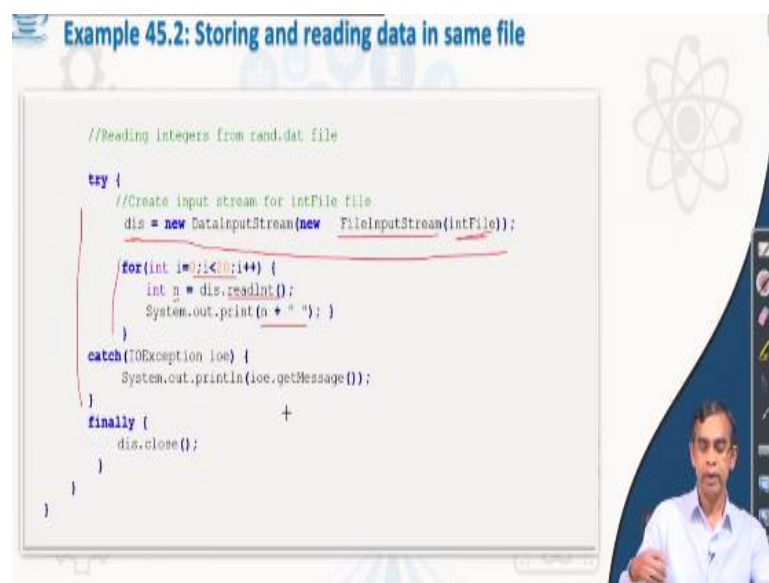
// Continued to next ...
```

So, the next part of the program if you follow it basically tell you how it can be done. So now, you see so the file or the object is there now data output stream we create another object it is called the data output stream that means it will write some data into it. So, data output stream and file output stream int file. So, this basically integer file from which we want to read some data. So, we want to write some data into this file first and then this file we will use to read data from there.

Now how the different data can be stored there? Now here we have used one random number generator so it is a random number. So, 20 random number will be generated using this random number generator and this number will be converted as an integer and then writing method we can call for this data output stream so that this integer we generated we will store into this file int file which we have already have this one int file object is created. So, this program this part of the program basically create a file and this file contain some random number 20 random numbers.

In the next part of the program let us see how we can read this file using our data input stream object. So, now data input stream can be connected to this int file.

(Refer Slide Time: 22:03)



```
//Reading integers from rand.dat file

try {
    //Create input stream for intFile file
    dis = new DataInputStream(new FileInputStream(intFile));

    for(int i=0;i<20;i++) {
        int n = dis.readInt();
        System.out.print(n + " ");
    }
} catch(IOException ioe) {
    System.out.println(ioe.getMessage());
} finally {
    dis.close();
}
```

So, this is the next part of the program where you can see how the file that have been created recently can be read actually. So here, these are data input stream object we have already declared there. So, we can create a connection this connection is basically you see Int file and this is as an input stream because we want to read the data from this file and data input stream basically it will read the data in a primitive fashion although it is in byte mode.

So, byte stream is there but return will be in a primitive mode. Now here again the loop because it contain 20 data we want to read all 20 data. So, read int all date is integer form so read int n and then print the data. So, whatever the unknown number it was generated in the earlier file store in a file and in here we are reading the file. So, this basically gives a reading and writing file using data input stream it is clear.

Now I can repeat the same program and I have not given I just give as an exercise to you so that you repeat the same reading writing, but not using byte stream like data input stream or data output stream rather you can repeat using file reader and file writer that basically we will first you call the file writer to create the data with integer random 20 random numbers and then use the file reader to read the data and print it in the console.

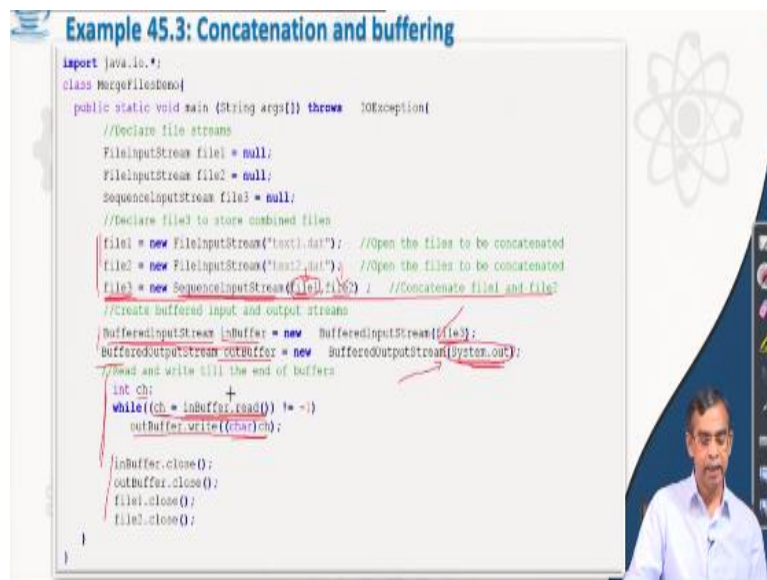
So, that program you can practice of your own just left as an exercise. So, we have learn about how the data can be stored and stored data can be read this is an example that you can follow it.

(Refer Slide Time: 23:49)



Now we will discuss about how two files can be merged together. Merging means content of the two files can be copied into a third file like. So, so this is there is unique one input stream is available and this is called the sequence input stream class. The sequence input class we have not discussed in details, but now we get an opportunity to cover it this is basically useful for merging two files or it is called the concatenating, concatenating two files.

(Refer Slide Time: 24:26)



```
Example 45.3: Concatenation and buffering

import java.io.*;

class MergeFilesDemo {
    public static void main (String args[]) throws IOException {
        //Declare file streams
        FileInputStream file1 = null;
        FileInputStream file2 = null;
        SequenceInputStream file3 = null;

        //Declare file3 to store combined files
        file1 = new FileInputStream("text1.dat"); //Open the files to be concatenated
        file2 = new FileInputStream("text2.dat"); //Open the files to be concatenated
        file3 = new SequenceInputStream(file1, file2); //Concatenate file1 and file2

        //create buffered input and output streams
        BufferedInputStream inBuffer = new BufferedInputStream(file1);
        BufferedOutputStream outBuffer = new BufferedOutputStream(System.out);

        //read and write till the end of buffers
        int ch;
        while((ch = inBuffer.read()) != -1)
            outBuffer.write((""+ch));

        inBuffer.close();
        outBuffer.close();
        file1.close();
        file2.close();
    }
}
```

It can be used 2 files, 3 files or sequence of files also it can be there. Now we will consider this example how the two files can be merged together. Let us have the illustration this is the program. This program is basically merged file demo and here we create the two objects the file 1 and file 2. The two objects are of file input stream they are basically for reading. So, two files are for reading initially they are not connected to any files so we made the null. Then we create another file 3, but this file 3 as an object call sequence input stream.

This sequence input stream I was just referring here this basically a class which basically takes an input of two files. It will read one file and then copy into third file once if the first reading is over then it will open the next file and copy this one. So, automatically it will do it. Now let us come to this file 1 we just connect this file 1 to this text 1 data. So, this file 1 is basically is a input stream collection from your program two input stream. Here the target file is basically a source file is text1 dot data. So, file 2 this is another source text 2 data. Assume that these two files are already been created and available under your working directory.

So, we just open the two files file 1 and file 2 for the merging purpose, so two files are here file 1 and file 2 that needs to be merged together. Now you note this statement this is very important statement here this basically call this sequence input stream this class object we created. Name of the object that will be created is file 3, but it will return the file 3 taking file 1 and file 2 are the two input.

So here, two inputs are given to this constructor actually this constructor will create the object is a third file concatenating the content of file 1 and file 2. So, it is called the sequence input

stream because first it will take file 1 content so it basically byte by byte copy, copy into file 3. Once file 1 is copied then it will open the file 2 byte by byte it copied into the end of the file 3 that is there. So, file 3 will basically the concatenation of two file, file 1 and file 2. Now so this basically give the concatenation.

Now here you can see how we can write open this file to read this content file 3 we use the buffer input stream as an in buffer. You can read some file reader also no problem that can be done also and here out buffer we want to create the output for purpose. So here, basically file 3 as an input we read this file 3 and we write this file 3 as an output that means it is a console. This is basically standard output system dot out just opposite to system dot in. System dot in is a standard input called output and system out out is basically the standard output console or monitor. So, we create the two stream one is input stream from which the data we will read.

And another stream called output stream to which the data will be sent. Now here basically let us see how we can read the data from input file and then print the data on to the console. So, here we read a character so this is basically the character reading. So, it read a character is basically this is buffer input stream. So, we read as a byte so this basically read byte and this basically returned as a byte.

Byte can be store as an integer so it is an integer and then we just write this byte converting into the character and display on the output buffer. So, this is basically system dot out. So basically this statement is basically continue reading byte and byte and display on the monitor. So, this way the file 3 which merging of the two file that you can create giving some initial text in it or data in it and then you can write it and you can set it.

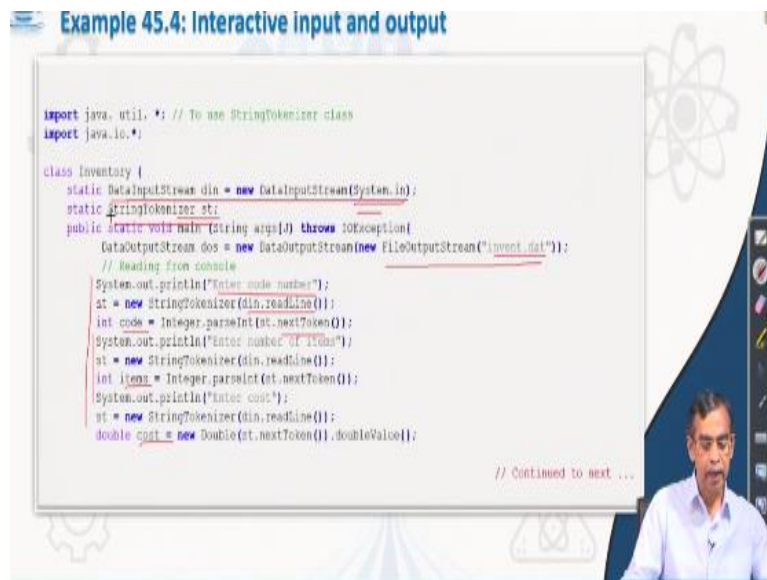
And finally you can close the buffer both file 1, file 2, file 3 so many objects have been opened related to the file and those thing should be copied here. So, this program explain how the data or how concatenation and buffering both can be done together using a simple example. So, these are the program that we have discussed about.

(Refer Slide Time: 29:21)



And then we will quickly come to the interactive input output. The idea is basically you read something from your side and put into the output you can understand what is basically the program to be done.

(Refer Slide Time: 29:35)



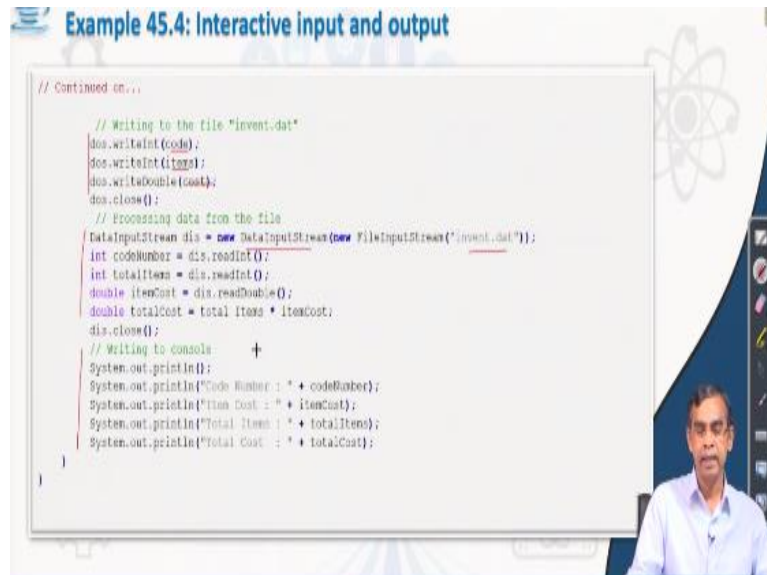
I have just given an idea about how this program can be done for you. So, you program you can check it so this program is basically interactive input output is basically read at the token. Token means symbol one by one and then it basically print on the screen. So, here data input stream the object is created from the standard input that mean we want to read something from the keyboard and we use the stream tokenizer basically read as a token.

Here, the string token by default is a blank space. So, you type something and then space it can send a token then next again type something again space again another token and it will continue until you end the program by control z as a ending. So, this is the string tokenizer that you can consider for this purpose. Now here we create so this is the data input string to read something from the keyboard.

And here whatever we read we want to store into a output is called invent data that we just want to read. Now here we are giving some prompt and reading that token and then we convert it into the data and store into the file here. So, enter code number like read line converting the string tokenizer, string and then next token read the next token and then convert.

And this way you read few data from the keyboard and then store the data like code, items, cost these are different value that we read from the keyboard and is store as a temporary variables and finally we are to redirect to them to an output file the data output stream that we have invent data actually we have created.

(Refer Slide Time: 31:32)



```
// Continued on...

// Writing to the file "invent.dat"
dos.writeInt(code);
dos.writeInt(items);
dos.writeDouble(cost);
dos.close();

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

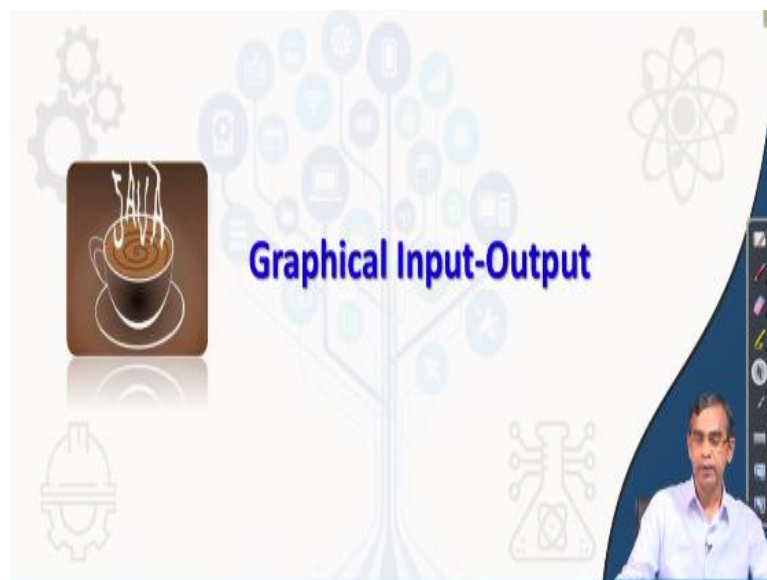
// Writing to console
System.out.println();
System.out.println("Code Number : " + codeNumber);
System.out.println("Item Cost : " + itemCost);
System.out.println("Total Items : " + totalItems);
System.out.println("Total Cost : " + totalCost);
}
```

So, here the next part of the program which basically save the data. So, here you can see the data that we have read code, items and cost and everything we can write into the output file and then the same output file can be again opened and then we read the content and here basically reading. This is again another input string we have created, reading the data and we can read the data.

Now this program is very useful whenever we want to write some objects into the file. Now writing objects and reading that is again another concept is called the serializable while we will discuss the concept of random access file in our next lecture while we cover it. Anyway so this program also can be used to I mean write the object actually because object has some fields those fields are is a primitive data type like this one.

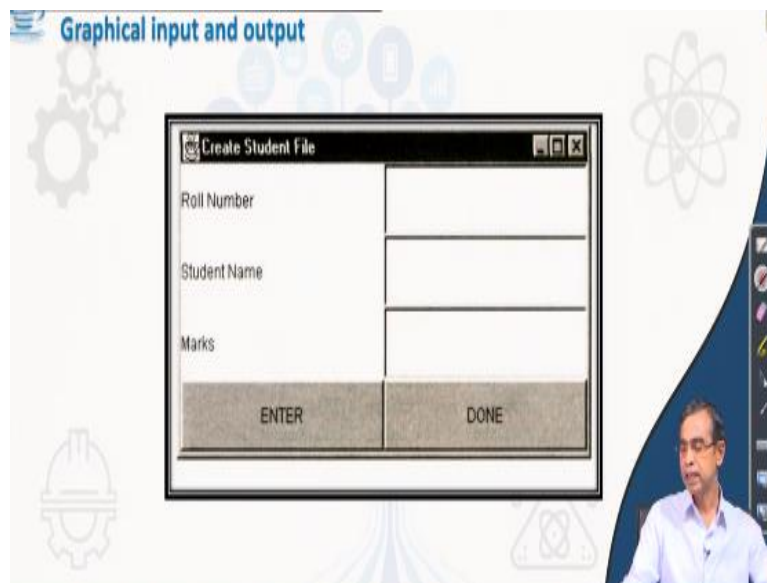
So, we can traverse each objects and object wise, object wise you can store into data and reading also in the same fashion the way we write it and so this is the one naive approach of course. So, reading field by field for an object and then again getting back field by field reading from the input and then reconstructing the object then are also possible and directly as an whole object also you can store the data and whole object also we can read that is another topics to be discussed. So, this is example and is a interactive input output.

(Refer Slide Time: 33:08)



Let us have some other example about another application rather we can say is a graphical input output. This program if you are not familiar to the Swing and others you will find it bit difficult. So, I will not think about that sense, but assuming that you have little bit knowledge about graphical user interface.

(Refer Slide Time: 33:26)



It basically first create an interface. This interface will look like this so this interface has the few text field it is there so here user will type as a roll number, then student number and then marks and then enter if you just click the button then data will be saved to a file and then next entry can be allowed and then this way data can be read from this interactive it is basically GUI and then data will be pushed to the output.

And then whenever your work is done then you can click the done button and then that program will end. So finally, data whatever you input it will be stored in the file then you can open this file, read this file and whatever it is there. Using file utility many things you can do it. So, this program is basically is a simple way I have written for you.

(Refer Slide Time: 34:27)

```
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
    }
}
```

// Continued to next ...

And I just explain this program because it is okay I explained the program rather we can say. First of all we have to create the user interface using text field, button, label and then we create the data output stream because you want to write the data and here the label these are the 3 different level, their names, roll number and then marks obtained by the student and those data will be created in a file called student file. So, student file we have to create it and then this is basically display the graphical user interface with some keyboard even handling mechanism in it.

(Refer Slide Time: 35:06)

```
// Continued on...

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

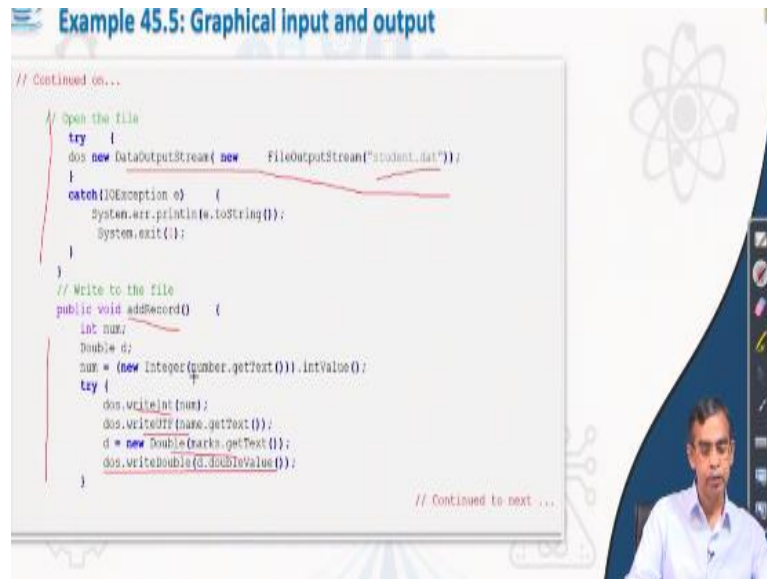
// Add the components to the frame
add(numLabel);
add(number);
add(nameLabel);
add(name);
add(markLabel);
add(marks);
add(enter);
add(done);

// Show the frame
show();
```

// Continued to next ...

So, this is the continuation of the program is basically how you can create the interface it is shown the interface, there are different buttons and label, text field those are created it needs to be entered into this one and when this interface is created we have to print the interface. So, the interface can be print.

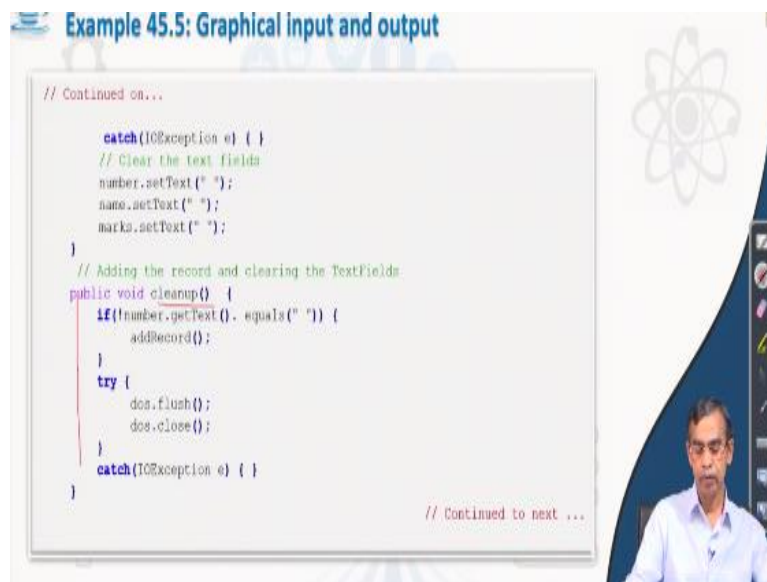
(Refer Slide Time: 35:26)



```
// Continued on...  
  
// open the file  
try {  
    dos = new DataOutputStream(new FileOutputStream("student.dat"));  
} catch(IOException e) {  
    System.err.println(e.toString());  
    System.exit(-1);  
}  
  
// Write to the file  
public void addRecord() {  
    int num;  
    Double d;  
    num = (new Integer(number.getText())).intValue();  
    try {  
        dos.writeInt(num);  
        dos.writeUTF(name.getText());  
        d = new Double(marks.getText());  
        dos.writeDouble(d.doubleValue());  
    }  
}  
  
// Continued to next ...
```

In the next code you can see how the interface can be print by running the print method here and then this basically create the object data output string where the file will be created. Name of the file is student dot dat and then here write to the file add record this basically using the right int is a student name as a string the marks just write and then these are the value that you have read from the interface will be here is the read from the interface will be stored into this output data file. So, this basically work for you as a simple graphical interface.

(Refer Slide Time: 36:04)



```
// Continued on...

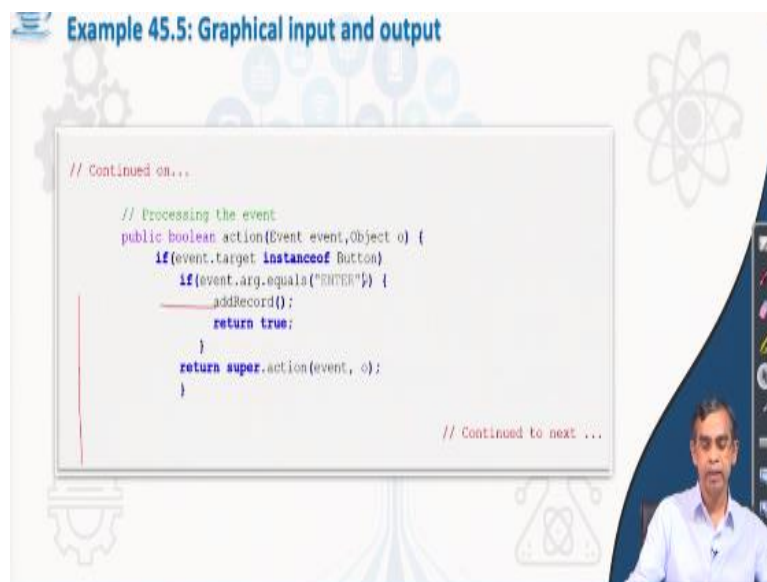
    catch(IOException e) { }
    // Clear the text fields
    number.setText(" ");
    name.setText(" ");
    marks.setText(" ");
}

// Adding the record and clearing the TextFields
public void cleanup() {
    if(!number.getText().equals(" ")) {
        addRecord();
    }
    try {
        dos.flush();
        dos.close();
    }
    catch(IOException e) { }
}

// Continued to next ...
```

And here is basically the program that you can write and once you complete one writing so clean up is there because if you done enter so it basically fresh the interface so cleanup. So, the next record will be allowed to enter it and this program you can practice, you can run in your own machine and you can check that it is working.

(Refer Slide Time: 36:28)



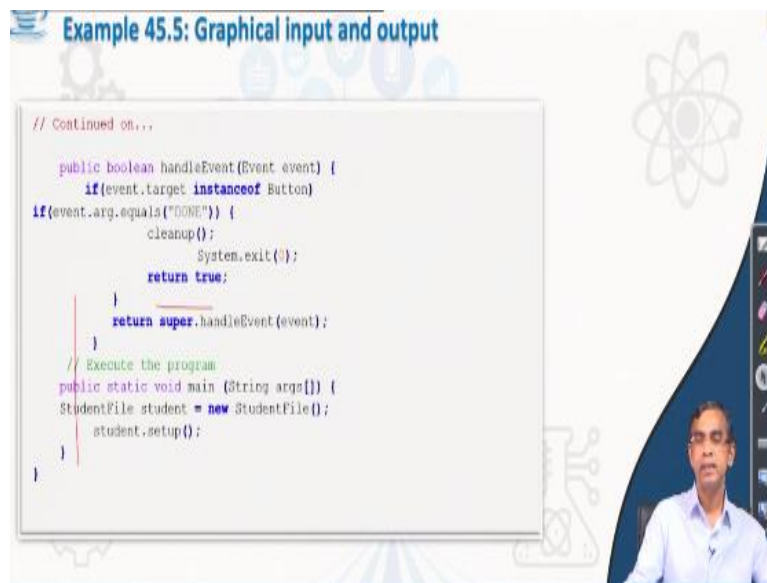
```
// Continued on...

// 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);
}

// Continued to next ...
```

This is keyboard handling, event handling it is used because we have to enter the document and then either click the button. So, there are two events to be care here there are button events are there enter and then done two things that we have to use it.

(Refer Slide Time: 36:43)



The slide displays the following Java code:

```
// Continued on...  
  
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();  
}  
}
```

So, there are done and enter the two buttons are to be clicked once it is there then it will call some events to be takes place and this is the main method where it basically call this file create student file and then student dot setup that method we have already define it earlier. So, this program you can run it is very interesting program actually is a very simple, but you can run it and you can check that this is working and the student dot data file will be created. This file will store the data which basically is here. So, it is a data entering activities that we have studied about it.

(Refer Slide Time: 37:14)



The slide lists the following references:

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

Now we have discussed about file IO, file IO is exhaustive content actually many more information that you can obtain for which I have given the link for you. So, as a supplementary material I should suggest you to go through the link and study in details. So, these are the topics that I want to convey it for today. Thank you, thank you very much.