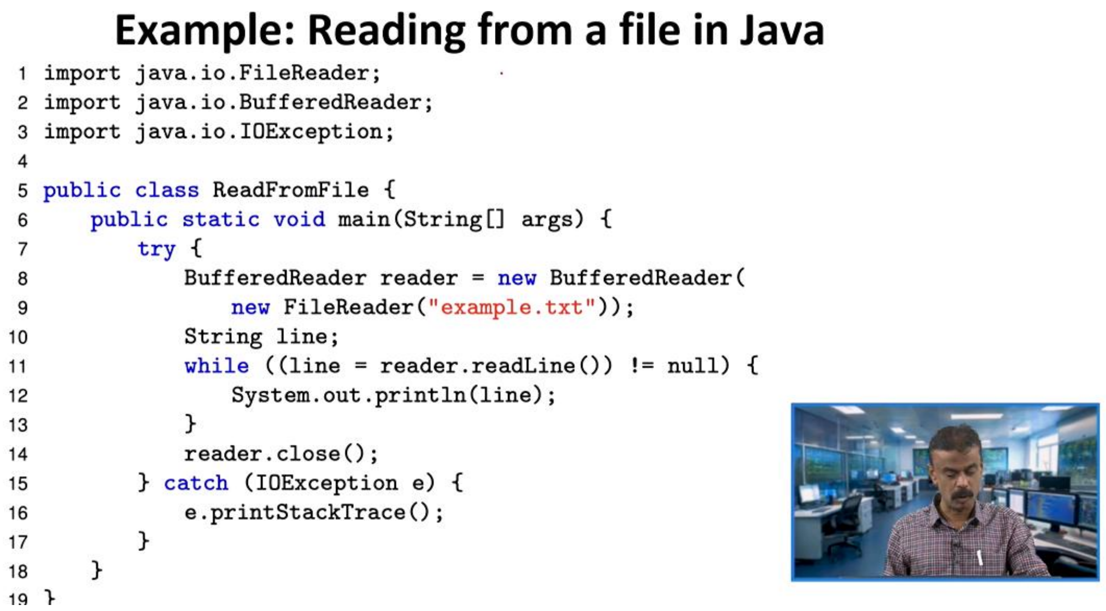
FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture32

Lecture 32: File Handling - Solved Problems



Example: Reading from a file in Java

```java
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;

public class ReadFromFile {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(
                new FileReader("example.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

So, welcome to lecture number 32, File Handling. So, in the last class, we saw writing a file in Java, right? So now, what we can do is we will read from a file in Java, right? So, like what we have done in C++, we are reading line by line and then what we are doing is we are writing in the console, correct? The same thing we will do here.

So here, we have to use the package, right? IO, right? Input Output. FileReader, right? You have BufferedReader and then IOException, so that means I am going to use try, throw, and catch. So now, your read from file is the driver class because you have main under this, so now I use try, like exactly what we have done in the case of writing, so try.

So, the reader is the object under the class BufferedReader. So, you are allocating memory, right, with the constructor BufferedReader, right? So, BufferedReader, if you look, you will have one function, right? So, this is nothing but the constructor. So, the constructor you have is new FileReader example dot txt, right?

So, slightly complicated syntax, but you will understand right under the BufferedReader constructor you have one function. So we use new again, right? So, FileReader, correct? So, FileReader example.txt, okay. So now, in fact, you can see you have imported FileReader, correct? So, under FileReader, this will be the constructor, right? FileReader class, and here you have the FileReader constructor. So that means you are going to handle example.txt. So, I have string line; line is a variable. Exactly what we have done in the case of C++, we are going to Java.

Slightly complicated syntax, right? Because you have to use all the keywords, right? The FileReader, BufferedReader, etc. Now, while line is equal to, right? Line is a variable.

In C++, it was very simple, right? Now, the reader will invoke, the reader object will invoke the readLine function, right? So, readLine. So that means when it is invoking and then you are assigning that to line, for example, assume that you are reading the first line, alright? Welcome.

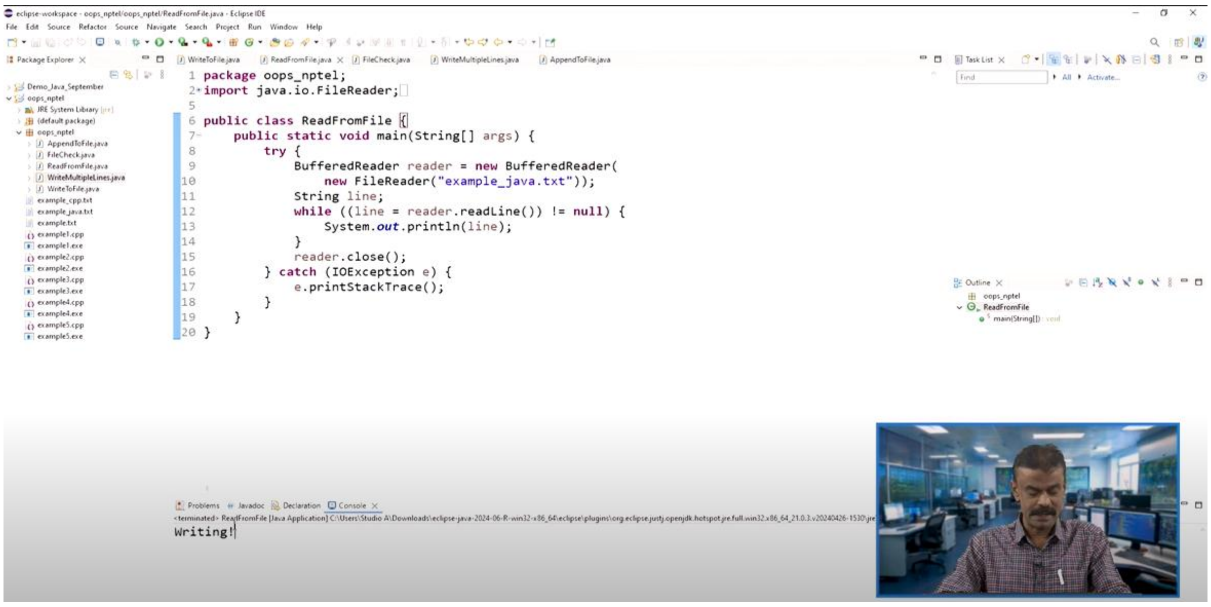
The example.txt, the first line is welcome, alright? Which is not equal to null, alright? Which is not equal to null. Assume that only one line is there. It is not null.

So it is going and printing System.out.println, I am printing the line, alright? So whatever you are reading in the line, you are storing in or copying in line, line is the variable under string. So if it is not equal to null, that means assume that I have the first word, right, or first line, hello, welcome to IIT Roorkee, right? If I am able to read that, that is not equal to null, right?

When it is not equal to null, System.out.println line, so that particular line will be printed. In fact, in C++, we have not equal to null. It is understood, right? Even when I put like this, while line equals to reader.reader, readLine which is equivalent to not equal to null, right? So that means if I am able to read the first line, right? Assume that there is only one line.

So that particular line will be printed; that is your syntax of line number 12: System.out.println. That particular line will be printed. Assume the second line; I do not have anything, right? So, that means it is equal to null. So, it will come out of the while loop, right? So, once it comes out of the while loop, you have to do reader.close.

Otherwise, if I am not able to access the file or open the file, example.txt, correct? So, it has to give the exception, and the exception will be caught, right? e is the exception under IOException. So, e will invoke printStackTrace, right? So, that means you are not able to; it is a built-in function, right, a built-in method, correct.


```
 1 package oops_nptel;
 2 import java.io.FileReader;
 5
 6 public class ReadFromFile {
 7     public static void main(String[] args) {
 8         try {
 9             BufferedReader reader = new BufferedReader(
10                 new FileReader("example_java.txt"));
11             String line;
12             while ((line = reader.readLine()) != null) {
13                 System.out.println(line);
14             }
15             reader.close();
16         } catch (IOException e) {
17             e.printStackTrace();
18         }
19     }
20 }
```

```
<terminated> ReadFromFile [Java Application] C:\Users\Studio A\Downloads\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre
Writing!
```

So, it is not able to access. So, therefore, right, you cannot do anything with example.txt, right. So, where you can see the example, we will see an example, and then again I will just explain a few things. So, we will go to Java, and we will see the next example. ReadFromFile.

So when I run the code, it is the same as what I explained. So, in fact, if you remember, we had this file. Maybe we will modify the file. Just to open this example.java, example_java. We have put it in file handling.

Here it is. Yes. So, what can we do? We will open example_java. So, now I will do it line by line.

Well, writing. Welcome to file handling in Java. Or let us have only one line. Okay. Only one line now.

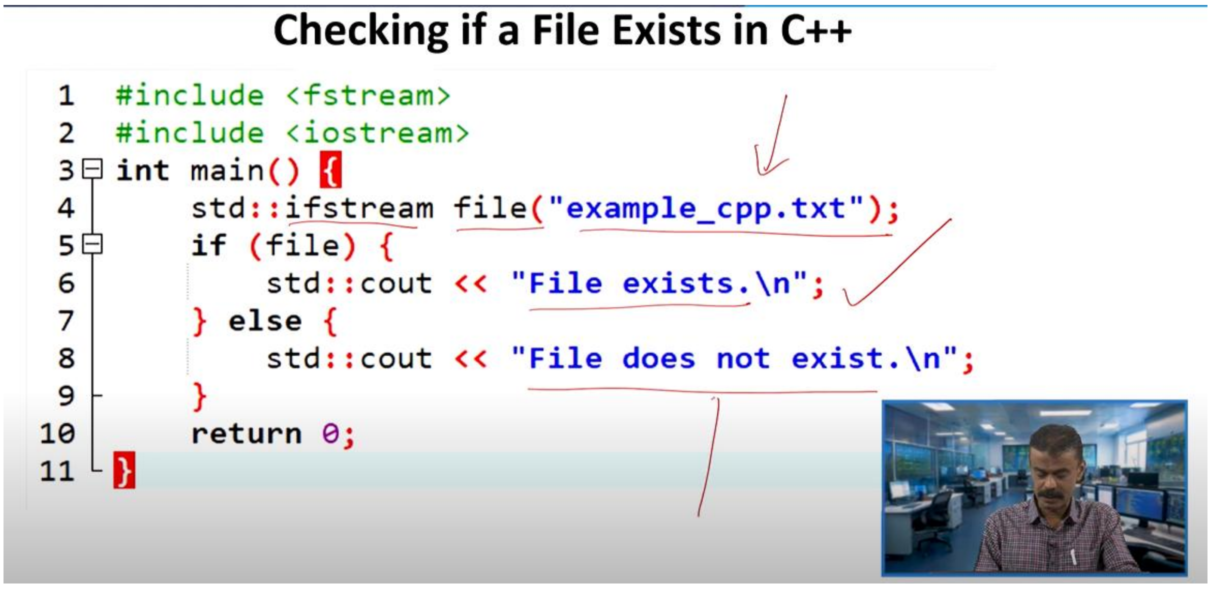
Maybe we will see. We have a program. Alright. There, we will have multiple lines also. So, writing.

File saved. Let us save this file. I saved it. Okay. So, now what can we do?

I have only one line written. Alright. So, now we will run the code again. Run. Fine.

So, you can see. Alright. In the console, we have the output. So, the output is nothing but writing. Right.

We are getting writing. We have only one line. So, that means while line equals to reader.readLine. If you look at line number 12, the object reader is invoking the readLine function, right? The readLine method is an inbuilt method.



Checking if a File Exists in C++

```cpp
#include <fstream>
#include <iostream>
int main() {
    std::ifstream file("example_cpp.txt");
    if (file) {
        std::cout << "File exists.\n";
    } else {
        std::cout << "File does not exist.\n";
    }
    return 0;
}
```

So, writing is only one word or one line. So, it is able to read and then it is storing in line. That is the meaning. In fact, we have used the getline in C++. So, that means it is copying.

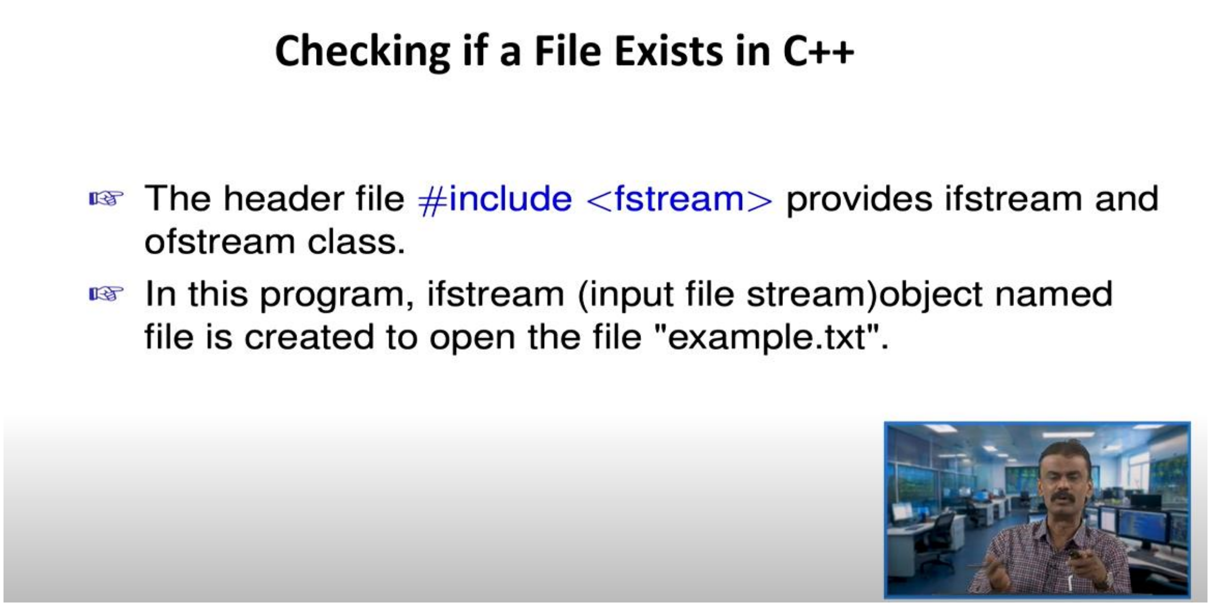
So, the readLine is copying into line, which is not null. The first line I am able to read, which means it is not null. So, now line number 13, System.out.println(line). So, line here, in this case, line is equal to writing with an exclamatory mark, right. So, that is being printed.

So, the second line does not have anything null. So, when it is null, it is coming out of the while loop, and then it is closing, right? It is closing the file. Exactly, I mean, if you want to read something in a file, what will you do? You open the file, read it, and then close the file in the general case. Similarly, you have to do that, alright.

So here you have the FileReader, alright. It is trying to open the file. If it is not able to open the file, then there will be an exception, alright. So, example.txt, and then after opening the file, there is one line. So that line is being read and then it is being stored in line, right.

So the reader is invoking. The readLine function, right? The readLine method, an inbuilt method, is written with an exclamatory mark, so that is not null, which will be stored in the line. And then you are printing the line, so in the console, you can see the output, right? So this is how reading from a file in Java works. Last class, we saw writing. And here, we have started with reading from a file in Java. So here, the FileReader is nothing but a class, right? So the FileReader. It is nothing but a class.

And then the file has, let us say, character data. The file has the character data. And then you have seen the BufferedReader. That is also a class. Under BufferedReader, you have an object, reader.



Checking if a File Exists in C++

- The header file #include <fstream> provides ifstream and ofstream class.
- In this program, ifstream (input file stream)object named file is created to open the file "example.txt".

So then, what is happening when you are creating an object, reader, under BufferedReader? So you are creating memory, new dynamic memory, where you have the constructor, BufferedReader. So under this, you have another constructor, FileReader, because FileReader is a class. So which is handling example.txt, alright. So which is handling example.txt, then we had seen the example, right.

So if it is not able to open right, if it is not able to access, then there will be an exception, correct. So the exception will be thrown, and it will be caught in line number 15. And then it will invoke, alright. So that means, alright, I am in print stack trace, so that means it is printing the stack trace, so that means it will say not able to access the file, something like that, alright. So now we are able to access the file, and then we can read only one line, as we had seen, and that line

we are able to print also. So, now we will check if a file exists in C++ or not. We will see the simple C++ code. Again, I am going back to C++, alright. So, let us include fstream, the file stream, and then include iostream.

Now, we have a file, right, which is an object under the ifstream class, correct. So, which is handling? The file is handling example_cpp.txt. It is like a constructor. So one argument, the argument is nothing but it has to handle example_cpp.txt.

If file, that means if the file exists, it is going to print 'file exists,' correct? Otherwise, if the file does not exist, this is what I kept on telling, right? We have used the try-throw catch in both, I think, C++ and Java, particularly in Java. Assume that the file does not exist, and then you want to read something from the file. If the file itself does not exist, then it will throw an exception.

So now, here in this program, what we are doing is checking whether the file exists or not. So, if file, that means if the file exists, that is true. If file, that is true, then it will print 'file exists.' Otherwise, it will throw 'file does not exist.' It will tell 'file does not exist.'

So, a simple program, right? So, let us go back to the C++ console and assume that we have 'example_CPP.txt.' Just check, right? So, here we have 'example_C.' Yeah, we have, right? One we will do with the file.

Another one, we will delete it and see. We will run the code, right? So, here you see. It is giving the output: file exists. The file is there.

Therefore, we are getting 'file exists'. Now, press any key to continue. So, now what will we do? We will delete it. Shift delete.

Alright. So, the file is not there. Correct. So, now what will we do? Again, we will run the code.

Execute. Run the code. And you can see that the output shows the file does not exist. Correct. So, I hope it is clear.

It is a simple program. But we are slightly going into the advanced level. Whether In that particular directory, the file exists or not. So, we had seen we had a file, right? And then we opened it. In fact, we have seen the file exists, right? Then we deleted it, so now the file does not exist. In fact, we have seen both outputs, right? One with the file exists,

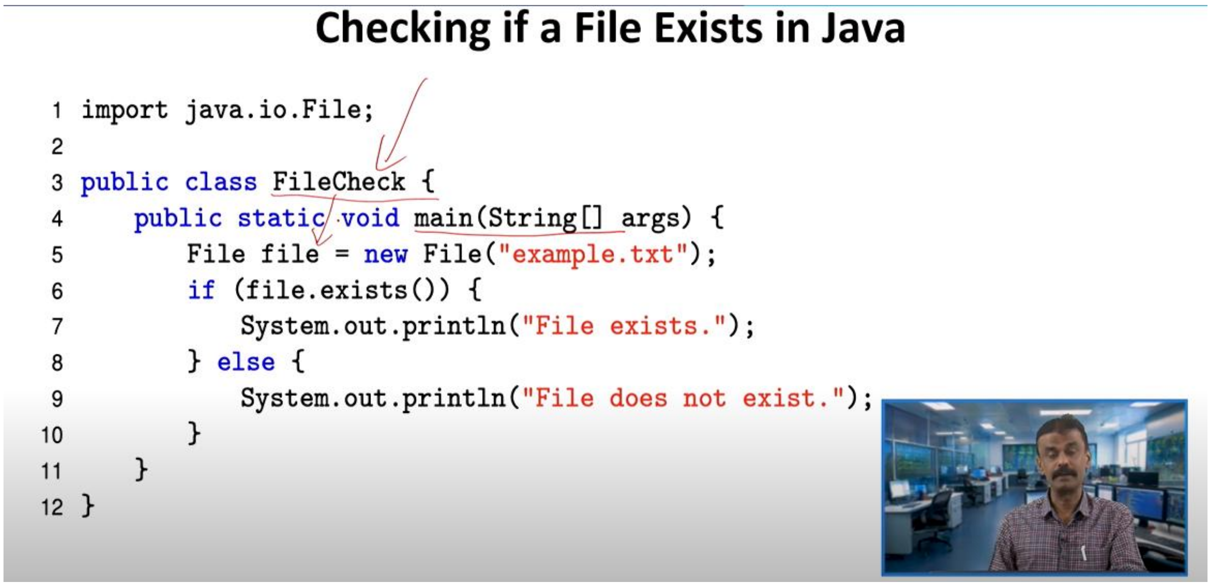
So that means the file exists output we got. Another one, we deleted, so that means no file is there, so we got the output file does not exist. I hope it is Clear now, right? So here we have the header file include fstream, alright. So this has both input and output, input file stream, and as well as output file stream, correct. And then you have the object name file. The object name file is under ifstream, alright.

So If the file already exists, correct, then it is able to open, right? So that is what the file, correct, the line number 5. It can access the file or open the file. So therefore, initially, if you recall, we got file exists, right?

Otherwise, assume that there is no file at all, correct? So you can do the same thing in the previous program that we have written: try, throw, catch, right? So what can you do? One, you can try with the file. Let us assume you have the words, you have the lines, a string of lines, right?

And then you read it. It will work. Now your exception should be thrown. Then what do you do? You delete that file, right?

And then you can run the code. So now this program will help you with the previous programs, right? Suppose you are going to see the usage of that exception, delete the file and see, right? You will not be able to access that. So then it will throw the exception, so here in this case, we are handling it without an exception.



Checking if a File Exists in Java

```java
1  import java.io.File;
2
3  public class FileCheck {
4      public static void main(String[] args) {
5          File file = new File("example.txt");
6          if (file.exists()) {
7              System.out.println("File exists.");
8          } else {
9              System.out.println("File does not exist.");
10         }
11     }
12 }
```

So here, there is no file at all; the file does not exist. So if the file will be false, it will go to the else part in line number 7, and 'file does not exist' will be printed, right? So both cases we have seen, so this is how We check if a file exists in C++. Similarly, we can try it in Java, right? So here, we are including the input-output file, right? Java input-output package dot file. And you have a public class FileCheck, right? Which is a driver class, and then you have the main program. So here, I have created an object small f file under the class file, right? The dynamic memory will be allocated with the constructor File, and then you have a one-argument constructor example.txt. So that means the file is having access to this example.txt.

Let us assume the same thing we will do, right? Assume that example.txt exists, right? So let us assume in that directory, example.txt is available. If it is available, 'file exists' will be printed, right? In line number 7.

If it does not exist, 'file does not exist' will be printed. It is not very difficult code. So, whatever we have done in C++, you can extend it over here and then try to print either 'file exists' or 'file does not exist'. So, if it is available in the directory, 'file exists' will be printed; if it is not available, 'file does not exist' will be printed. So, go to the Java console, alright? Here you go, the class FileCheck, correct line number.

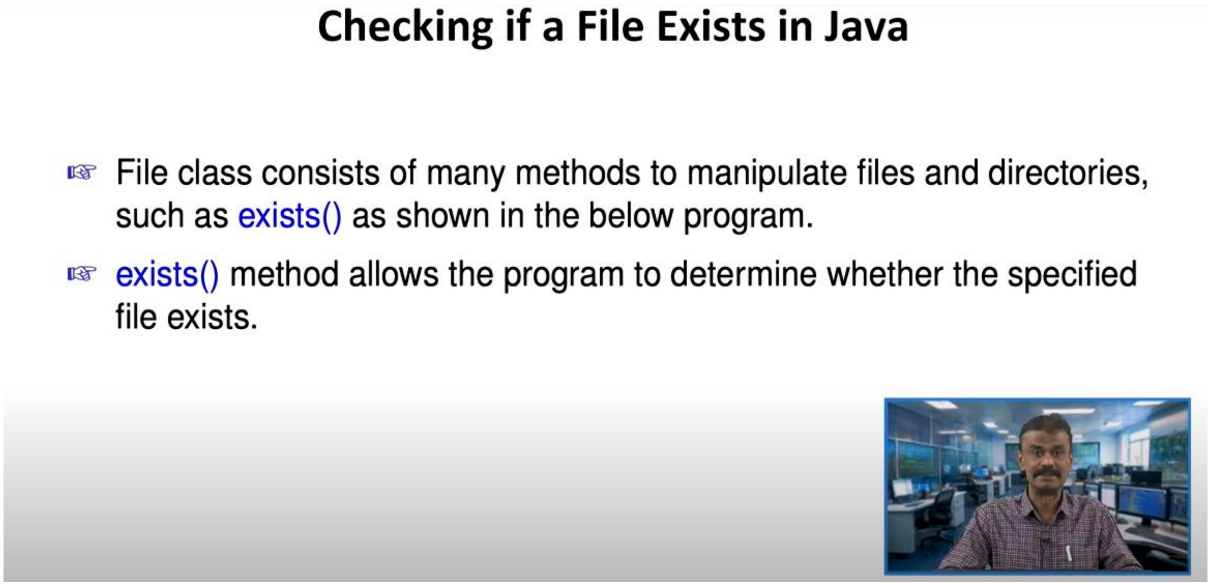
And then you have main. Here, what we are doing is example_java.txt. We will see in the directory whether the file is there or not. Yes, it is there. Example_java.txt.

Can we open this? Yes, you are seeing writing. Okay. So, it is there. If it is there, let us run the code.

Yes, the file exists. You can see the output file exists. So we can see the file, and it is saying that the file exists. So now, similarly, what we are going to do is we will delete the file, alright. So here we are deleting, and the file does not exist now.

So again, we will run the code, alright. So it is saying the file does not exist. Is it clear now? So both C++ and Java we have seen. So in one case, what we have done is the file was there, and then it was printing that the file exists.

Then we deleted it, and then, alright. So if you look at line number 7, the if file exists function, right, the file is deleted. Invoking the exist function, right, the exist method, if it is existing, that means in the directory if the file is there, in fact, we have seen the file exists, and then what we have done is deleted it. So deleted means it does not exist, so that will be false. False means it will go to line number 10, and then we have got the print that the file does not exist, correct.



Checking if a File Exists in Java

- File class consists of many methods to manipulate files and directories, such as exists() as shown in the below program.
- exists() method allows the program to determine whether the specified file exists.

I hope it is clear for everyone, right? So in the case of Java, what we have done is we have used the input-output package with file, right, the file handling. So here you have the method calls exists, alright. So when I want to use exists, I have to use the file. I have to use the File class.

So the File class has several methods. And then I want to check whether the file exists or not. So I have to use the method called exists. So we have seen both cases. In the directory, the file was there.

So that means if it exists, in this particular line, it is true. When the file is invoking exists, line number 6, that will be true. Assume that it is available in the directory. So then we have got the file exists output. If it is not available, then file.exists, right?

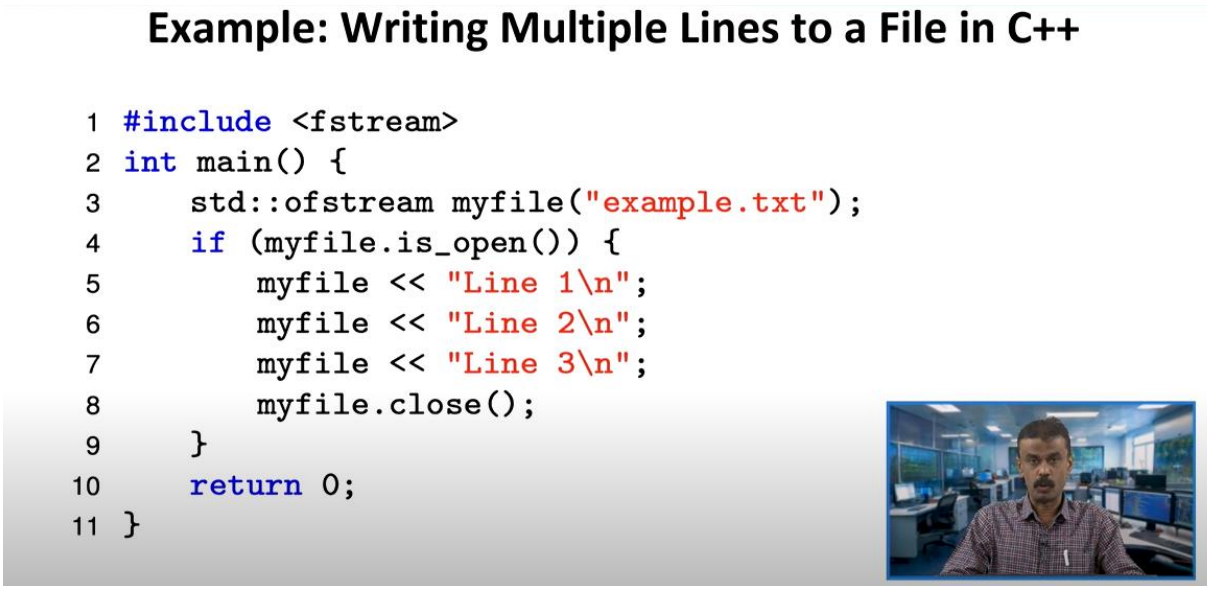
File is invoking exists, it will be false. So false means line number 9, you will get the output, file does not exist, right? So that means when I want to access this exists file, I need this java.io.file. So this is how you can handle. So reading, writing and whether the file exists or not, both Java and C++ so far we have seen.

So this almost we have already done. So one of the cases we have not put the new line. I think it is in Java. But what we are doing now, so again the writing multiple lines. If you want to write multiple lines to a file in C++, I include file stream.

to the main program and I have myfile object whose class is ofstream output file stream and then you have the constructor that means myfile is accessing with example dot txt alright. If myfile dot is open if this is true if it is able to open if myfile object can able to write myfile is invoking is_open if it is true. Then the following line so it is reading right myfile line 1 new line right it is going to write so it is going to write line 1 right so then it is going to write line 2 and the next line it will be line 3 and then there will be empty right because you are going to write only 3 lines. In fact we had seen myfile writing if you recall only one line we have written and then in fact we had seen some more example also put some more hello world welcome to IIT Roorkee

Alright, so here I put slash new line. Alright, so that means it will open if I am able to open example.txt. So there, we are going to write line 1, next line line 2, third line line 3. So once it is done, we are going to close, right? Myfile will invoke close, so the file example.txt will be closed. We will see this example, alright? So same. So example_CPP.txt, alright? So let us run the code, yeah. So the program is running successfully, and if I open, yes, you can see the output: line 1, next line 2, and line 3, right?

So what we can do is we can change line 3 to, let us say, welcome to file handling instead of line 3. Let us say welcome to file handling. In C++, welcome to file handling in C++. Good. So when we write this now, I will run the code. Earlier, line 3 was there, fine. It will be overwritten. I will execute the program, run the program, compile and run. Yes, it is running successfully. Good. Now if you look at the output, I open it. Yes, I see line 1, line 2, and the third line you can see: welcome to file handling in C++. Okay. So this is how we can write multiple lines.



Example: Writing Multiple Lines to a File in C++

```
 1 #include <fstream>
 2 int main() {
 3     std::ofstream myfile("example.txt");
 4     if (myfile.is_open()) {
 5         myfile << "Line 1\n";
 6         myfile << "Line 2\n";
 7         myfile << "Line 3\n";
 8         myfile.close();
 9     }
10     return 0;
11 }
```

I hope you are slowly familiarizing yourself with the file concept in both C++ and Java. So here, what is happening? The if statement is checking is open. So if it is able to open, that means it is true. Correct?

And then it is writing line 1, line 2, line 3. So what are you doing? Suppose I am asking you to write this. Assume that you do not know C++. I am asking you to write.

Open the notepad. Write this. So what will you do? You will open the notepad. Open.

Right? So that is exactly what it is doing. Line number 4, myfile is invoking underscore open. So if it is able to open. Correct?

So, example.txt. If it is able to open, well and good. So then, in that file example.txt, I will write line 1, line 2, and line 3, or we have done 'Welcome to file handling in C++,' right? And then you have to close this; myfile should invoke close. This file, example.txt, contains line 1, line 2, and maybe line 3. In this example, line 3.

In the one we had seen, the last line is 'Welcome to file handling in C++.' So, we can write, right? So, if it is true, we can write, and then you have to do the close. Since, yeah, you can see the new line is appearing. In Java, we had seen, right, without a new line, right?

We have also written multiple lines without a new line, if you recall. So, then it is printing in the same line. Now here, in this case, we have \n, so we are getting the output, and then, right? So, in fact, when you open the file, we have got line 1, line 2, and line 3. And once it is over, right, the file should be closed. That means myfile should invoke close, so it will be closed, right. So, what we can do in the next class. We will see how to append to a file in C++.

So, this particular program we will see in the next class. So, what have we done so far? Reading, writing, and checking whether the file exists or not. So, these important programs we have seen. In fact, based on the examples that I have shown, you can practice it.

So, in fact, you can think of manipulating the values. That also we will see. Suppose you are reading the integers 10 and 5, all right. Now, the next line I want, I have the multiplication, you can write now, the multiplication of 10 and 5. The next line you have to get the answer 15, all right.

So, these things you can try. Now you know how to read the file, how to write the file, all right, how to check whether the file exists or not. So, whatever we have studied, all right, with the help of cin and cout. So, now here after we can think of Suppose I am reading a file from the file, so can I do certain manipulations?

Can I perform certain mathematical calculations? Or can I write the lines? Whatever lines are appearing, can I write them in the console? This means we know how to handle the file. In the next class, we will learn appending to a file in C++.

With this, I am concluding today's lecture. Thank you.