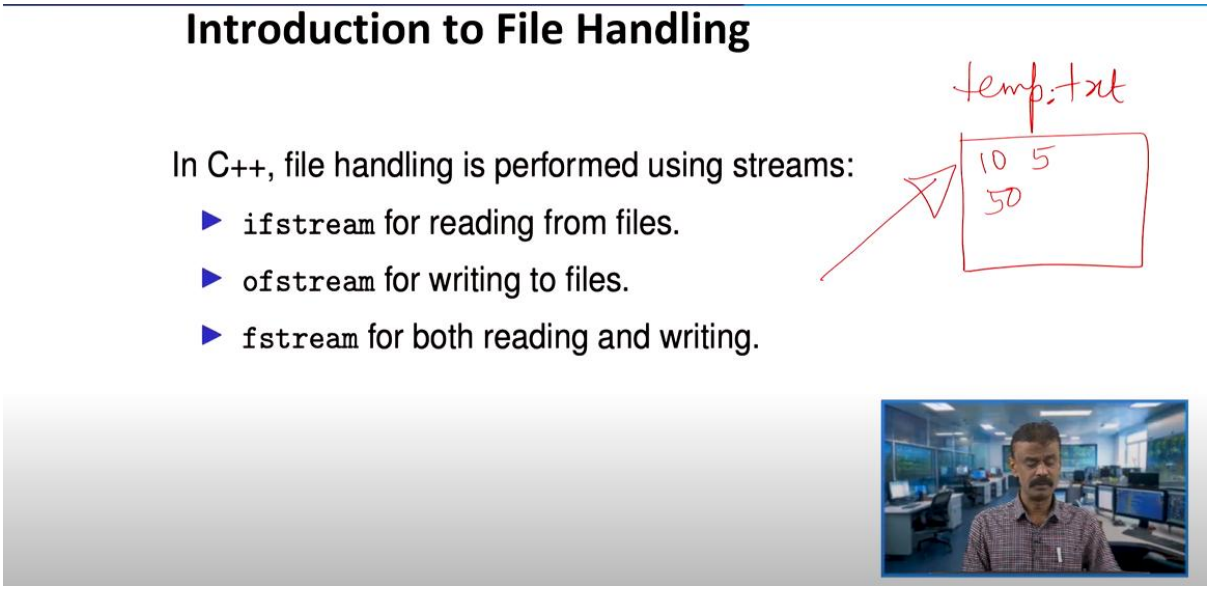
FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture31
Lecture 31: Basics of File Handling



So, welcome to the new chapter on file handling. So, this will be your lecture number 31. So, suppose right, what we have done so far is you are giving the input, right, and then you are running the code, and then you are getting the output. So, what will happen if I give the input using a file, right? So, that is called file handling in C++ and Java.

So, we are going to see it in both object-oriented codes, right. So, first, we will talk about C++. So, when We are dealing with this problem. Right.

File handling. So, you have to include. Right. So, we have input file stream. ifstream.

Like iostream, we have studied. Right. Input-output stream. So, input file stream. Right.

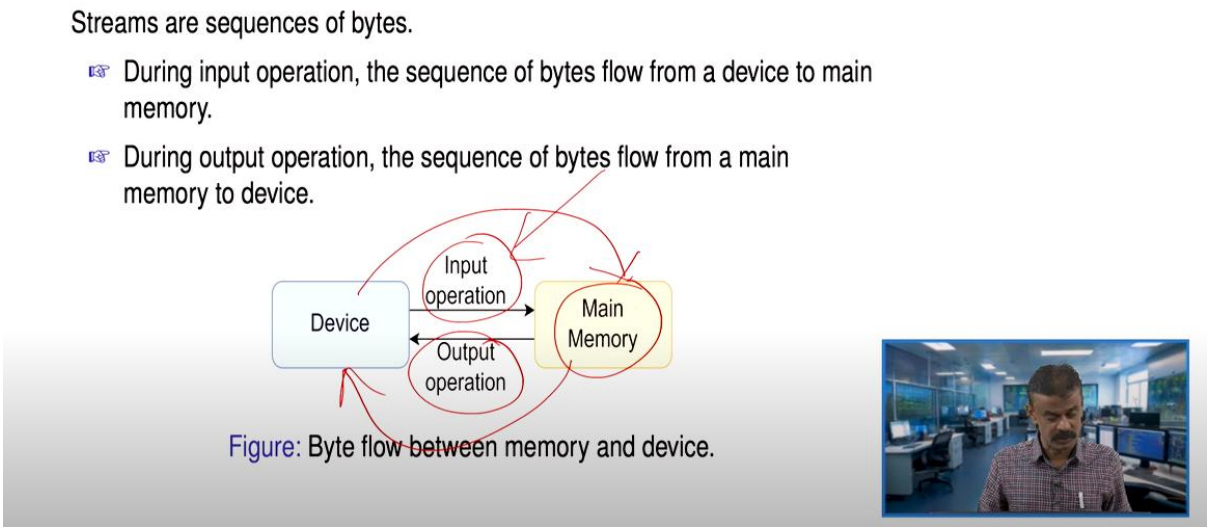
So, this input file stream is for reading the file. Right. So, reading from the file. And ofstream. Right, output file stream for writing to the file.

Suppose, let us assume, right, we are taking two values, right. So, I will just use the pen. So, let us say a and b. So, in a file, assume that I have a temp.txt, right. Assume that I have temp.txt. So, now, I am giving, let us say, two values.

In the temp.txt, I have, let us say, 10 and 5, right. So, 10 and 5. So now, I will be writing the program to multiply two numbers, right? 10 into 5. So let us say I have to get 15.

So now, you are sending it as a file, right? The input stream, right? That means the input file stream. So now, we have to read these two values, 10 and 5, from the file, right? And then, 10 into 5, we know it is 50, right?

So, in the same file or in a different file, I am writing the output, right? So that can be done. So, the output file stream, for example, I am using the same file. So, 10 into 5, I have to get 50. So, for example, I am getting 50, right?



So this is for the output file stream. And fstream, so that will be the file stream, right? So which is for both reading and writing, right? So this is what... We are going to handle.

So that means we are going to give the input as a file. So this will be useful. For example, assume that this is a world. I mean, if you are using some machine learning problem, you may get huge data. So these data, if I want to handle, there are several other languages that I mean you can read the file, but if you know, let us say in basic C++ and Java, you can do it with other languages also.

So that means I will be reading the file, and from the file, whatever the input is, let us say it has several lines. So every line we can read, and then you are writing a program. So what exactly are you going to do? So initially, what we can do Since all

are new to this file handling, we will see how we are going to write and read or read and write.

So if I take ifstream, that is for reading from the file. And if I take ofstream, that is for writing to files, and only fstream for both reading and writing. It is like cache include. You have done IO stream. So in this case, we are going to do ifstream, ofstream, and fstream.

So now, consider. So what do you mean by stream? I talked about streams in the previous slide. So streams are nothing but sequences of bytes. So during the input operation, the sequence of bytes flows from a device to the main memory.

So when you have the input operation, that means I am reading the 10 and 5. Assume that you are going to read 10 and 5, right? So that means the sequence of bytes. So we know that these 10 and 5, or any text, will be stored in some memory location, right?

Even the complete file, right? So when I am reading line by line, the sequence of bytes flows from a device to the main memory. So that is the case of an input operation, correct? So then you may ask the question, what about the output operation? So in the output operation, assume that the main memory

We have taken a simple example. 10 into 5. So when you are multiplying 10 into 5, I have to get 50. So that means,

In the case of an output operation, The sequence of bytes flows from The main memory to the device. So this is exactly what is happening. Correct.

So the streams are nothing but. The sequences of bytes. So this diagram is nothing but. The byte flow between memory and device. So during the input operation.

The sequence of bytes flows from the device to main memory, and vice versa happens for the case of the output operation. So let us consider the first program using a file. So hash include fstream, that means both input and output, right? So that is what both reading and writing are. So that is what we said, right?

Example: Writing to a file in C++

```
1 #include <fstream>
2 int main() {
3     std::ofstream myfile("example.txt");
4     if (myfile.is_open()) {
5         myfile << "Writing!\n";
6         myfile.close();
7     }
8     return 0;
9 }
```

Both reading and writing, we use fstream. So you are including fstream. So int main. So I have myfile, which is the object, right, which is the object under the inbuilt output file stream. So that is a class.

Output file stream is a class, my file, and the constructor you are passing in example.txt. That means it is handling the example.txt file. I think we have taken the same example, right, temp.txt. I said here temp.txt. So, here example.txt.

If my file, right, the myfile object is invoking the inbuilt function is_open, right, is_underscore open. So, which is available in the fstream or ofstream, all right. So, because you have included fstream. Yes, it is there, both for reading and writing. Right, that is an inbuilt function.

So, if you are able to open the file, right, if you are able to open the file, so that my file, right, this is like a cout statement, all right. So, my file, so that means you have the object, all right, so which is writing, right, so which is writing. So, finally, assume that I have written to a file, all right, the title if you look, So, this is a C++ program for writing to a file, right. So, myfile, if myfile dot is open, if you are able to access the file, example dot text.

So, myfile, you are putting the writing and then slash new line. That is a usual one, like a cout statement. So, myfile, the object, which is writing. Once it is over, assume that you have written something, you have to close the file. So myfile will invoke the close function and then return 0.

So this is what is explained over here. So in this program, we have the combination of, so I have done, you have fstream, hash include fstream, and then we are using

the class ofstream. Right. We are using the class ofstream, and myfile is nothing but the object. Right.

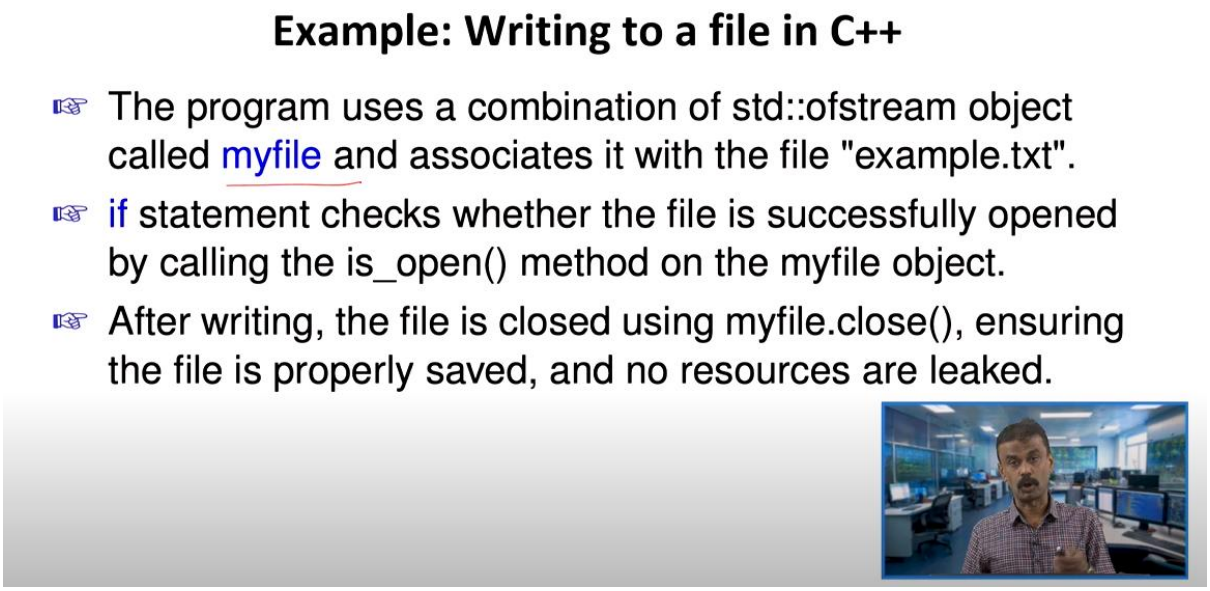
So myfile is an object, and this is associated with the file example.txt. Right. So this is being associated with the file example.txt. Right. And then what is happening?

The second case is the if statement. Right. myfile is invoking the is_open method. Right. Or the is_open function, a member function.

It is an inbuilt function. So, is it possible to open the file successfully? So, if it is so, if the example.txt file can be opened successfully, then we have, if it is true, then myfile is writing. So, myfile will write, and then you have to close the file. So, that means myfile is invoking the close

member function, which is ensuring that the file is properly saved, right. So when you use this close function, right. So this ensures the file is properly saved and no resources are leaked from the file, right. So this is how we are going to write to a file in C++. Maybe we will take an example or we will run the code.

So, let us consider this program, and also we can see example_cpp.txt. Now, we will open the example_cpp. So, when you are running the code, here you go example_cpp.txt. So, there you are getting the writing as the output. So, here you are getting



Then, when I run the code again, it will overwrite. Right. Writing will also come, and then 'Welcome to file handling' will also come. In fact, you have seen the output. So, this is the basic program.

Right. So, from here, you can develop. For example, you are developing a logic for adding two numbers or multiplying two numbers. Right. So, you can take the input from the file.

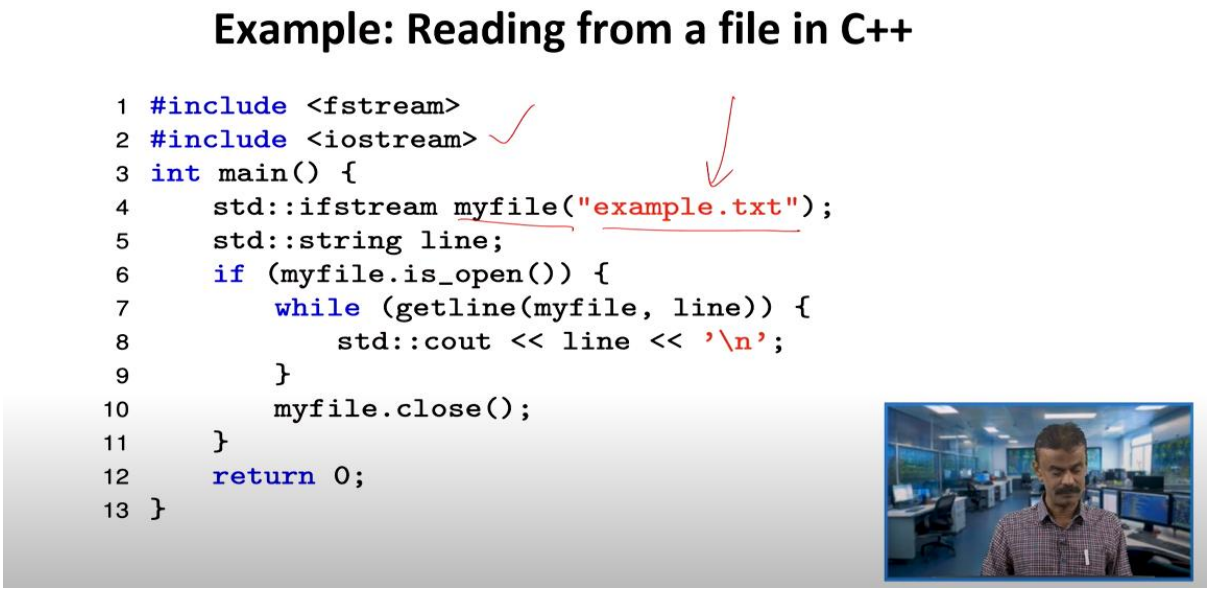
So it has to read. Maybe we will see some more examples. Right. So then you will understand. Okay.

So now we will go back to the theory. And again we will come back to C++ as well as Java. I hope you understand the basic file handling program in C++. Right. So we have done this.

So now another interesting example: reading from a file in C++. Right. So reading from a file in C++. We include iostream. Right.

Input output stream. This is a usual one you have studied. Right. So only fstream. Right.

So now we have myfile under input file stream. The previous program we had seen output file stream. Myfile is output file stream under the class output file stream. And here, if you look in the case of reading, we have ifstream, which is a class, and the object is myfile. Again, we will have the same example.txt, right? We have the same example.txt. So now we have One line variable, right? Whose data type is string, right? Whose data type is string? Again, similar, if myfile is open, so if I could open this example.txt, right While getline myfile, comma, line, right? So that means myfile is an object, correct?



writing as an output. Once again, we can see the code. Alright, so here you go the same. Right, my file is an object, ofstream is a class. So here you have right, my file will access example_cpp.txt. Right, initially it will be empty, and if my file is invoking is_open, if it is able to open it, then what we are doing, my file so this is under fstream is going to do the writing. Maybe we will include one more right after line number 5, my file, right. I am writing that next line. I will put control C right next line, my file, welcome to file handling, something like that. We will write a new line anyway. I put my file, right, my file output operation.

Welcome to, welcome to file handling, right. You put the new line, \n, right, \n, semicolon. So, let us see. So, example.cpp.txt is there. Maybe initially, I think you have to do empty, right.

Can you open this example? Yes. So, it will override. So, no need to worry, right? So, let us run the code again.

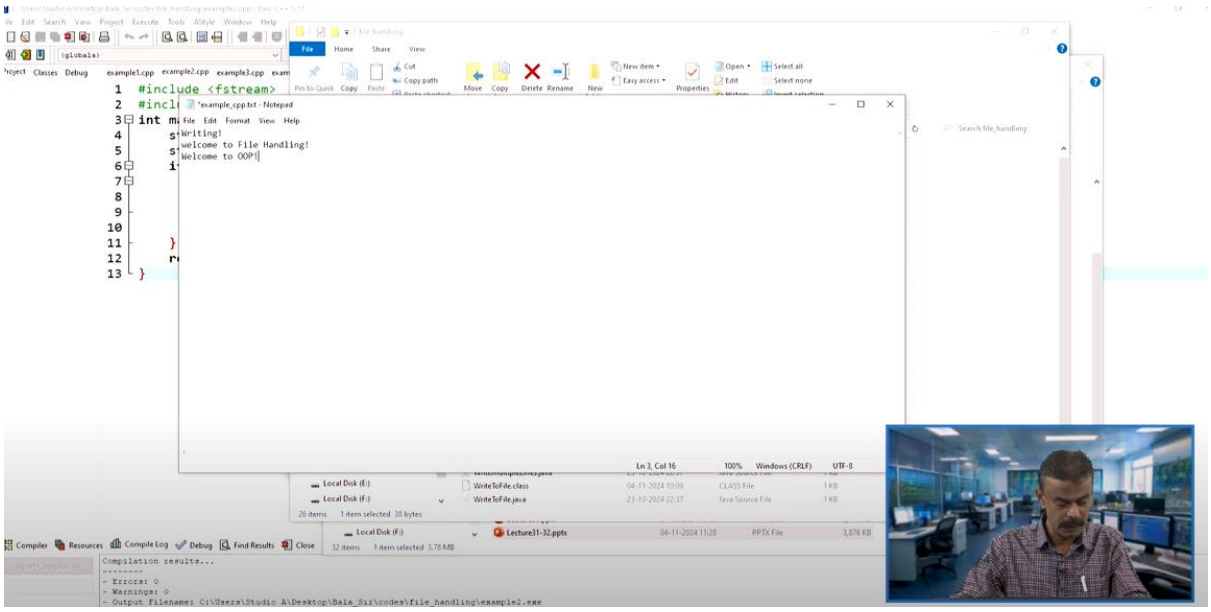
Let us see what happens. Compile and run. We will see it executing successfully. And if you look at the file, yes, you can see writing because I put a new line. It is printing 'Welcome to file handling' again.

So, this is how you can do it. It is like printing 'Hello, World', right? So, I have not given any input, right? It is reading the file. Example_cpp.txt.

Even we have handled the second case. The second case, the file contains only writing, right. Initially empty. And in the second case, it has only writing if you recall. Right.

And line is a string. Assume that I am reading the first line: Welcome to IIT Roorkee, correct? So what it is doing is cout, so whatever You have every line, right? Assume that I have line 1, line 2, line 3, line 4, line 5. So, it is printing those lines, right? It is printing those lines.

Cout line. So, it will print. In the console, we are going to see the output. And then, as usual, you have to invoke the close function on myfile, all right? So, then you have to return 0.



I hope you understand. So, myfile is an object. It is under the class input file stream, ifstream. Alright, and then it is trying to open example.txt. If it is successful, while getline(myfile, line) is an inbuilt function. So that means in the myfile object, and line, as you can see, is line number 5. It is the variable under the string data type, right?

Alright, so that means line by line it is reading, and then it is printing in the console. Every line is being printed in the console. So now, once it is over, if you are at the end of the line, correct? So you have done everything. That means when it becomes false, it will come out, and then myfile will close it, right? So myfile will invoke the close function. That is the meaning, right? And then return 0. So let us run this code. This is an interesting code. How you are reading, okay.

So, we will go to the program. So, example2. So, we will see what example.cpp contains. First, we have to see what example2.cpp contains. Before running the code, what we can do is we will close it, right.

Press any key to continue, right. Now, we will see, maybe we can add something also. I have to open example_cpp.txt. Let us open it here. In the directory, we have to open example_cpp.h. Okay, shall we add some lines?

Welcome to file handling. Next line, let us say welcome to OOP. I am filling the file. All right. It is a text file.

Welcome to OOP. Welcome to IIT Roorkee. All right. Welcome to IIT Roorkee. The next line may be hello world.

All right. So, assume that now we have to save it. Let us save this file. Yes, we have saved the file. So, now what can we do?

We will go to the program. So, you have five lines. Alright. So, these are the five lines. So, what will it do?

You have example cpp.txt and my file. If my file is open, if it is true, and while loop. So, what is the while loop doing? It is reading line by line. Right.

getline. myfile is nothing but the object, and line is the variable under string. So, you have a string. Right. Hello.

Welcome. Hello world. Right. Welcome to IIT Roorkee. Welcome to OOP.

Right. So these are all we have filled. If you look at the file. Right. So this is what we are going to do.

Line number 8. If you look, we are going to print this. Right. So we are going to print this in the console. Right.

We are going to print. So now we will execute this code. Compile and run. Right. So it is running successfully.

And here you go. All right. So, you can see all the output. We have written writing. Welcome to file handling.

Welcome to OOP. Welcome to IIT Roorkee and hello world. All right. So, these we have written in example_CPP.txt file. So, this means it is reading perfectly well.

This code is reading the file perfectly well, and then I am trying to write every line I am going to write in the console. So, you can see in the console all the file lines have been printed. So, this is how we can read and write. So, we can do read and write. So, in the previous example, we had seen writing to a file.

And in this example, you have understood how we can read. It is reading from the file. Whatever example underscore cpp or example dot txt. And then we are able to read it line by line. Read the data or the text line by line.

And then we are printing those in the console. That is what we have included IO stream also see out line so that was printed right. I hope it is clear now how you can read and write. So this is the basic you have to do that. So then you can develop. I am giving an assignment.

So what you can do is read two numbers, all right? Read two numbers, let us say 10 and 5, and then you do the multiplication or the division, right? So now you know that. The basic logic you know that, so 10 and 5 you are reading, and then there itself you can write the output. Now you understood from here, alright, how to read and how to write. So this is what, I mean in the previous case, so you have the ifstream class, ifstream class, input file stream class, we have an object called my file.

So the object is nothing but myfile, alright. So it is associated with the file example.txt. Right in the code, we have taken example_cpp.txt, correct? So here you go. When you have an if block, the if block we had seen in the last program itself, that is a case of writing. myfile.is_open, if this is true, that means myfile is invoking is_open, correct? If this is true, then it is going to the while block, right? So the while loop, what it is doing, it is reading line by line, right? And it is reading line by line from the file using the getline function, right? getline, inbuilt member function, correct?

So myfile is an object, and line is a variable under string, correct? So then each line. Right, so that is stored in each line, in the sense I am talking about the file, right? So these are all stored in the variable line. For example, the first one we had seen, hello. So initially, the line will be nothing but hello, correct? So the line will be nothing but hello. That is what here we are doing, getline(myfile, line). So myfile is an object, the line, the first line, hello. So now line is nothing but equal to hello.

So that is being printed. And whatever be the second line. Let us say welcome to OOP. Assume that. It is the second line.

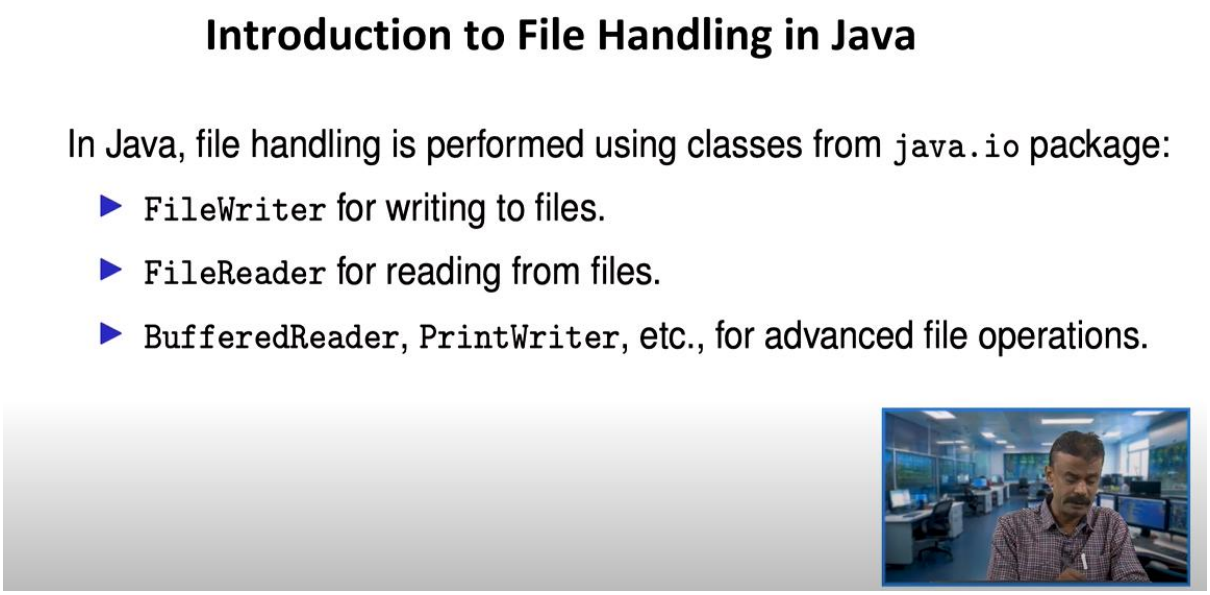
Now, while. Right. Myfile is reading the next line. So, the next line is being read. Correct.

So, whatever. Let us say, 'Welcome to OOP.' So now, the line is equal to 'Welcome to OOP.' And that is being printed. The next line is being printed.

And you can see all are being printed in the console. Right. So, you know how to use cout. Right, and then it is being printed in the console. In fact, we had seen in the console it is being printed. Right, so now after reading, the file should be closed. So, this will be done using the close function. That means the myfile object will invoke the close function, so the file will be closed. Right.

So, this is how we can read from a file in C++. Now, you are familiar with reading and writing in C++. So, in Java, right? So, a similar one we can see in Java as well, right? So, here, what we have to do is use the java.io package, the input-output package, right?

So, here, we use FileWriter for writing to files and FileReader for reading from files, exactly similar. The only thing is the syntax will be different. In fact, the keywords will be different. So, once you know one language, you know how to handle files.

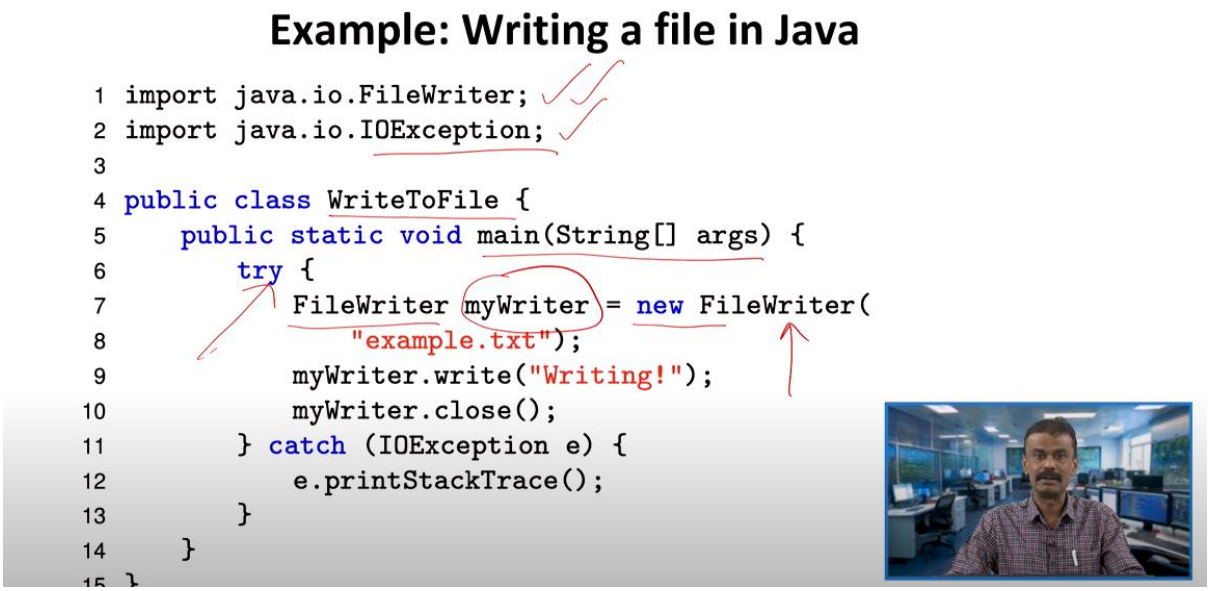


So, since this is object-oriented programming, we keep on using C++ and Java. And then, we have several advanced file operations. So, here you go, BufferedReader. which is an advanced file operation, and similarly, PrintWriter, which is also an advanced one. So, let us consider the same similar one: how we are going to write a file in Java.

So, we import java.io.FileWriter and here import java.io.IOException. So, IOException means we are going to use try, throw, catch. This we have already studied. So, now you have a public class which is a driver class WriteToFile. So, now you have a main program.

So, here you have a main function, main method. So, I use try. IOException is there. Therefore, I use try. I will try to create myWriter.

That is an object whose class is FileWriter. Whose class is FileWriter. And then dynamically allocating memory with the constructor, so it is a one-argument constructor. You have example.txt, that means, similar to C++, your myWriter is going to associate with the example.txt, correct. So, if it is able to open the file, all right, if it is able to open the file, then there is no problem. The flow will continue; it will go to line number 9.



Otherwise, if there is any problem, assume that the file does not exist or there is a problem creating the file. So, in that case, the exception will be caught over here, right? So, this can be done in the next case also, even writing. Assume that there is a problem, right. So, the exception will be caught as IOException under E. So, E is invoking printStackTrace, right. E is invoking the function printStackTrace, right.

So, the meaning here is that printing the stack trace is nothing but you are not able to access or I am not able to create a file or I am not able to write it, alright. So, if these kinds of exceptions occur, right, it is an inbuilt correct. So, this will tell you that you are not able to access the file or open the file, or you are not able to write it in the file, right. So now, see over here, the next line assumes that you are able to access the file or you are able to create the file. example.txt, right.

Now, this is myWriter, like whatever we had seen in C++, similar to C++. So, this is invoking write, right. It is invoking the write method where you have one parameter,

writing, right. So, we can do like what we have done in C++. You can put something like, write 'Welcome to IIT Roorkee.'

Hello world, right. Hello world in Java. So all these things we will do while seeing the example. So this writing will be there in the example.txt. So when I open example, when I run the code, assume that I can access example.txt.

There is no exception. The flow will go. Line number 9, in this file, you are writing something. So the word is writing you are doing. So you can write whatever you want in the file.

And then line number 10, you are closing the file. myWriter will invoke close. Exactly almost similar. So some keywords are different. Otherwise, whatever we had seen in the case of C++, this will work in Java as well.

So, this is what the explanation is. So, we have the combination of FileWriter and IOException, lines 1 and 2. So, you have to import these packages, right? So, once you have done this, now you are trying to write the character data. Character data is nothing but writing into the file.

The name of the file is example.txt. So, in case I am not able to open the file or there is a problem with the file, there will be an exception. An exception will be thrown. And then you can see that, I mean, it can be handled. All right.

It is catching some exception. So, that exception will invoke printStackTrace. All right. So, because we have to come out of this code. All right.

We know the usage of try, throw, and catch. All right. So, printing the stack trace. So, that means I am not able to access the file name. example.txt.

I am not able to open the file named example.txt. So, that is the meaning. So, now we will see this example in Java, right? So, let us open the Java console. So, here you can see we have created the package oops_nptel.

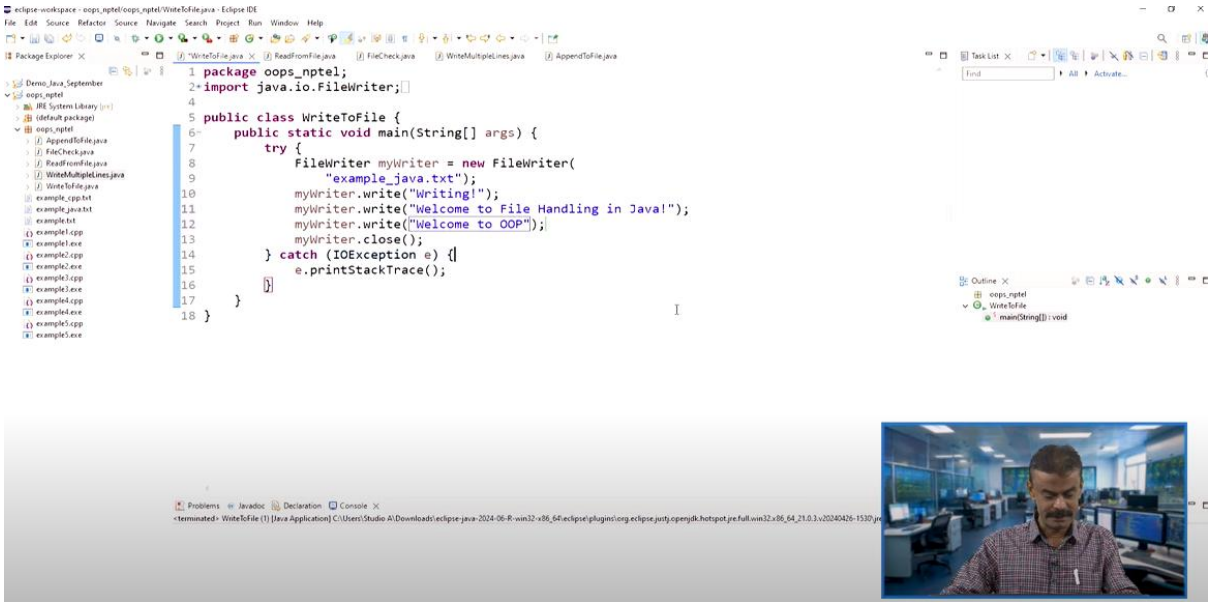
Here you have imported java.io.FileWriter. The public class name is WriteToFile. I have That is the driver class. And then I have the main method. My writer is nothing but an object of FileWriter.

So if I am able to access example_java.txt. Right. Or able to create. Then there is no problem. If you are not able to create.

Then an exception will be caught. Or even if you are not able to write. An exception will be caught. Right. Or even if you are not able to close.

There are reasons. Right. There are. I mean, you might have observed this. Sometimes you open, let us say, some file.

Alright, so there is a possibility that you, I mean, it is very difficult to close it. You have to exit; you have to kill. Alright, all these things are nothing but exceptions. Alright, so assume that when you are trying to access this example_java or you are trying to create this, if you get an exception, it will be caught in line number 12. Right, and it will invoke printStackTrace, correct? So now, what we will do is we will run the code, right? I hope you understand. We will run the code and example, yes, writing.



So maybe what you can do is we will write some more. Welcome to, we will just copy this: myWriter.write, put it, 'Welcome to file handling in Java.' I am just writing more, right? So that you can understand now, we are writing. Welcome to file handling in Java. Okay. Next line.

Maybe I think you have to put a new line. Otherwise, let us see. It is not a problem. Next line. Welcome, myWriter.

Write. We will just copy and paste it. Welcome to OOP. myWrite or control V. Right-click paste or in the file we have. Okay.

Okay. We will write it. Dot write. Welcome to OOP. Fine.

Right. So, I made three write statements. Correct. So, let us see. We will run the same code, and then when you open this example_java.txt, maybe one more time we have to open it, run the code.

We have to run the code. Fine. So, now we will open it. Example. Yes.

Since we did not put the new line. Alright. So, it is printing in the same line. Alright. So, this is how

In fact, we are getting all the statements. Writing, welcome to file handling in Java. Welcome to OOP. Okay. So, I hope you understood how we can write to a file in Java.

All right. So, what we have done over here. So, here you have the public class WriteToFile. All right. This is the driver class.

And the main program is there. myWriter is nothing but an object. And you have FileWriter, which is nothing but a class. Right. So, we can create or access example.txt.

If I am able to open the file. Well. I will not go to that exception. Correct. And then I will do the sequence of writing statements.

Right. So myWriter will invoke write. So, the writing statement. And once it is over, you have to close the file.

So, this is the writing a file in Java. All right, in the next class, what we can do is we will do reading a file in Java. Right? So, reading a file in Java, we will see in the next class. Thank you.