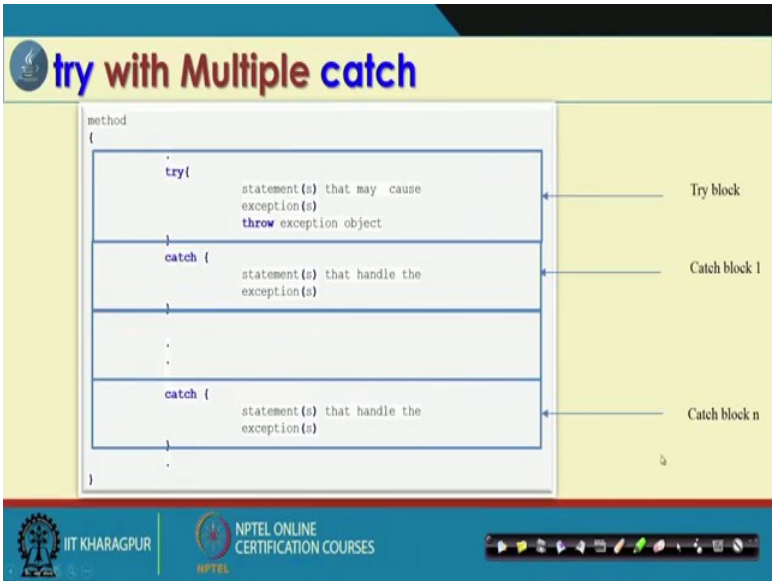
Programming in Java
Prof. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 24
Exception Handling - II

So, we are discussing about, Exceptional Handling in Java and in the last module, we have discussed the try-catch mechanism. In addition to this try-catch mechanism; there are many more features, so for the exception handling is concerned. So, today we will discuss a few of them and then other features advanced features also there. So, those things will be discussed in our next module. Now so, now we have discussed about try and catch block and there is another, this is simply what is called the exception handling mechanism. Now, apart from this simple there is also one more mechanism, it is called try with multiple catch.

(Refer Slide Time: 01:10)

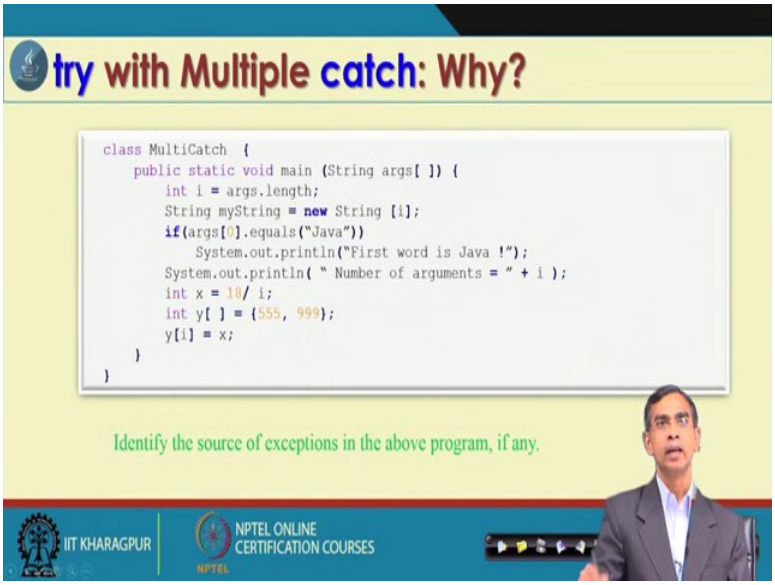


So, there are many features, try with multiple catch is the one. So, here the idea is basically, the methods, where we are interested to maintain the exception handling. So, suppose, this is the entire method and then, so try with multiple catch implies that there is only one try block and there is many catch block. So, two or more catch blocks are there then it is called the try with multiple catch.

Here, if the concept is that in the try block it will catch whatever the exception will occur, so only one try. So, try block will responsible for two catch all the exceptions and for each exception, we have to maintain the catch block. So, that the corresponding exception, for a corresponding catch block, we will take care.

So, this is the concept. So, a try-catch block is basically to indicates that if there are multiple errors or exceptions occurs in a program, they can be tried it with one try block and they can be caught with multiple catch block.

(Refer Slide Time: 02:23)



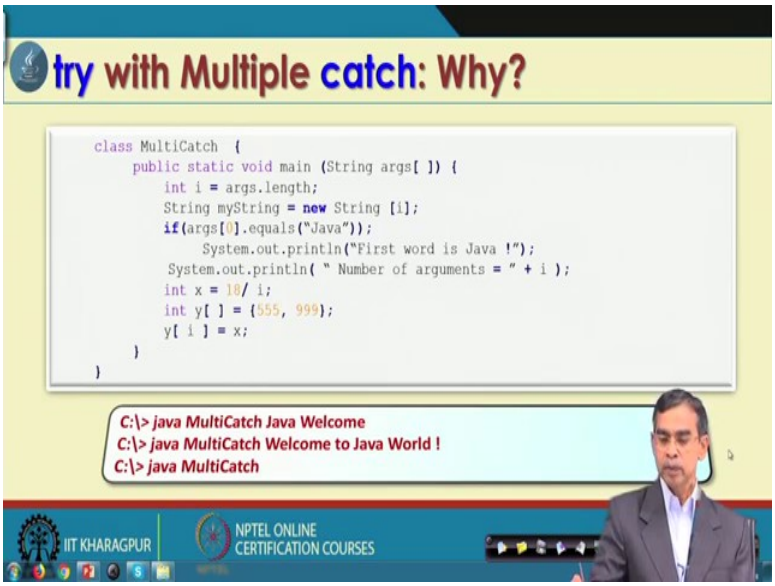
Now, here is an example, very simple example to explain how the try with multiple catch we will work. Now, let us look at this program. Now, we can see there are multiple points, where the errors may occur, as you can see in this program. So, it is basically considering the common line arguments. So, if the common line arguments are there. So, it basically has the args length; that means it, how many inputs that user has passed through the common line that will be stored here and after that we can see this is one point, where there is a scope of error or exception.

Why? This is because if a user does not provide any input, in that case, i equals to 0 and if i 0, then this is not a valid one. What is call the declaration, is called the null pointer exception. Now, another here is also if you see, there is also one scope of error, if a user does not pass any input then args 0 is also a null pointer and there is another also, you can see here, at this point also one error.

So, if i equal to 0 so, it is basically the arithmetic exception namely, the divide by 0 error and here also if you see there is a possibility of error, if the value of i is more than 2 because here i then an array is initialized with two elements. So, an array index is varying from 0 and 1. So, size of the array is 2 and then here if I suppose 3 or more than 1; that means, 2 then it basically, gives an error.

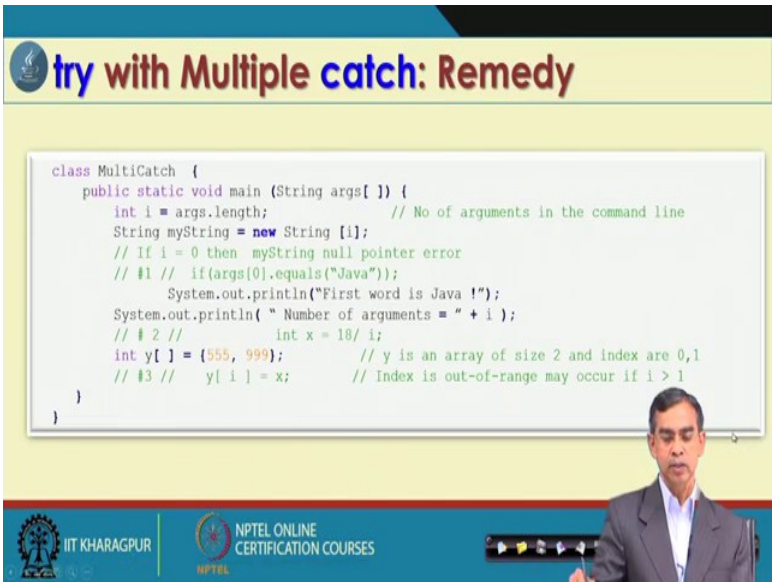
It is called the array index out of bound error. So, as you see this program can run for some input, but not necessarily it will run for all inputs. So, if the program does not take care about all those exceptional inputs, then it is the exception and this way this program is in fact not so robust.

(Refer Slide Time: 04:53)



So, there are few input as you can see, if you try this program with all these inputs you can see these are the; so, this is the one exception occurs in this input, here also exception occurs and here also the exception may occurs. So, there is a different exception that may occur for the different input like this.

(Refer Slide Time: 05:20)



Now, so, as a way out of these things, so they are basically, as I have mentioned again, let us specific as there is a one vulnerabilities, here is also errors may occur and there is also error occurs. So, there are three points where the error occurs, so multiple errors, multiple exceptions. So, here is the idea about how we can handle with a try with multiple catch. So, idea is that we can include all the statement where an exception may occur under the try block and for each exception, we can plan the catch block. So, here is an idea about these things, which is the program has been rewritten with a try with multiple catch. Now, let us look at this program little bit carefully, we will be able to see.

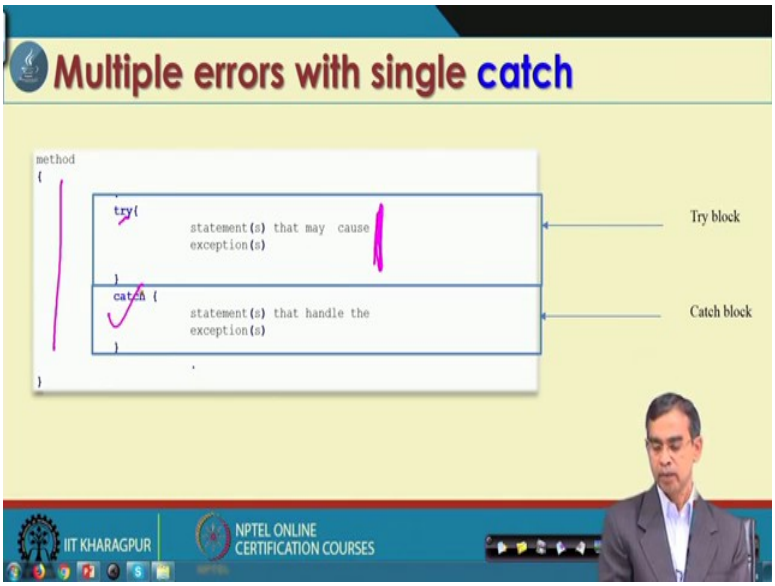
This is the program that where we have included try with multiple catch as you can see here is that, this is a basic methods portion, where the exception may occur at this point. This is the one point, this is the here and this is the these are the three points at the exception that may occur and to catch this exception, we have mentioned one catch. This catch is basically to handle the null pointer exception error mainly here and this catch is basically to handle arithmetic exception error divide by 0 error here and this catch is basically to handle the array index out of bound error here.

So, as you see for the different exceptions, we can plan the different catch block. Now, this is; obviously, not possible with a simple try-catch block, because it will catch only one exception, but there is a master exception we can consider. This is possible, but if you want to pinpoint or more specific about what particular errors or exception occurs

then definitely we should think for try with multiple catch. Now, so this is the one features.

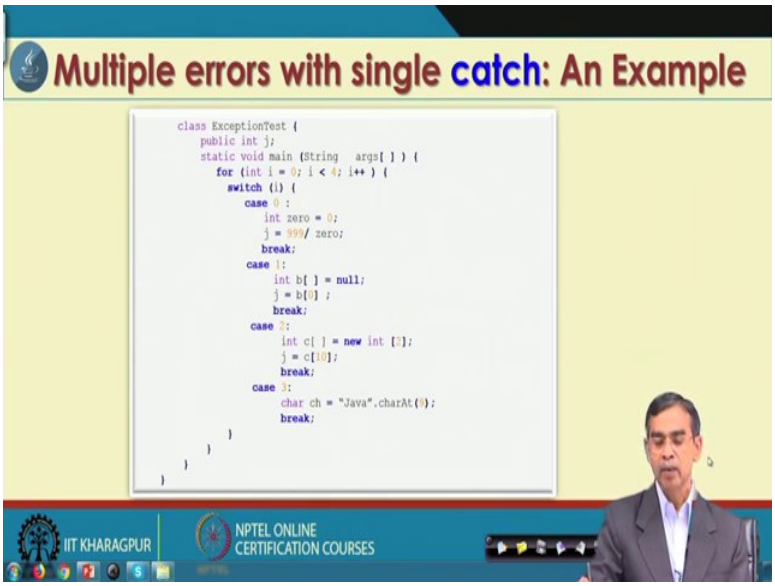
We have learned about simple try, try and catch, then we have discussed about try with multiple catch and also it is possible to catch by means of only single catch block, but the multiple errors are there. That is what I just wanted to discuss about these things in a different way, that we have discussed in the last. See, try with multiple catch by more errors, in the last example that we have seen, but we have planned more catch block, but here multiple errors, but single catch box. So, this is the one idea about so, here is one concept, the concept is like this.

(Refer Slide Time: 08:06)



So, this is the method as where the exception may occur and then these are tried, here in this part many errors may occur as in the last example that we have discussed, but there is one catch that can handle, whatever the exception it is there.

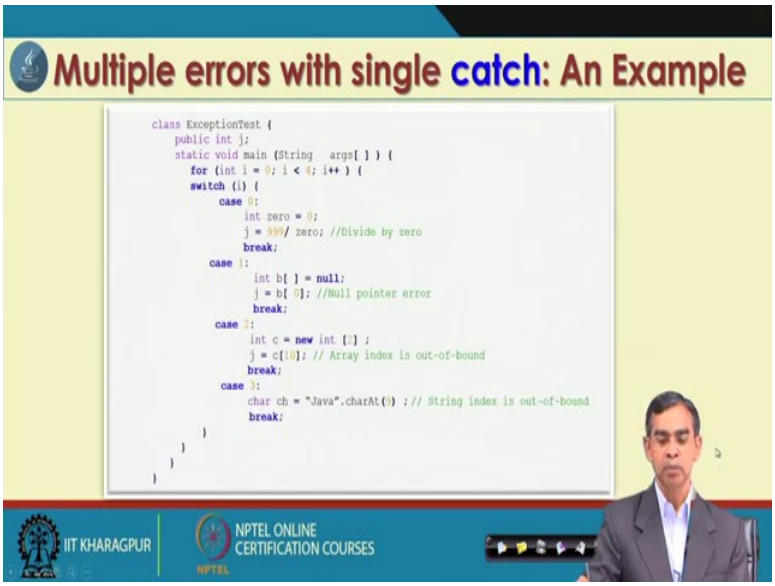
(Refer Slide Time: 08:29)



Now, here is an example that we can see, this is a very nice example to explain this concept very carefully. Now, here if you see so, this is the method, main method actually here. Now, under this main method as you see there are many points of exceptions that may leads. So, here is the one point of exception, because 0 equals to here divide by 0 error. Now, here also you can see as b is initialized is a null pointer and we want to have access b 0, the zeroth element.

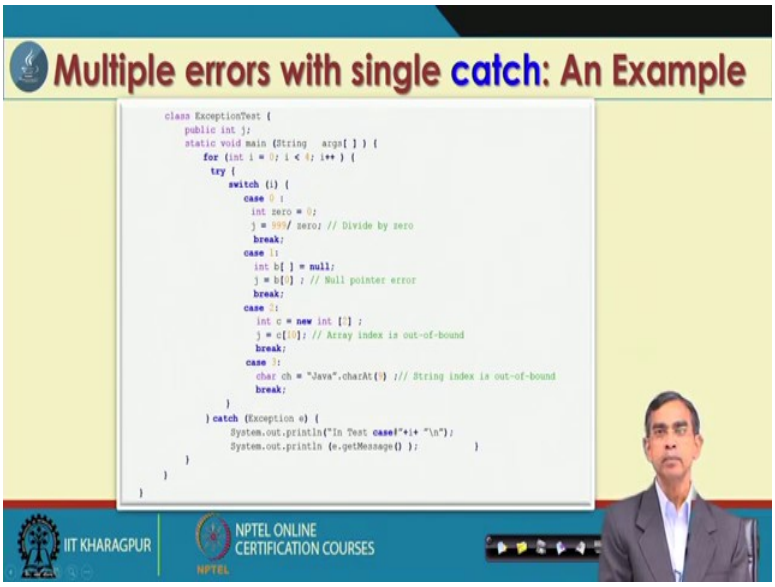
This is also an error and here also you see, here the array c is declared, which size 2 only, but here we want to access the tenth element, which is also not possible. And, here also if you see these a string and we want to find the character at ninth position, where is the string; has maximum up to third location, index is 3. So, here the string index out of bound, here the array index out of bound, here the null pointer assignment and here the arithmetic exception. So, there are so many errors that may lead to whenever we want to run this program.

(Refer Slide Time: 09:48)



So, this program, if we run then it will give always this error, because of static error that is there. Now, let us see what we can do so that this program will be more robust, more reliable.

(Refer Slide Time: 09:53)



So, this can be implemented with only one catch and then on try. So, an idea it is like this, here you can see. So, this is basically, as you have mentioned the try block. So, that whatever the error will occur, it will try to see whether if any exception error occurs or not and then here we can see a single catch. Now, in this catch, if you see, we have

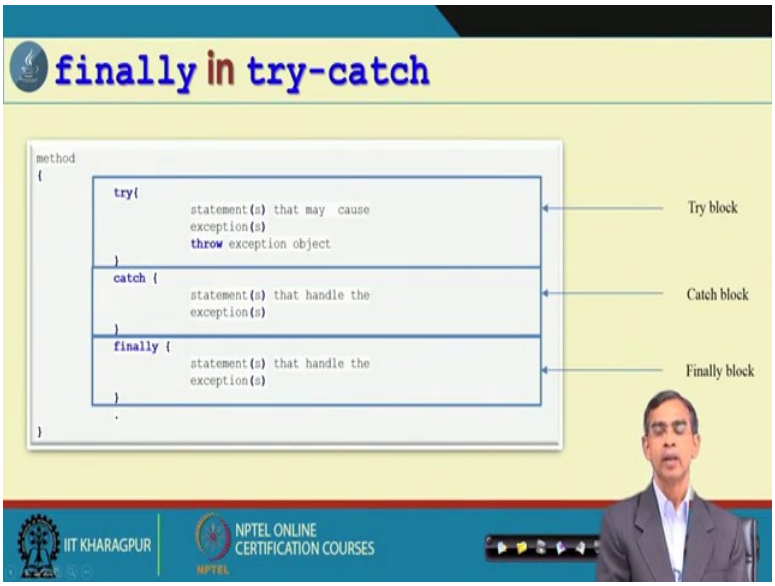
thrown an exception of the class exception. Now, in other examples; in the last example, we have mentioned the exception of a particular class, say; a null pointer exception or arithmetic exception or divide by 0 exception or like this ok.

So, here is basically, we will catch-all. This is basically the super exception class. So, it will catch any sort of exception whatever it is there, but if we write the catch, which says array index out of bound exception e, then only that kind of exception that this catch box will handle. But here the single catch will catch any kind of exception that may occur. So, it is a general, general one implementation and here also you can see for the different exception, the different messages meant already in the exception class, it is there.

So, this basically will print e.get message; that means, it will tell that ok. It is array index out of bound exception or what type of exception it is there. So, this is the idea about multiple errors which single catch. So, sometimes whenever we are not sure about, if any exception we will occur in our program, then usually programmer will prefer this kind of things. So, only say one try and one catch that catch will consider or handle only the object exception e.

So, this is the very simple one approach, of course, sometimes it is, but in some sense cases whenever we need say details error or exception mechanism, this is not so much preferable, anyway. So, this is the things that we have learn about simple try-catch, try with multiple catch, multiple errors with a single catch. Now, in addition to this, there is another concept, it is called the finally, associated with this try-catch block.

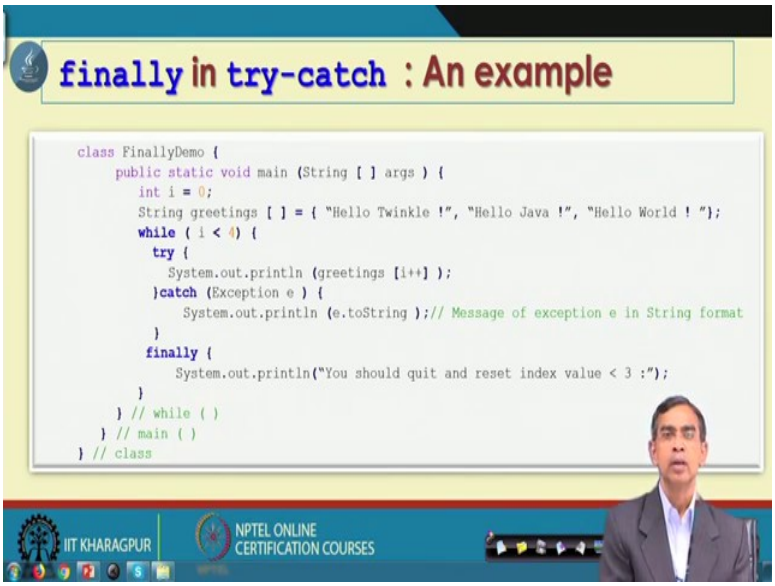
(Refer Slide Time: 12:18)



So, the idea about this finally, is that. So, try-catch we have already learned about it, then it will contains another block is called the finally. So, it is basically, the idea is that if there is any exception occurs or not occurs. So, in that case the final you will always execute for you. So, this means that it always execute this code.

So, this code basically helps the programmer that if an error occurs then, what is the remedy for that error? So, the programmer can have the flexibility to mention or inform to the user so that if this is the error occur so what the user should do like this and if we does not occur you can say that, the program is running successfully, this kind of things. So, this is an idea about finally.

(Refer Slide Time: 13:14)

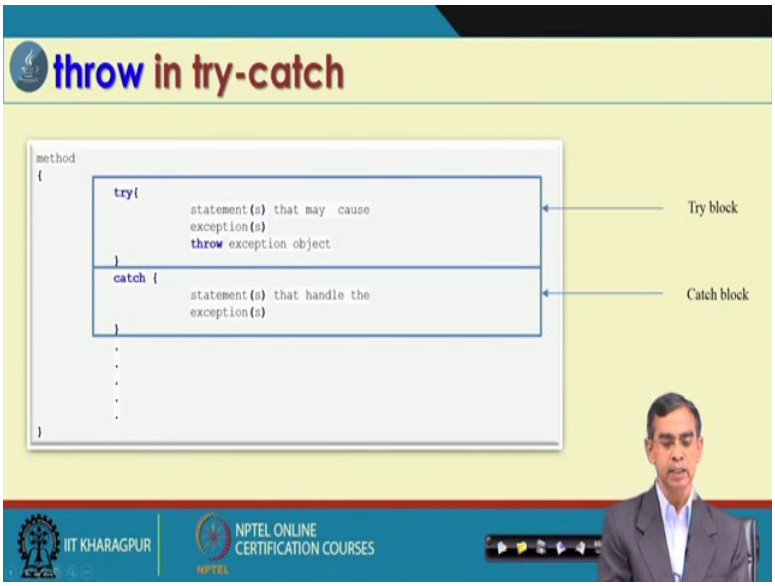


Now, let us have a small example to explain the concept of finally, here. Now in this program, if you see we have declared an array, the name of array is greetings, which will be initialized with three strings; "Hello Twinkle!", "Hello, Java!", "Hello World!" and so, it is basically greetings that have the index from 0 to 2 0 1 and 2. Now, here if we run this y loop then ok, for i equal to 0 1 2 it will work, but for 3 and after that is, it will basically give an error or exception, this is, because so i greater then, if i equal to 3 and it will try to find this one right. For i equal to 3 then greetings, because it is 0 1 2 maximum so 3, in this case, it will gives an error.

So, this loop if we roll, it will roll for first three input 0 1 and 2, but for 3 it will give an exception. Now, here the finally, catch is basically for each iteration it will, there is no exception so catch will not do anything, but finally, we will say that whenever i get less than, greater than 3 it will basically give that error. So, this is the idea about that the final you will, finally a block can be added in addition to try-catch block, to give more flexibility to the user. Now, Java also allows programmers to throw exception explicitly.

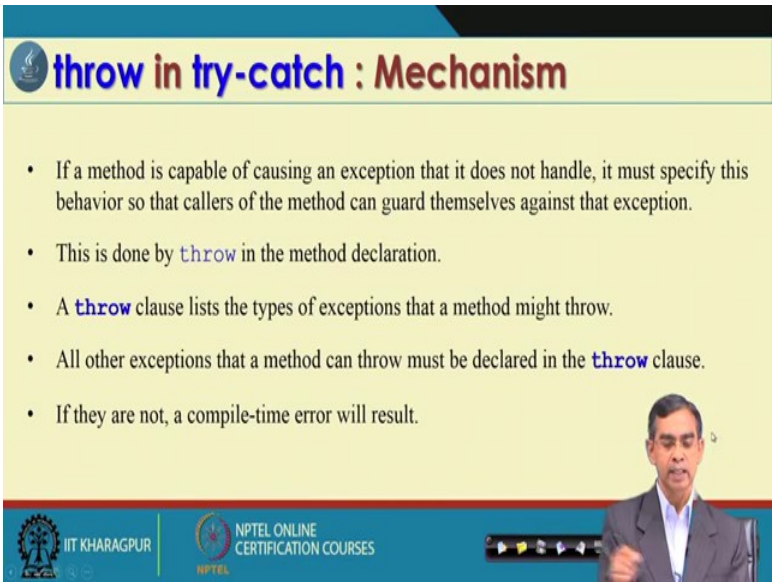
Now, all this example that we have discussed that, it is basically implicit throw. That means, whenever the exception, we will occur the Java runtime management manager basically, we will check it and then try for an exception, if it occurs and then catch it, but sometimes we can throw the exception of our own. So, this can be done by a throw clause.

(Refer Slide Time: 15:25)



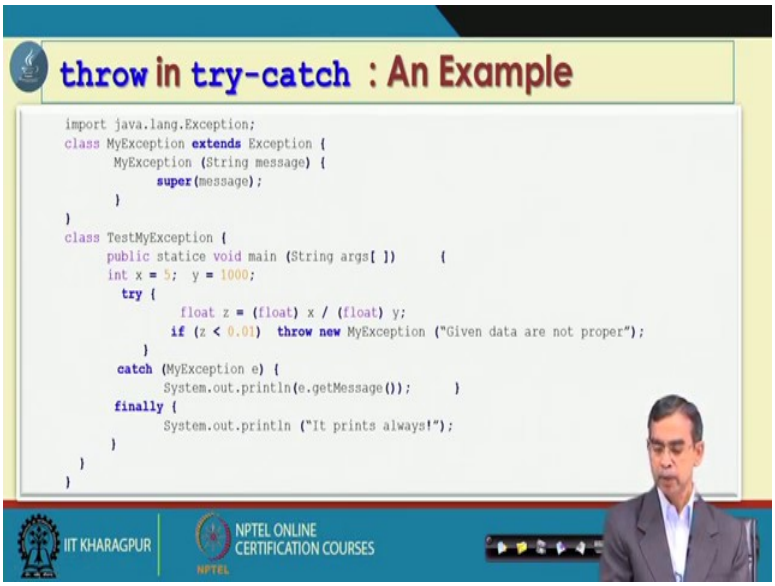
So, there is a throw clause, it is there. Now for example, here as we see, so this is a simple try-catch block in a block and here is a throw exception object. So, a different type of exception and corresponding exception can be thrown by this one. So, this is the throw clause ok. By writing this throw so far, in our previous example, we have not used this throw. Now, we are going to use this throw that explicitly, it will throw and exception objects. Now, this is basically, if you find some error during some input or in your program or system, then you can throw especially that is why the Java developer has maintain this kind of concept here.

(Refer Slide Time: 16:11)



Now, let us have an example here. So, a throw can, can also throw many exception that is also possible and so, there is again, in other than this throw there is also in throws; that means, if it throws multiple exception. So, this is the idea about, let us have an example so that we can discuss, we can learn this example and then have the full idea about it.

(Refer Slide Time: 16:39)



Now, this is the one simple program that we can just check it. Now, here we have created one exception, the name of the exception is my exception and this is basically than inherited, a superclass of exception actually. So, it is a derived class, superclass exception. As you know the exception class is already defined in this java.lang package.

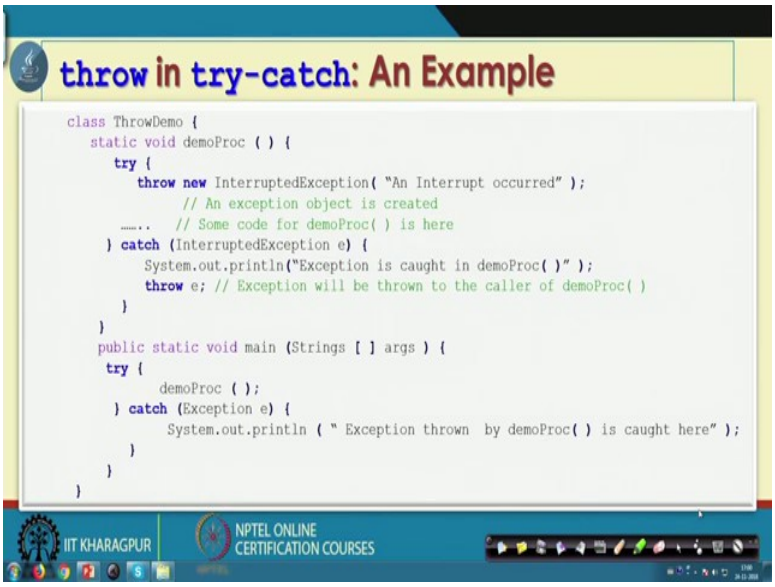
So, this is the concept that ok, we have to create an exception class. So, it is basically, user-created exception class and so what message that this exception class can give to the user is basically, this is the message and this is the constructor that is defined in the exception. So, using this constructor is basically overloading and this basically is a definition of the constructor of this, my exception class, actually.

And so, this part is basically declared an exception class of users own. So, it is the user's own exception. Now, I will discuss about there are many built-in exception classes are there. Now, this is the user-defined exception class, anyway. Now, once you defined the user exception class then you can create an object of that class and here is an example. Now, if we see this example. So, this is a very simple one.

So, this is the main method as we see and hear the possibility of causes of error it is there; obviously, y 0, but in this case, you see y is always 1000 static value. So, no, but sometimes the user can tell that ok, if this ratio is within not a threshold for example, if z is less than 0.01 and then only you can raise the exception. So, here actually if z, which will come from this arithmetic operation, is less than this one, then it will throw, this is the exception with this message.

So, this is the i use of the explicit, throwing an exception in a method and then this is the catch, as you see, this will catch the exception of this type and then the message, the block for the catch. And this is the finally, as you have mentioned, that it will always execute whenever there is an exception occurs or if not occur if there is an exception occur, you can maintain the code, what it should do and whatever it is there. So, this is the idea about, that idea about throw clause.

(Refer Slide Time: 19:19)

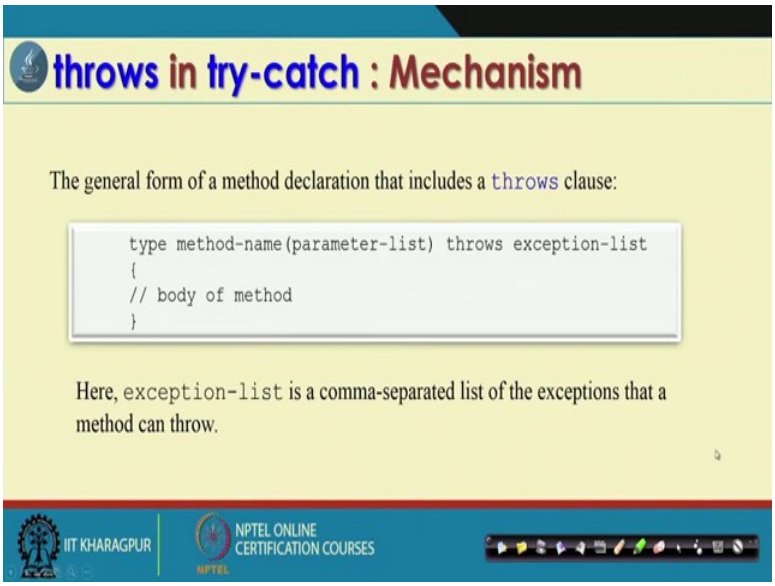


Now, here is another example of a throw clause that is the same thing. Here we have declared one class, the name of the class is throw demo and where there is one method call the demo proc and in this method, we declare our own try block. Here is try, throw a new interrupted exception. This is basically user-defined exception and interrupt occur and whatever the code, there is the demo proc and then this exception if it occurs, it will catch using this catch block and this catch also can throw another expression, is basically

throw and expression to the color program actually, because it is a procedure whenever it is called, it will throw to the caller actually.

So, now here this is the caller program here which will call this demo to proc and it calls it. Now, if any exceptions occur, because of some code here, it will throw and this exception e we will catch. This exception, which is called by this one here also and this will print like this. So, they are, there are many usages of this throw exception. More basic concept here is that, it can throw any exception which user wants to, to throw it in this program so that the program is more robust. So, this is the throw concept there in a block.

(Refer Slide Time: 20:51)

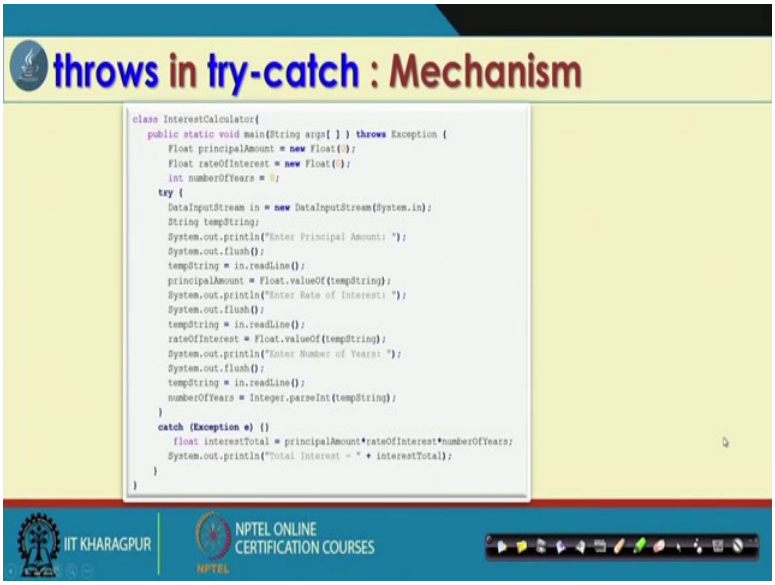


And there is again throws also possible in exception handling for, to handle the exception, to handle the exception in a Java program. Now, for this purpose, the Java program proposes, an idea about it; so, suppose we want to throws an exception, whatever it occurs in a method. Let this is the name of the method, where we want to throws multiple exceptions, then you should use throws clause and then whatever the exception that you have, you anticipate, you can be made it their list.

So, if there is a say array out of bound exception. So, array out of bond exception comma, if there is a null pointer assignment then comma and so. So, this basic list of all exception that this throws can throw and so that the corresponding, the catch block will

be there, they can catch it. So, the idea here, it is that this throws command; we will throw multiple exceptions that may occur in a method.

(Refer Slide Time: 22:06)



Now, so, idea it is now, let us have the one simple example so that we can understand about this one. Yes, now this program, we have once discussed in our demonstration as well as while we are discussing about the input-output. As we see, in this program there are many points, where the exception may occur. Now, let us examine one by one.

So, here is basically if you see, in this part there is no point of exception, but here if you see, here data inputs stream, in this one. Now, here if suppose, it is not possible to, this program is not able to create an object of a data stream in, then an exception we will leads because this in is referred in subsequent here in many a situation. So, if this object is not created then in the subsequent points, it will raises many exceptions are there at, this is because Java is basically an interpreted manner.

So, if this, this fails then; obviously, it will not stop here. It will try to give, execute it and then in every point it will point out the exception, anyway. So, there is a possibility that here an exception may occur in the sense that that in the data input stream object is not successfully created for some reason and also system.in, it is basically standard input sum. Suppose, your keyboard is not working properly while you are running this program or once the input device is not working, which basically may have to system end than in that case also this object will not be created.

Now, so, these are the basic system.out.plus there is no point of which basically clean the buffer the concept it is there. Now, here you see there is again in read line, if suppose this buffer is not able to read successfully, for some reason then here also there will be an exception and here also another exception, if the buffer is not able to read at this point and here is also another exception if the buffer is not able to read at this point ok.

So, we see there are three different points, where the exception occurs and in this particular example only one particular type of exception occurs. Now, you can define that exception of your own, if you defined then you can mention that in this throw block. So, that is exception, but here without taking any, what is called risk, we have discussed the throw exception, any type of exception, if it occurs, in this particular code, then this throws will throw an expression expressively and then this catch, will catch this exception e.

So, the idea about that any exception if it occurs here, this throws as it is mentioned here, will throw the exception in the try block and then the corresponding exception, will this one. Now, here if I do not write this one also and only this part, also it will work. This is a particular case is that, because there is a; obviously, the responsibility of Java runtime manager. Java manager will automatically take care this part, whenever an exception occurs, but sometimes the exception, which is basically, as per the requirement of the program.

Say some (Refer Time: 25:46) in this case it is not working of course, but in some situations. Say suppose, I am telling you one example say in .read line it is working correctly, but I want to see that if suppose I have to enter say two-digit number if user printer one digit. So, in that case, after reading this one I can throw an exception occurs, according to my own requirement like this one. So, in that case, this throws or here also after seeing these things, we can write throw.

So, out of this, there are many ways actually. So, we can write this one and then do all these things. We can have this one, not required, but this one or we do not have this one here, but throw here, throw here, throw here. So, expressively there are many ways the exceptions can be handle actually.

So, whatever be the way basic idea is, that all these things should be catch, either using multiple catch or simple catch, whatever it is there. So, this is the idea the concept for the

throw-in try-catch mechanism. So, we have discussed about many other features regarding the exception handling in Java and there are many more in this line. they are basically, we have discussed only few exception handling concepts. Apart from this, there are many other different exception classes they defined in java.lang package.

So, those things definitely a programmer should know about it then they can use it for a program in a better wider domain actually if you want to develop it. And there is also another possibility Java makes more, what is called the flexible in the sense that it allows a user to define their own exception. Now, in our next module, we will discuss about, how a user can define their own exception and then those exceptions can be handled and whatever the built-in exception classes are there in Java program.

Thank you very much.