FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture29

Lecture 29: 'throws' keyword in Java

Welcome to lecture number 29, Exception Handling. So in Java, so far we have seen try, throw, catch and here you are seeing throws. What do you mean by throws? So suppose the user knows or the programmer knows that there may be an exception in the program, right? For example, let us assume that I am encountering this particular case, right?

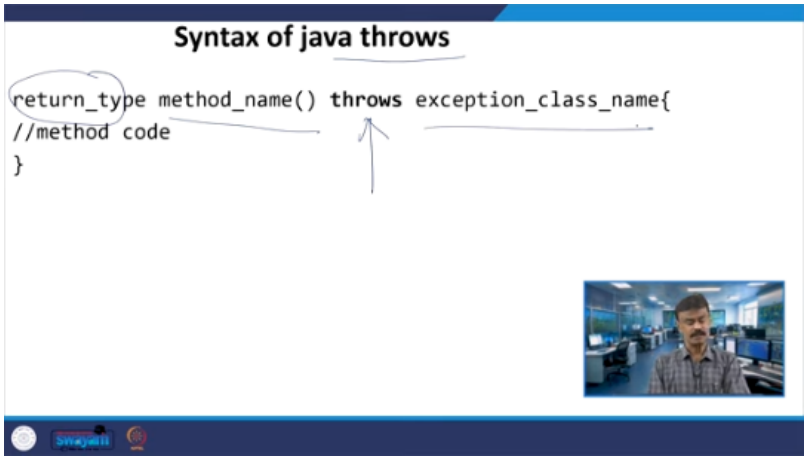
E is equal to A star B slash C star D. So I know that if either C is equal to 0 or D is equal to 0 or both 0. So there will be a divided by 0 error arithmetic exception. So in that case what we can do when you are writing the class there itself you are making that there may be an exception IO exception input output exception. So you use the keyword throws and that particular class we can write that there may be a exception.

So exception handling This is primarily used to manage the checked exceptions, right? If there is an unchecked exception, for example, the null pointer exception, right? So, we do not have any control, right? So, suppose you are trying to create a space and then the pointer, right?

Assume that in the case of a linked list, you are trying to create millions of nodes. At one point in time, the pointer may point to null, right? So, in that case, a null pointer exception may occur. So, these are all unchecked exceptions, and we do not have any control over them. So, basically, when I am using the keyword 'throws', it means

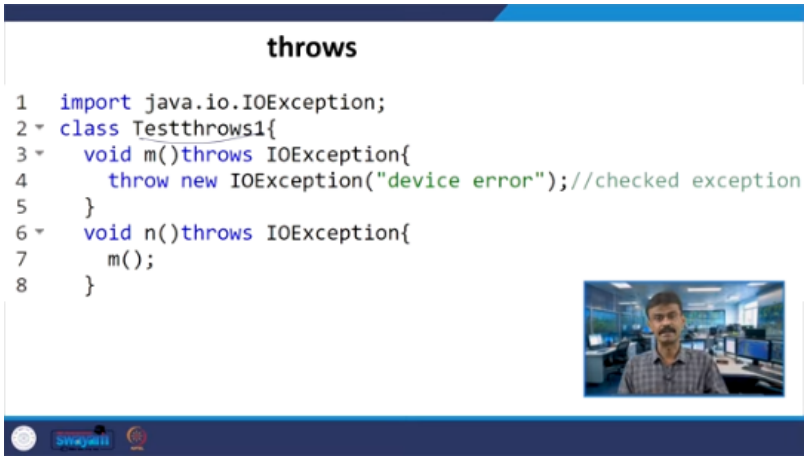
So, this is for checked exceptions. So, this is the syntax for Java 'throws'. So, you have a return type. You have the name of the method. Then you use the keyword 'throws' and then the exception class name, right?

So then you will have the method name, right? So anyway, we will see an example. So this is the syntax, right? So we will see an example. So as I already said, which exception should be declared?



You are going to declare the exception. So to check the exception, right? should be declared because the unchecked exception, right, so this can be controlled, right, unchecked exception can be controlled and also if you are getting error, for example, the stack overflow error or virtual machine error, right, so you cannot do anything, right, and in fact, these are all beyond your control, right, so exception should When it is a checked exception, so I can always declare the exception. So this we are going to see how we are going to define the exception.

So here we are going to see a class test throws one. Test throws one. I am giving the name test throws one. Here I have a member function void m. right which uses throws right which uses throws IO exception says that means I am expecting some exception over here that is a mean in this function I may expect right some exception.



So you are defining throw new IO exception device error. So suppose when you are calling when you are going to call the function or your object is going to call the function and if the compiler is reaching that particular right line so then there will be an error. That is a device error, right? So, this will throw the exception.

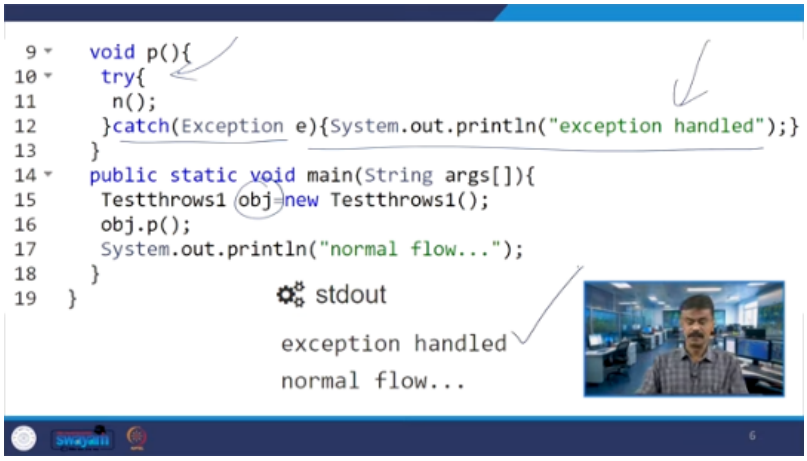
That we will see how we are going to handle this. Now, another function void n throws may be, right? But we have not defined any throw over here. There is no exception here. So, it is calling the function m, right?

It is calling the function m. So, obviously, when suppose, right? In the main, we are calling n, right? So, that n function will call m. N method will call M and M you have IO exception. So let us have here another function called void P. So here you are trying.

So try calling the function N. So when you are calling the function N, N will call M and M will throw the IO exception. So When it is throwing the IO exception, that will be caught over here, right? So, when it is throwing the exception, that will be caught over here, right? So, let us have the main function over here.

So, that means you have only one driver class, which is a test throws one driver class. And we are creating an object OBJ under test throws one class. this obj is invoking p when this obj is invoking p it is going over here right try calling the method n and n is calling the method m right and if you look at over here m again throws io exception so here it is throwing the io exception correct so here it is throwing the io exception so when it is throwing the io exception It will be caught by catch. So this we have seen several times.

All right. So catch exception. So system.out.printl and exception handle will be printed. All right. So then it will come over here line number 17.



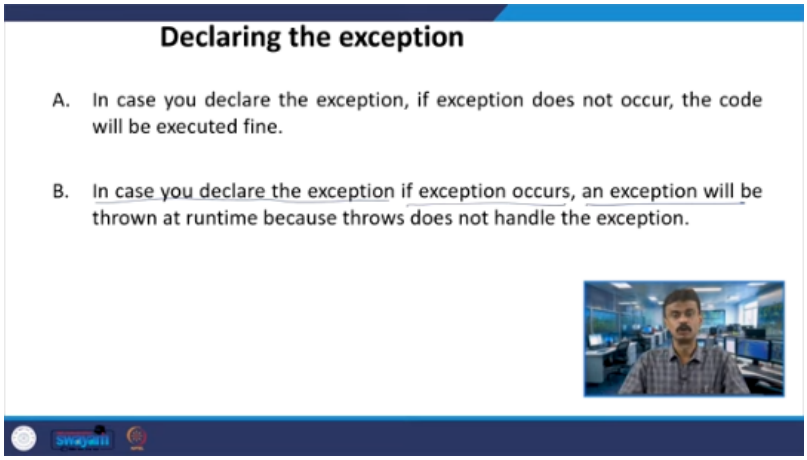
Line number 17 you have one print statement. Normal flow will be printed. So these two will be printed. All right. So here we have used.

First time you have used throws, right? So from this function, I am expecting that this may, right? So when it goes inside this function, I may expect that this will throw some exception. So that is what you are writing. That is what you are defining over here, right?

So this is a checked exception. And since it is being checked exceptions, right? So I can define like this. I can declare like this. correct.

So, I can declare like void n then I am using the throws keyword then input output exception that means in this function I can expect some exception right. So, this is how this is being handled and you know you are getting these two output exception handled and normal flow will be printed. So, when you are declaring the exception So in case you are declaring the exception, if exception does not occur, the code will be executed. So in the namesake you put, but exception may not occur at all.

Nothing wrong. The code will execute. Suppose if exception occurs and the exception will be thrown, suppose you are declaring. In case you are declaring the exception and if the exception is occurring, so the exception will be thrown at runtime because Throws does not handle the exception.



Right. So the exception will be thrown during the runtime. The reason is the keyword throws. Right. So does not handle the exception.

Right. So these are all the two cases when you are declaring the exception. I mean you can see that in fact we will see an example. So program if exception does not occur. But whereas you are using throw an IO exception.

Right. Nothing wrong. It will work. So, let us consider class M. So, you have a void method that throws an IOException.

So, here you are putting System.out.println and device operation performed. So, you have one print statement. And let us have TestThrows3, which is a driver class. You have the main program. Again, you are writing throws IOException.

You are declaring an object m of class M. Object small m of class M. And then you have a constructor. Dynamically allocated memory. Now, M is invoking the method. When M is invoking the method, right, so this will be called, and 'device operation performed' will be printed, right. So, the first print is this, and the second print is this, right.

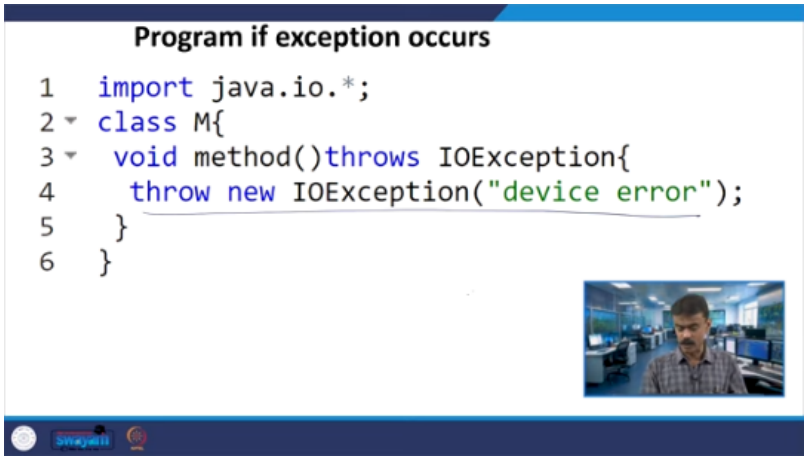
So these two will be printed. So device operation performed and normal flow will be there. So, this is what the first in output we are expecting this device operation perform and the second output is normal flow right. So, these two we are getting as a output I hope it is clear for everyone. So, here in this case we do not have any exception.

So, this program does not have any exception. So, when it does not have any exception it will work like your usual program whereas, here you are using throws

1. And also here you are using second time throws. So, two times you are using throws. You can do that.

Right. Not a problem. That is what the first point over here is. And in the second point, suppose if the exception occurs. Right.

Assume that if the exception occurs. So, here void method throws IO exception. So, this is the exception. IO exception device error. Right.



Throw new exception. input output exception device error all right so in that case assume that the same program test throws 4 is the driver class main program creating a small object m under capital m class this m is invoking method all right m is invoking method when m is invoking the method so here you have this is the exception, correct? So, when it is, alright, giving the exception, the device error will be printed and then you will have, right, device error will be printed. It is printing over here, correct?

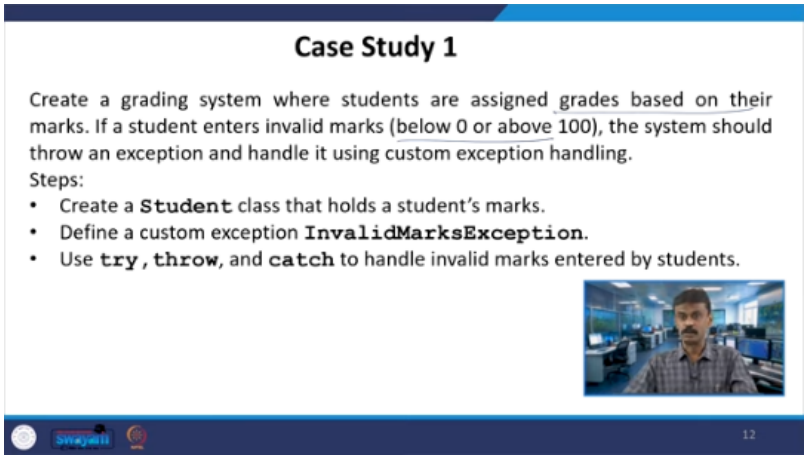
And then that is it, okay? So, you are getting in the case of a standard error, you are getting device error. So, you are getting device error, right? So, this is how it works. If the exception occurs, so this is the problem, right?

And then it is saying, So, m dot method when it is calling correct. So, here you go. You have a IO exception. So, that device error is being printed over here in the std error.

Right. I hope it is clear for everyone. And in fact, so when you are calling this. So, line number 10 when you are calling m dot method. Right.

So, this is creating a problem. You have IO exception. So, the device error will be printed. So, this is how we are taking care the exceptions. So now we will see two case studies, right.

So, the first case study: create a grading system where students are assigned grades based on their marks. If a student enters invalid marks—let us say below 0 or above 100—the system should throw an exception, right? So, this is how we can use the application. So, for many examples, we have seen division by 0, right? Other than division by 0. So, these are all some of the examples.



So, below 0 or above 100. Now, the system should throw an exception and handle it using custom exception handling. So, you are going to—so, usual exception handling. Custom exception handling is nothing but whatever the exception handling that we have studied: try, throw, catch. So, now create a student class that holds the student's marks.

Define a custom exception: invalid marks exception. Define a custom exception: invalid marks exception. So, now use try, throw, and catch to handle invalid marks entered by students. Right. So, let us see how this works.

So, I have an invalid marks exception class. Right. So, I have an invalid marks exception class. This is extending the exception. The inbuilt class.

Correct. So, this we have seen. Several times we have seen. So, now. You have this as a class, and here you have the constructor.

Invalid marks exception. Right. The public constructor. And you have one argument. So, argument.

```java
/* package whatever; // don't place package name! */

import java.util.*;
import java.lang.*;
import java.io.*;

// Custom exception class for invalid marks
class InvalidMarksException extends Exception {
    public InvalidMarksException(String message) {
        super(message);
    }
}
```

One-argument constructor. Parametric constructor. And then you have one superclass called Message. Right. Super.

So that means it will call the exception, and the message will be passed. Right. So here, in this case, the superclass is nothing but the inbuilt class called Exception. Another class is Class Student. The class Student contains name.

Whose data type is string and marks, whose data type is integer. All right. So here you go, the constructor. So the constructor is initializing the name of the student and marks. All right.

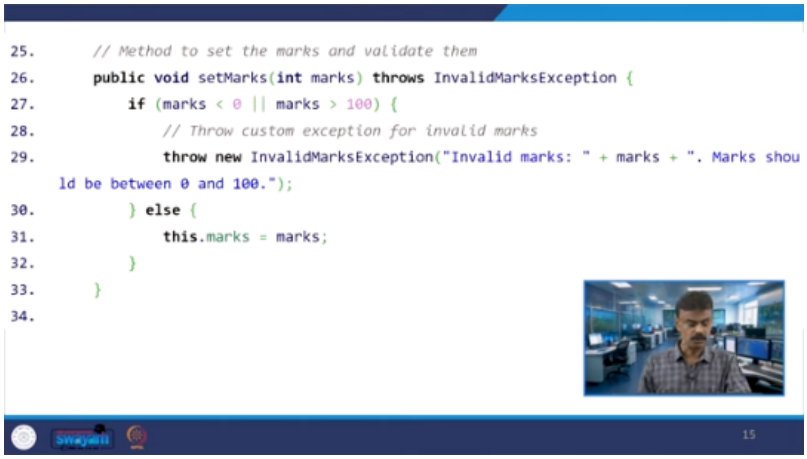
```java
// Student class to handle student details and grading
class Student {
    private String name;
    private int marks;

    // Constructor to initialize student name and marks
    public Student(String name, int marks) throws InvalidMarksException {
        this.name = name;
        setMarks(marks); // Set marks using setter to apply validation
    }
```

So here you have throws invalid marks exception. All right. So you have we have started using throws. All right. We have started using throws.

Here this dot name. All right. So this dot name equal to name. So, whatever name we are passing that will be copied into this dot name. So, that means once you put this dot name.

So, you will not get any null character all right. And then you have the setter method all right setter method set marks marks. So, we will see where you have defined the set marks. So, we have not defined so far. So, set marks it will come marks you are passing whatever you are passing you are passing over here all right.

In marks and you are calling set marks method. Now here you go. Here you have a set marks method. So set marks this method. You have one argument which is throwing invalid marks exception.



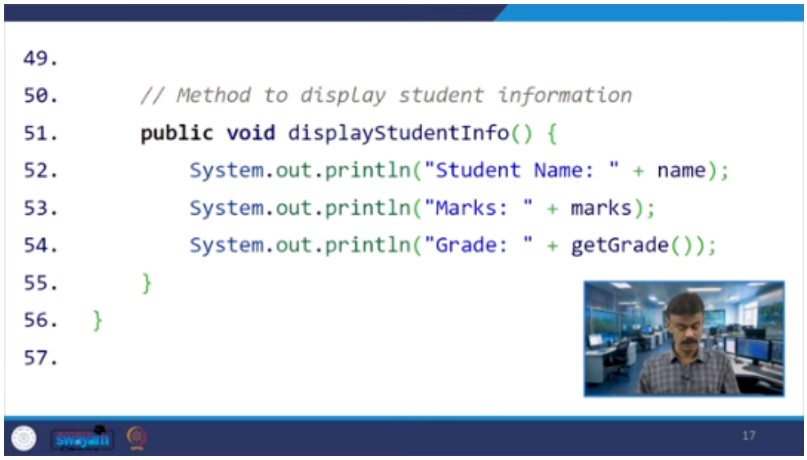
Right. If marks less than 0 or marks greater than 100. Right. If marks less than 0 or marks greater than 100. It is throwing.

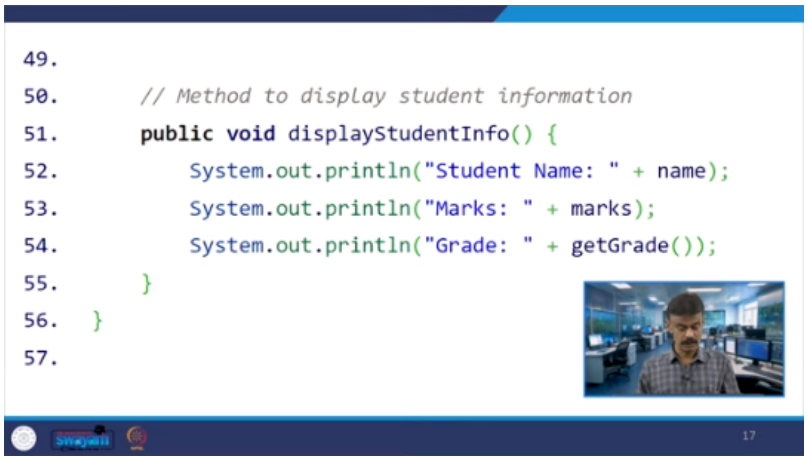
Right. So invalid marks exception. Right. Then it is printing certain statements, invalid marks. Suppose it is as a minus 5 or 105, right?

Whatever be the marks. Marks should be, then it will give information to the user. Marks should be between 0 and 100, right? Otherwise, if the user is entering the marks between 0 and 100, that means when it is satisfying this condition, marks will be assigned to this.marks, okay? So that is what the set marks function over here, set marks method over here.

So now you have one member function call, right? In fact, it is a method called get grade, right? Whose return type is character. So if marks greater than or equal to 90, return A. The grade is A. If marks greater than or equal to 80, return B. If marks greater than or equal to 70, return C. And if marks greater than or equal to 60, D. Else return fail. All right.

So the marks is going below 60. It is failed. So now you have public void display student info. All right. So we are printing student name, which is name marks, marks and grade is a get grade.





All right. So there is a getGrade function. There is a getGrade method. Right. So it's calling a getGrade method.

In fact, we have seen getGrade before. Okay. Fine. So that will be returning one character. Now, this is the main method. All right. So, main driver class: capital M and small main. This is the main method. So, we use try. All right. So, try—you are trying to create student one under student. Then here you have Constructor Student, and then you are passing the name Unschule and marks 85. All right.

So, student Unschule and 85. So, capital S. So you are having Student. Right. So this is the name marks. This dot name is name.

So now it is unschule and set marks marks 85 as it is valid. So this dot marks is equal to marks 85. So, name is Anshul. Marks is 85. So, this is being set.

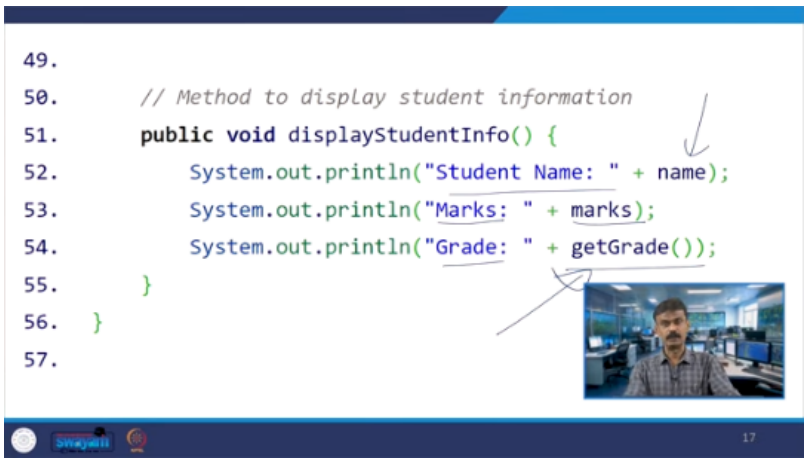
Try. You are using try. So, now student1.display studentInfo. So, this will display the studentInfo. Alright.

Name. Marks. And once marks we know. Right. 85.

Right. 85. So, getGrade. It is calling the getGrade method. So, 85.

If it is greater than or equal to 80, return B. So, greater than or equal to 90. No. False. So, else it is going over here. Marks is greater than or equal to 80.

Yes, true. 85 is greater than or equal to 80. Return B. So, the grade B will be returned. Alright. So, this is what is being printed over here.



Correct. So, it is being printed here. So, next student. Right. Student 2.

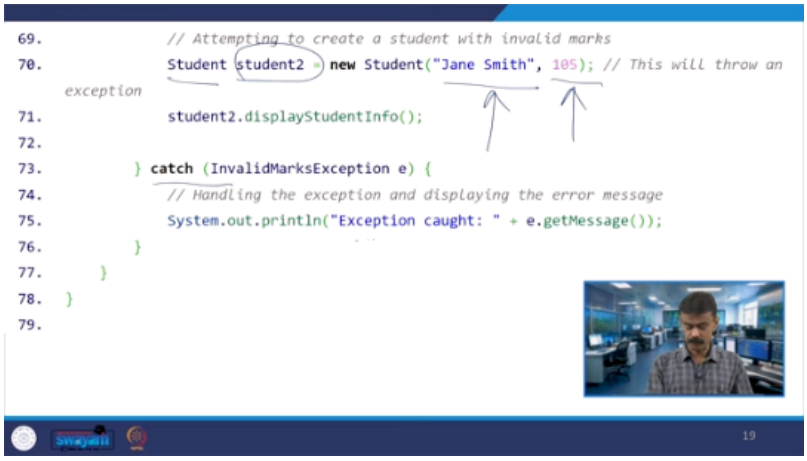
Right. Whose class is the student? Right. Assume the student's name is Jane Smith, and he scored 105. Right.

So, when you pass this, what will happen? All right. So, when 105 is passed, it will throw an exception. Right. It will throw an exception.

What is the exception? Invalid marks exception. Right. So, invalid marks—you have 105. So, marks should be between 0 to 100.

So, this exception will be thrown. So, when this is thrown, correct—you have the cut. Alright. So here, you have the catch. So, exception caught.

Correct. Because of continuation. So, exception caught. And e.get message. So, this e is invoking get message.

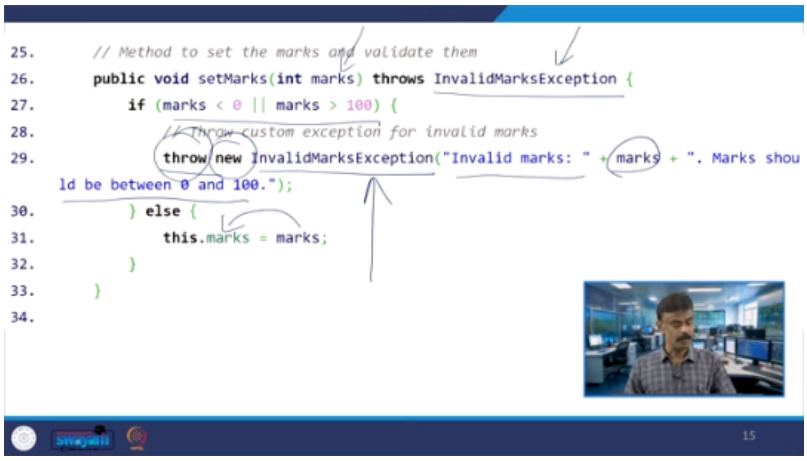


So here you go. The get message may be available in your superclass. Right. So that will be printed. So get message.

It is available in the superclass. So that will be printed. So exception is being caught. So now when you run the code. you will get the output correct.

So, in fact, I talked about the exception caught and the message will be right. So, the message will be this one right. So, invalid marks. So, this will be printed. So, this is what it is returning right.

This is what it is throwing, in fact: invalid marks. So, here in this case, 105. Mark should be between 0 to 100 and will be printed. So, this will be thrown, and you have the superclass exception, correct. So, whatever it is throwing.



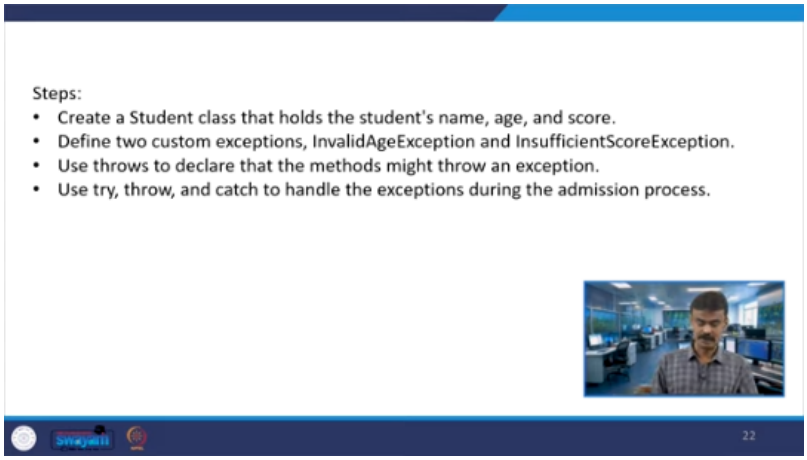
So, e.get message. In fact, we had seen this before, right? e.get message will be printing: invalid marks 105. Marks should be between 0 and 100 and will be printed. And in fact, I also told exception cot will be printed. It is being printed here, and this is nothing but e.get message. The object e is invoking, right? The object e is invoking get message, right. So, get message you have this that will be printed. So, now if you look at this case study, we have used throws, try.

So, here you go: try. So, try, and the exception is being thrown. Also, we have used the catch. So, here you go: catch, throws. So, whatever we studied so far, we have used it.

We will see one more case study. So, let us consider this case study. Let us create a university admission system, right? Let us create a university admission system where students must be admitted based on certain criteria, right? So, a university admission system.

The system will check whether the student meets the age and score requirements for admission. If the student does not meet the criteria, the system will throw an exception, right? Here, we have seen the application. We will use custom exceptions to handle the invalid cases and demonstrate the use of try, throw, catch, and throws, right.

So, all exceptions that we have studied so far, we are going to use them. So, the requirement is the student must be at least 18 years old, right, to apply for admission, right? Must be at least 18 years old. The student must have a score of at least 60 percent, right, or 60. marks to be eligible for admission, right? So, the university admission system follows certain criteria, and if they do not meet these requirements, then, right, it has to throw an exception, and we are using try, throw, catch, and throws. So now, what are the steps we have to follow? You have to create a class that holds the student's name, age, and score.



Right. So, define two custom exceptions. InvalidAgeException and InsufficientScoreException. Right. So, we are going to use two custom exceptions.

Right. So, we are going to define them. These are called custom exceptions. InvalidAgeException and InsufficientScoreException. Use 'throws' to declare that the methods might throw an exception.

Right. So, use try, throw, and catch to handle exceptions. To handle the exceptions during the admission process. So, these are the steps we have to follow. So, if you look at this code.

So, let us have a class called InvalidAgeException. So, which is extending the inbuilt class Exception, which is a superclass, right? Now, you have a constructor InvalidAgeException where you have one string message. So, here you have super(message), right? The superclass message.

So that means, So here you have a superclass. And then the superclass constructor. Will be called. Where you are passing message.

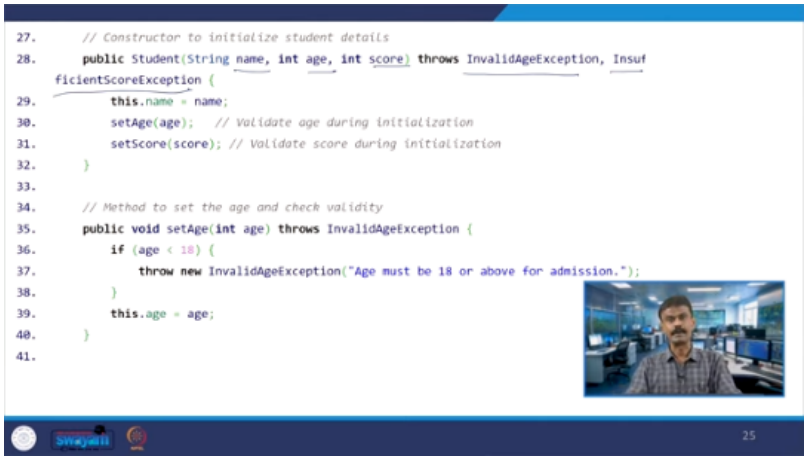
So another class, InsufficientScoreException. That is also extending Exception. So public, InsufficientScoreException.

So that means this is the constructor. A one-argument constructor. So again, it is a super message. It is calling the superclass exception. So now I have a class, Student.

The student has name, age, and score. Name, string data type. Age is an integer, and score is an integer. So now you have a student. Alright.

So, student, which is nothing but the constructor. Right. Line number 28, which is the constructor. Here you have string name, age whose data type is integer, and score whose data type is integer. So throws.

Invalid age exception and insufficient score exception, all right. So, this is the first time we are seeing throws, two exceptions, two custom exceptions, right. Invalid age exception and insufficient score exception. So, in fact, we are having this as a class, right. So, class invalid age exception and class insufficient score exception.



So, all right. So, here you have throws, right. And name is assigned to this dot name, and set age method will be called by passing age. Set score will also be called. So, set age is here, right. So, check the exception throws invalid age exception, correct.

So, it is throwing invalid age exception if age is less than 18, then it will be throwing. So, 17 or 16, they are not eligible for the university admission. So, throw new invalid age exception. So, you are passing this message. Age must be 18 or above for admission.

Otherwise, this age equal to age. So, age will be copied into age properly. This dot age properly. Similarly, set score. So, throws insufficient score exception.

I may get the exception. right score may go less than 60 right if the score is less than 60 through new insufficient score exception and that is printing score must

be or passing score must be 60 or above for admission passing in the sense for the output right printing the message this dot score equal to score so otherwise if it is greater than or equal to 60 score will be properly copied into this dot score fine so with this in mind Alright. So, in fact, one more method is there. Display student info.

Name of the student. Age. Score. Status. Right.

So, it is a printing. Admitted. Because non-admission, it is throwing the exception. So, we do not need to worry. Correct.

So, class university admission system. Public static void. Right. Main. So, you have come to the main and that means this is the driver class.

University admission system is the driver class. So, you are trying. Try. You are trying to create a Object student 1.

Under student. Alright. So, student. Constructor is passing. Three arguments.

Kishore. Age is 19. And marks are 75. Score is 75. Alright.

So then. In the student. Alright. This constructor. Here you go.

This constructor. So name. Kishore. Will be copied. Set age will be.

Called. So set age. Age is 19. Therefore this dot age is age. Right.

And then set score will be calling this 75 will be passed over here. Right. 75. This dot score will be 75. So this is not a problem.

Nothing will be thrown. The program will work as such. So when I display this information, Name is Kishore. Age is 19.

And score is 75. Admitted. Status is admitted. No problem. Suppose.

So here you go. Pradeep. So System.out.println means a new line. I am creating another student. Student 2.

Whose constructor is student. I am passing three arguments. Pradeep, 17, and 85. So what will happen? Age.

Name is not a problem. Pradeep. Age is 17. 17 is less than 18, which is true. So when it is true.

Your invalid age exception, right, the checked exception, and here you have invalid age exception, is going to print: age must be 18 or above for admission, right. That means it is throwing this exception. So when it is throwing the exception, it will be caught. Where? So, invalid age exception.

So, it is being caught. So, you are getting the information 'admission failed,' and what you are getting, e.getMessage, is 'Age must be 18 or above for admission,' right? So, this is what you are getting, and here you go, right? So, another catch function—I do not think we have any score less than 60, right? So, which is not a problem.

So, maybe let us see one more, right? Yeah. So, here Here, the catch will happen. Right. So here, either one.

Suppose both: 17 and, let us say, marks are 55. Correct. So, in that case, the second one will also be called. Correct. So here you go.

One more example. One more student we will create. Student 3. Try. Student 3.

Right. Under the constructor student. So I am passing the name Anshul. Age 20. Score is 55.

Right. So you are trying to display the student info. In fact, in the previous case, you are trying to display the student info. All right. Whereas this will be caught.

Right. This will be caught over here and then admission fail. So then it will display whatever be the information. So here in this case, Unshul. All right.

So here you have Unshul. The age is 20 and score is 55. Right. Age is 20 and the score is 55. 55, right?

Age is 20 and the score is 55. So what will happen? So here you go. So this.name equal to name, unshooled. This dot age is 20.

And when the 55 comes, right, the score is 55, which is less than 60. Right. So this insufficient exception will be called. Correct. So the insufficient score exception will be called, and this will be printed.

So you can see here. So this will be caught here. Catch. Correct. So this will be caught here.

So, insufficient score exception, admission fail, and then e.get message because of the score, all right, the admission, whatever we have written. Score must be 60 or above for admission will be printed, right. So, this is what exactly is happening, all right. So, in these two cases, I mean, it will not go to line number 71 as well as, all right, display student because status is admitted, we have put it here, right. So, that we will not definitely print it.

And similarly, in the previous case also, it will not go over here. Pradeep's case also, it will not go to line number 71. And similarly, here in this case, Anshul's case, we are trying to print in 82, it will not go. Alright. So, this is how we can have try, throw, catch and throws.

Alright. So, with the help of the exceptions. So, we have seen different exceptions and we also now know the standard exceptions, checked exceptions, unchecked exceptions, etc. Right. So, in the next lecture, we will talk about finally keyword.

Right. Finally keyword and maybe one example we will look at for the standard exception in C++. Okay. So, with this I am concluding this lecture. Thank you very much.