Programming in Java

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

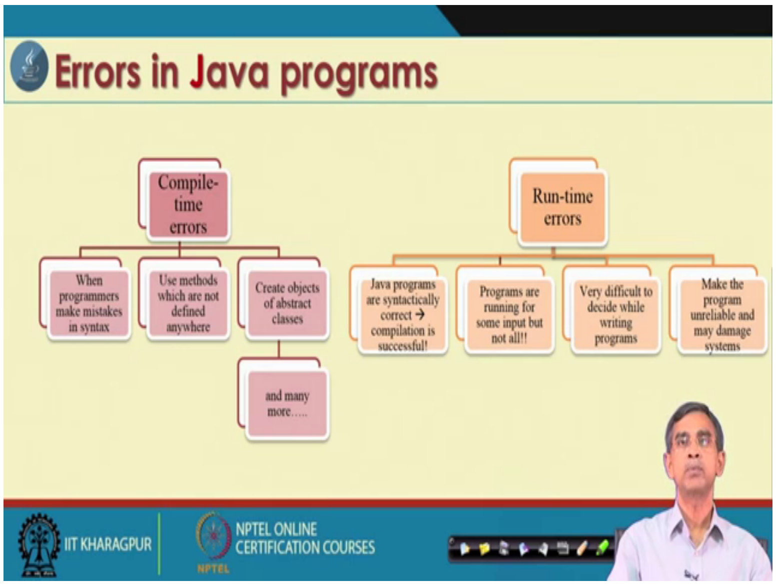
Lecture – 23
Exception Handling - I

So, let us start this lesson with a question. So, suppose what is the big headache for a programmer, whenever the programmer has to develop a very large and complex software. So, many questions may come whether we have to design it properly or the coding should be done by an expert programmer or there should be adequate testing while this one.

In fact, all three things are very much essential, and in fact is a concern for any software development team. Now, java is famous and java is basically at the top, because at the time of coding the programmer can introduce some things into the program itself, so that the program is fall free, error free, and then robust program development is possible.

Now, this is possible in java using a concept, and very useful one concept which is basically unique of its kind, and then java only the java language provides its other programming language C plus plus in some extent it provides, but it is a very unique, and new of its kind in java, the concept is called exception handling. Now, exception handling coming into the way to dealing with errors in the program. Now, before going to have the discussion on exception handling, we should understand about what it does mean about errors in a program.

(Refer Slide Time: 01:53)



Errors in Java programs

Compile-time errors

- When programmers make mistakes in syntax
- Use methods which are not defined anywhere
- Create objects of abstract classes
  - and many more.....

Run-time errors

- Java programs are syntactically correct → compilation is successful!
- Programs are running for some input but not all!!
- Very difficult to decide while writing programs
- Make the program unreliable and may damage systems

Now, so for the concept of error is concerned, there are two types of error in java programs. One is called the compile time errors, and another is called the run-time errors. So, so the compile time errors is very easy, because it is not your headache to debug it.

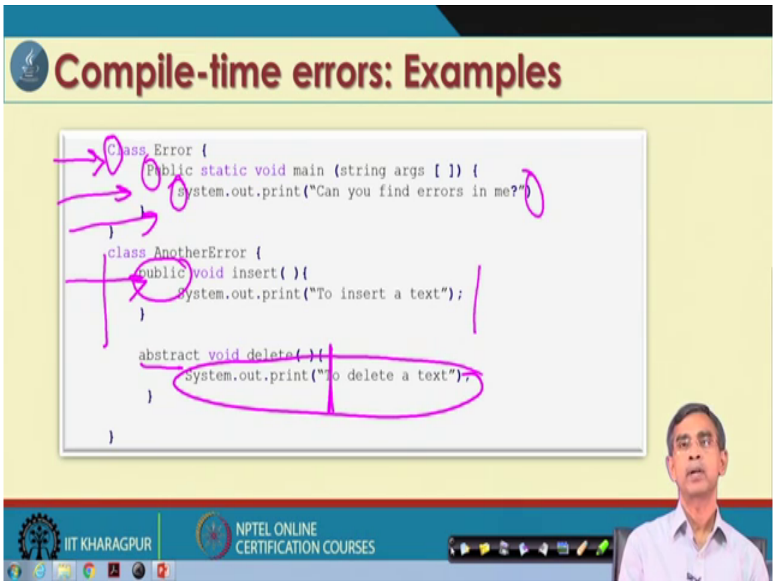
If you write some program which is not seen practically correct that is not written according to the language specification, then it will report an error. And this report, this error will be reported by the compiler java compiler java, C program itself. So, this is the compile time error. And there are many situations in the compile-time error is concerned.

But, I am not so much concerned about compile time error, because if you write a program according to a right syntax of the java programming language, then compilation error is no more there. But, if you have suppose some mistake programming mistake right, then compilation compiler will report an error, and you can fix the error, and then you can develop the program you I mean successfully compile. Once the successfully compiled a program, there will be compile-time error, error free compilation, compiler free program.

On the other hand which is more concern more headache, this is called a run-time error. This is very difficult this is very difficult, because java program is written successful is compilation is successful, but while it is running the program that time, it does not give the correct output or its abrupt termination or there maybe some causes with damage some files stored in a database or like this there are. So, these are the errors is very critical.

And if we do not take care in a very sincere manner, when it leads to a many (Refer Time: 03:38) there. So, many loss in many resources concern is due to this run-time error, so that is why java run-time error needs to be checked, needs to be controlled, needs to be maintained manipulated properly. So, this lesson is basically dealing with this one.

(Refer Slide Time: 03:55)



Compile-time errors: Examples

```
Class Error {
    Public static void main (string args [ ]) {
        system.out.print("Can you find errors in me?"
    }
}
class AnotherError {
    public void insert( ){
        System.out.print("To insert a text");
    }

    abstract void delete( ){
        System.out.print("To delete a text");
    }

}
```

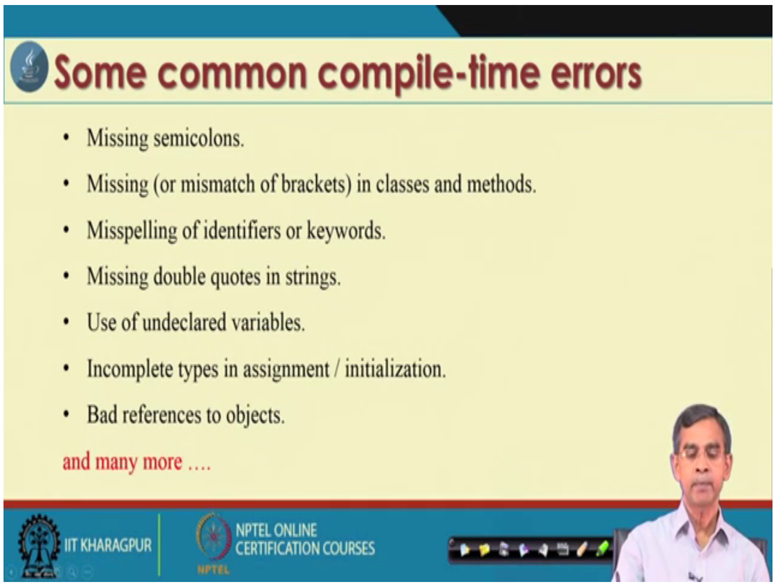
Now, let us have an idea about what is the compile time error, and what is the run-time error. If we see this program, could you check the program, and tell me what are the possible points or locations there, it gives the compile time error ok. If I can help you, you can see I am pointing out. So, there is a location of the error, and here is a error, and here is the error, and here is the error. You can understand why there are the so many errors are there. These are the errors, because according to the java syntax it is not written.

On the other hand if we see other program, may that we cannot find any errors here, this is correct in all sense; maybe that this is an error, because in this class a public is not accessible, whatever it is there. And so this is the idea here and this is also we can find one error, you can tell me why this error, because you have declared a method is abstract.

Whenever we declare an abstract, no code should be included here. So, there are few errors these are called compile-time errors.

So, when you try to compile this program, java will specifically state in this line number this error, in this line number this error, in this line number is this what about the things are there, in report the line numbers. And for each line numbers, whatever the compilation errors are detected will be reported to the programmer, so that programmer can correct it, and then right and then syntax syntactically correct program can be written. So, this is the idea about compile-time errors.

(Refer Slide Time: 05:35)



Some common compile-time errors

- Missing semicolons.
- Missing (or mismatch of brackets) in classes and methods.
- Misspelling of identifiers or keywords.
- Missing double quotes in strings.
- Use of undeclared variables.
- Incomplete types in assignment / initialization.
- Bad references to objects.

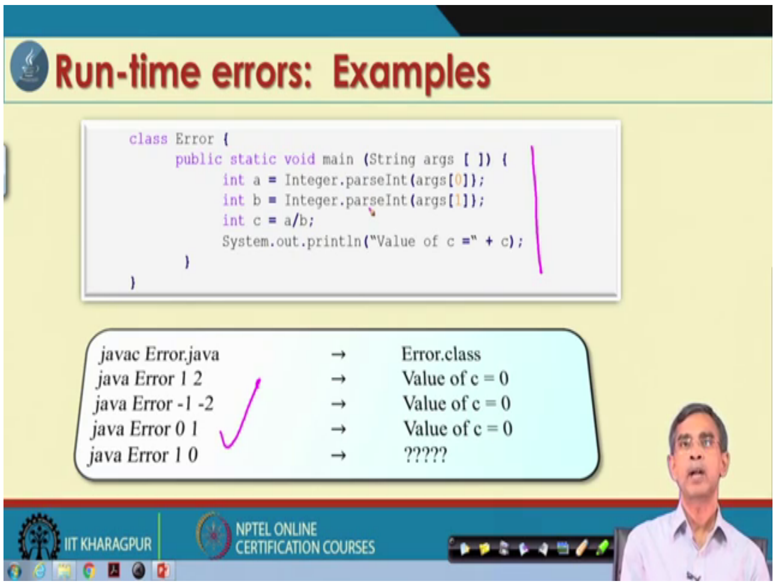
and many more ….

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, let us come to the example of there are many situations, when the compile time errors are there. From the last example, you can understand that. There are so many only few of course; there are many reasons where the compile-time errors are there.

In fact, a java program when starts is programming right programming have it is then usually at the very beginning he faced many errors compiled time errors. And whenever he become experienced, and then from beginners to little bit advance then next part, then he will be quickly able to see that this is a compile time error. Usually, do not face compile-time errors actually in the program.

(Refer Slide Time: 06:17)



Run-time errors: Examples

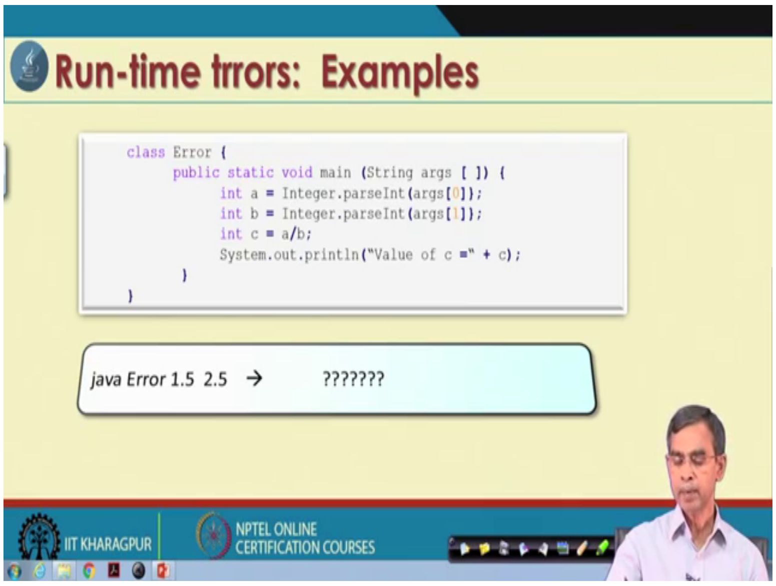
```
class Error {
    public static void main (String args [ ]) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = a/b;
        System.out.println("Value of c =" + c);
    }
}
```

```
javac Error.java     →    Error.class
java Error 1 2       →    Value of c = 0
java Error -1 -2     →    Value of c = 0
java Error 0 1       →    Value of c = 0
java Error 1 0       →    ?????
```

Now, so this is the compile time error, and then what is the run-time error let us have a look. Now, this is the one program we can see, and this basically if you see the program and if you compile, it this will compile successfully, because there is no error in this program; so, for the compilation is concerned.

But, whenever if you want at try to run this program, we will be able to see that all this input will be taken this program successfully, there will be no error. So, there will be run-time error. But, there may be run-time error, whenever we have to give some other input in the same program.

(Refer Slide Time: 06:53)



Run-time trrors: Examples

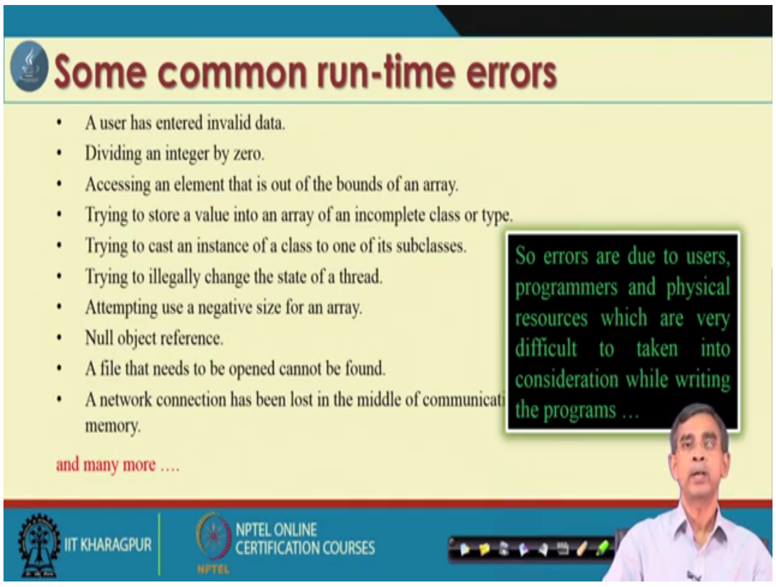
```
class Error {
    public static void main (String args [ ]) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = a/b;
        System.out.println("Value of c =" + c);
    }
}
```

java Error 1.5 2.5 → ???????

For example, if we give this example this input, then you will see these program reports an error. So, this is basically is a unknown format error because of this statements. Because, this statement only take the integer; but you have passed as a floating point number and that will not allow. So, this is basically run-time error.

So, what we can say is that so far the run-time error is concerned; the program can run for some input. But, if it cannot run for all possible inputs, then it leads to run-time error. So, when you write the program in this case for example, you have to define the program in such a way that if it works for the integer type data only, then, you have to check it. If it if a user gives other type of input, then accordingly it will it will check it and then report it, so that that kind of input can be bypassed, and then program cannot terminated abnormally. So, this is idea about run-time error is concerned again like compile time error.

(Refer Slide Time: 08:05)



Some common run-time errors

- A user has entered invalid data.
- Dividing an integer by zero.
- Accessing an element that is out of the bounds of an array.
- Trying to store a value into an array of an incomplete class or type.
- Trying to cast an instance of a class to one of its subclasses.
- Trying to illegally change the state of a thread.
- Attempting use a negative size for an array.
- Null object reference.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communicati memory.

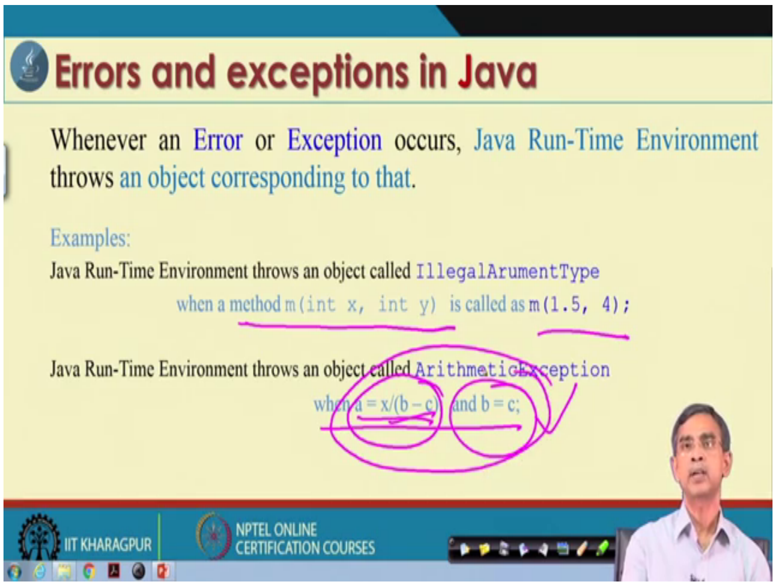
and many more ….

So errors are due to users, programmers and physical resources which are very difficult to taken into consideration while writing the programs …

There are many sources many reasons that line run-time error will occur here. I have mentioned only few some important things which usually occurs there, and there are many more in fact. We will discussed in our while discussing about that robust java program writing, then some errors will be discussed here. I have listed few errors, which are usually common type of errors, but there may be many many more errors are possible.

Now, so these are the basically run-time errors. Now, this is very difficult, because while a program is writing that time program has to take care of all possible checks that this one. But, sometimes it is very tedious, and obviously the programmers say ability that programmer cannot do all kind of checking’s, so in that case what is the way out.

(Refer Slide Time: 08:57)



Errors and exceptions in Java

Whenever an Error or Exception occurs, Java Run-Time Environment throws an object corresponding to that.

Examples:

Java Run-Time Environment throws an object called `IllegalArumentType` when a method `m(int x, int y)` is called as `m(1.5, 4);`

Java Run-Time Environment throws an object called `ArithmeticException` when $a = x/(b - c)$ and $b = c$;

So, the as a way out java provides a mechanism call handling errors and exception in Java program. So, it is basically the who will do it, definitely compiler not do, compiler only check the syntactical error, which reports as a compile time error. But, this is the error and exception which basically occurs; I should not say exception at this stage, because I have not discussed any exception here. I can tell only error. This is checked by java run-time interpreter environment is our java run-time system.

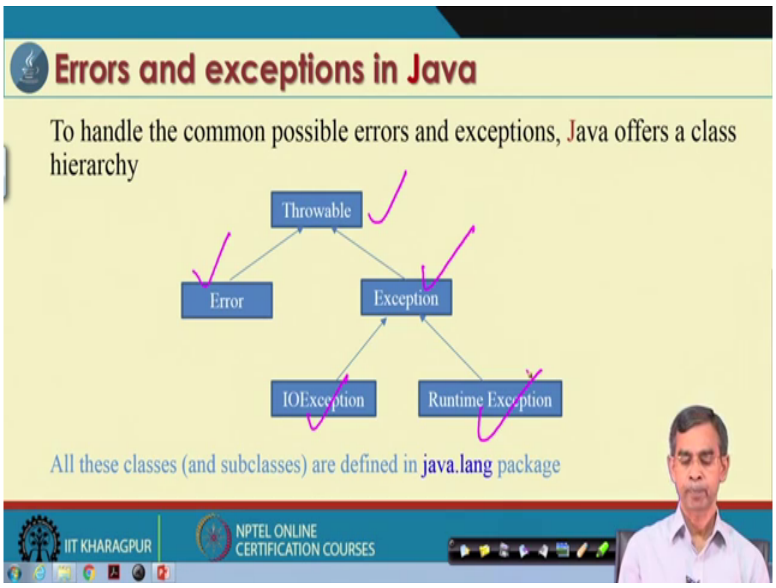
Now, java run-time system whenever find an error is basically in order in leave of abnormally terminating the program, then it basically throw an exception, and the java programmer should have the facility to catch the next session, so that (Refer Time: 09:55) bypass this error. And then for the rest of the part, it can be executed successfully, if they are correct. So, this is a concepts.

So, whenever there is an error or some abnormal things is occurred, java run-time environment will catch it, and then throw it, and then java programmer will process it, so that abnormal termination can be avoided. Now, here is an example suppose here is a method suppose you have declared as integer X and Y, and in some program by mistake user called this as a this one 1.5, 4 and right. So, compilation error will not be reported here. But, at the time of execution whenever input is 1.5, it will throw an exception, so that this method cannot be invoked success invoked, and then by past.

As an another example here you can see, whenever we have to perform an arithmetic operation like this one, and if suppose b and c these are the two variables values is equal, then it basically gives rise to a an error, it is called the divide by zero error. And if you show in C programming, if there is a divide by zero error, then the programming (Refer Time: 11:10) without giving any in output for the rest of the programs or whatever it is there.

But, in case of java if we can catch this one or in this kind of error occur for this kind of situation, then java run-time environment will throw an exception. And then with the catching this exception, we can take care about this kind of errors in the program. So, this is the concept that error and exceptions in java can be handled during the run-time and then, avoid the errors.

(Refer Slide Time: 11:41)



Errors and exceptions in Java

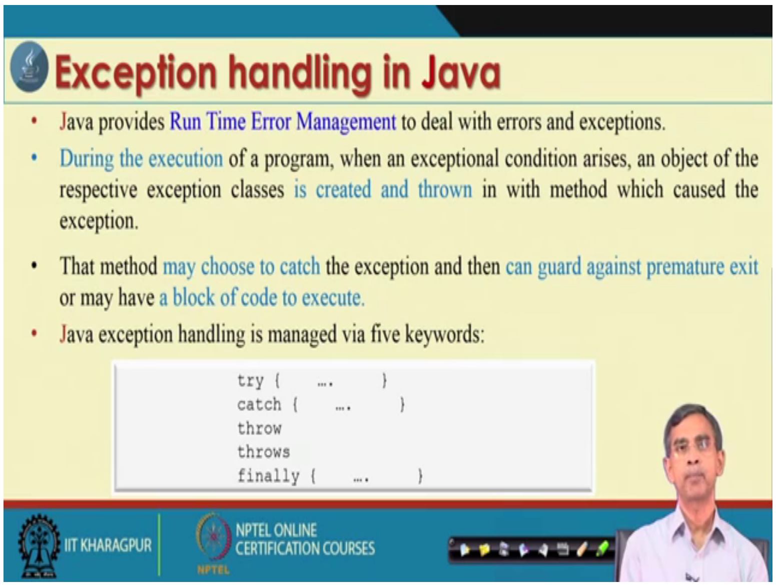
To handle the common possible errors and exceptions, Java offers a class hierarchy

All these classes (and subclasses) are defined in java.lang package

Now, so for the errors and exceptions is concerned in java program, there is a big what is called the development. And there in java, there is a package called the java dot lang package, which basically discussed many classes. And here is a class hierarchy as we see there is a class called Throwable.

And this throw able can throw two type of classes called error and exceptions. And for the exception there is IO exception and then run-time exception. So, this is a total class hierarchy or the mechanism dealing with error and exception in java program. So, this is a concept called n exception handling in java. So, there is a class as I told you java dot lang package, which basically define all the classes it is there.

(Refer Slide Time: 12:43)



Exception handling in Java

- Java provides Run Time Error Management to deal with errors and exceptions.
- During the execution of a program, when an exceptional condition arises, an object of the respective exception classes is created and thrown in with method which caused the exception.
- That method may choose to catch the exception and then can guard against premature exit or may have a block of code to execute.
- Java exception handling is managed via five keywords:

```
try {     ….     }
catch {     ….     }
throw
throws
finally {     ….     }
```

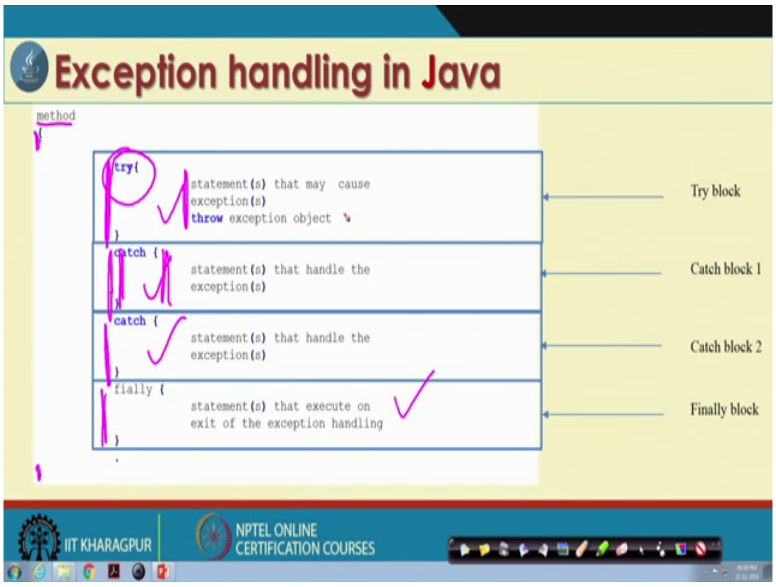
Now, let us see about little bit details about the different possibilities about the class and then exception handling mechanisms that is there it is very big story. So, we will cover this kind of discussion not in one module, we may take two more modules to complete the discussion with illustration. And then finally, we have a demonstrations; so that we can pin point exactly how the things the mechanism works for us.

Now, first let us have the idea about in general what is the exception handling mechanism it is. As I already mentioned about that java provides run-time error management on concept. In fact, java this is the error management, there is memory management, and there is synchronization management, there are many other management schemes are implemented there.

So, java run-time error management is a policy is a scheme which basically take care, if there is any error occurs in the program. If there any occurs, if there is any error that that may occur in a program, it can catch it, and then this can provide say solution to get the premature exit of the program, therefore loss of other data or damaging of any system.

So, regarding these things in java or I can say the java exception mechanism handling mechanism is consist of five specific keywords. So, five keywords are called try, catch, throw, throws, and finally. So, these are the keyword likes a class keyword, interface keyword class, then extent keyword like this one these are the also keyword, we have to embedded into the program.

(Refer Slide Time: 14:27)



Exception handling in Java

```
method
{
    try{
            statement(s) that may  cause
            exception(s)
            throw exception object
    }
    catch {
            statement(s) that handle the
            exception(s)
    }
    catch {
            statement(s) that handle the
            exception(s)
    }
    fially {
            statement(s) that execute on
            exit of the exception handling
    }
    .
}
```

Try block

Catch block 1

Catch block 2

Finally block

Now, let us see how we can do all this things are there. In general, these are the keywords that we have already mentioned here can be embedded in a program. So, for example, so suppose you are declining method, these are beginning of the method, end of the method.

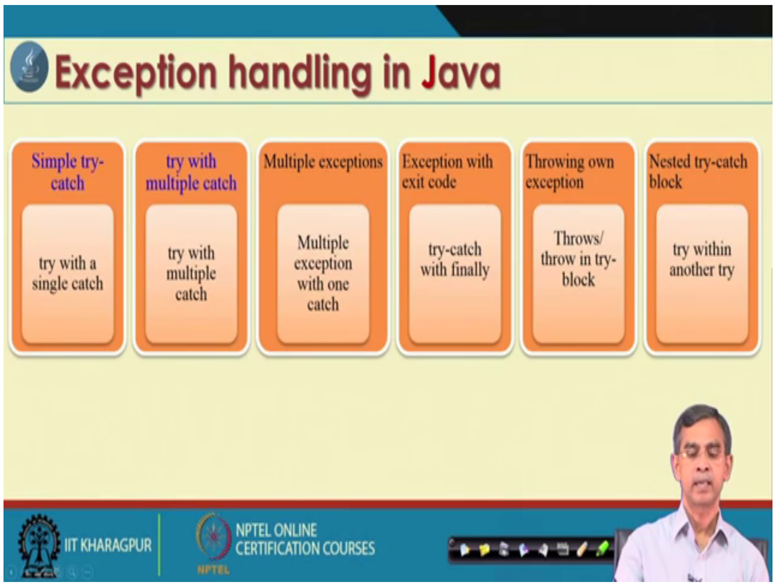
And if we see this is the possible scope of error, so that obviously as a programmer should know that there may be chance. For example, here and user should give an input an integer, but sometimes user by mistake or whatever it is can give other type of integer. So, we can have a check, so like this on. So, we can put all check.

Now, here is basically try we will take care. So, try can check that these are the possible sources of error in a program. So, whatever the course those are basically under scanner, they can be put into this block is called that try block. And then for many errors there are many different type of exceptions occurs. So, for each errors and corresponding the exception, we can plan the catch, so if it is a divide by zero error, if it is a null pointer error, if it is out of mount error, whatever it is there for each error, we can mention the catch that means, if an error occurs, this block will take care what to do. So, for each error they are maybe one or more block, we can plan for it.

And these are the finally right, so it basically whatever it is error occurs or not, it will execute this one so finally. So, this is the basic what is called a scheme. So, for the handling exception is concerned in a java program. So, usually call try-catch mechanism try catch.

Now, here also in addition to try-catch two more things is a throw, in other sense is called the throws also there. So, throw can be put anywhere there that mean explicitly if you have to throw an exception, as I told you implicitly means java run-time management whenever fine and error, it will throw and exception. But, sometimes you can the programmer can explicitly throw and exception, so these can be achieved by means of throw keyword. So, this is the basically in brief the concept of exception handling mechanism in java.

(Refer Slide Time: 16:57)



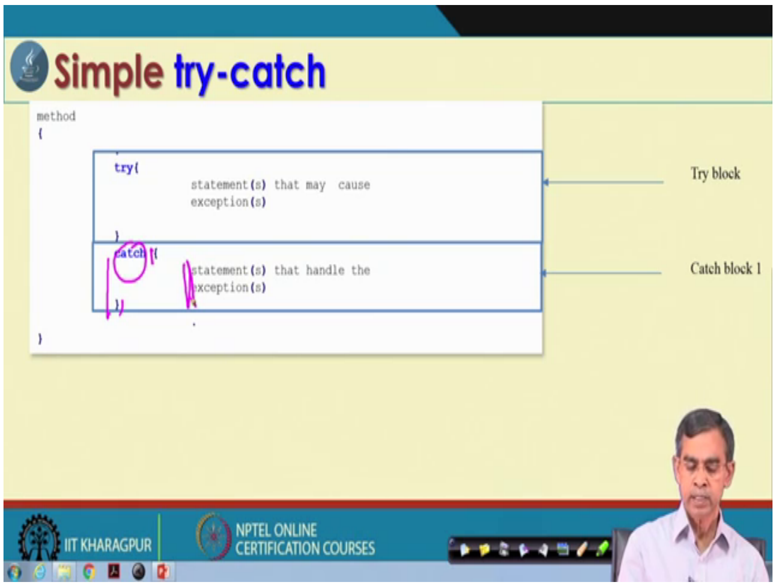
Exception handling in Java

Simple try-catch	try with multiple catch	Multiple exceptions	Exception with exit code	Throwing own exception	Nested try-catch block
try with a single catch	try with multiple catch	Multiple exception with one catch	try-catch with finally	Throws/ throw in try-block	try within another try

But, now it is very important to discuss about each the block one by one each concept one by one, now we can structure this concept in here is a six different way that is the exception handling can be done by using simple try-catch that means, only one try and one catch. There may be also one try with multiple catch. So, so this is the simple try, then try with multiple catch.

And then they are may be multiple exception, which can be caught by only one catch that is also quite possible. And they are also possibility that whenever the exception is occurs, it should finally execute some code. So, there is a concept called finally. And throwing the explicit exception so throws or throw in a block. And another is a nested try-catch; there mean within the try-catch we can defined another try-catch, within the try-catch another. So, nesting; so try-catch within try-catch. So, these are the different six different way of handling exceptions in java will discuss all these cases one by one in our nah subsequent slides.

(Refer Slide Time: 18:21)



Simple try-catch

```
method
{
        try{
                statement(s) that may  cause
                exception(s)
        }
        catch {{
                statement(s) that handle the
                exception(s)
        }}
}
```

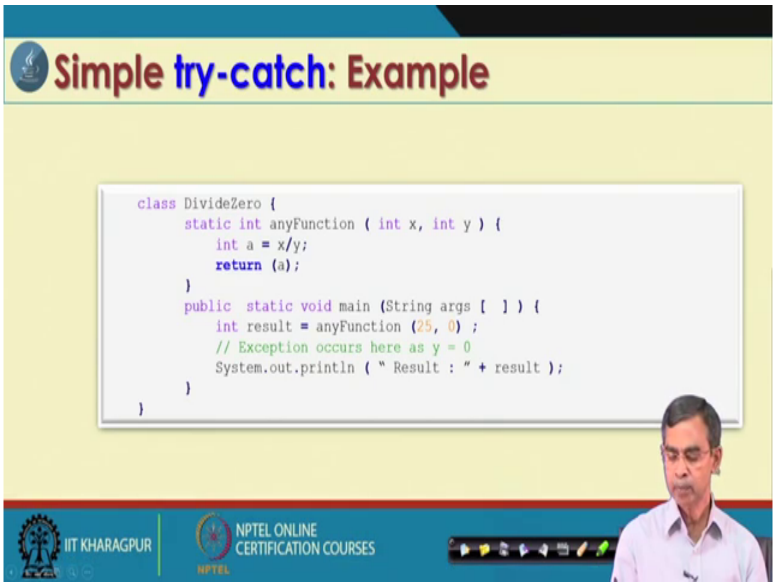
Let us first discuss about simple try-catch block. Now, typically a simple try-catch block looks likely which is shown here. So, this is the method that we are going to discuss here. As you can note that usually LR occurs in a method. So, we are discussing in the context of method only, so this is a method declaration.

Now, in this method the statement which is basically the critical statement. So, this critical statement can be put under the try-catch that is ok. So, eventually in a very default sense for every statement, we can write one try catch. Then obviously; but it is very TDS and programs very bulky, looks very clumsy and it is not desirable. Only if you see that this is a possible errors causes of error, then we can put that block that coat under the try-catch.

Now, so this is the try block, we will contains this is the code (Refer Time: 19:22) to cause and error. And then what that cause at the end of this try, we can put a catch. Now, here the syntax you should see, try is the keyword. And beginning and ending that basically called the try block. So, this basically whenever an exception occurs, this will be basically check it, so that is if you do not put this one, if any exception occur, java run-time manager will not be able to throw it I mean handle it, so that is why we should put it explicitly there.

And then this is the catch like this try, it is a keyword catch. And it has also beginning and ending that means, if an exception occurs, what are the next steps the program should do? So, this basically includes the code, if an exception occurs. So, this is the idea about the concept of simple try-catch block. Now, definitely it is if you have an example, so the concept will be understood better.

(Refer Slide Time: 20:25)



Simple try-catch: Example

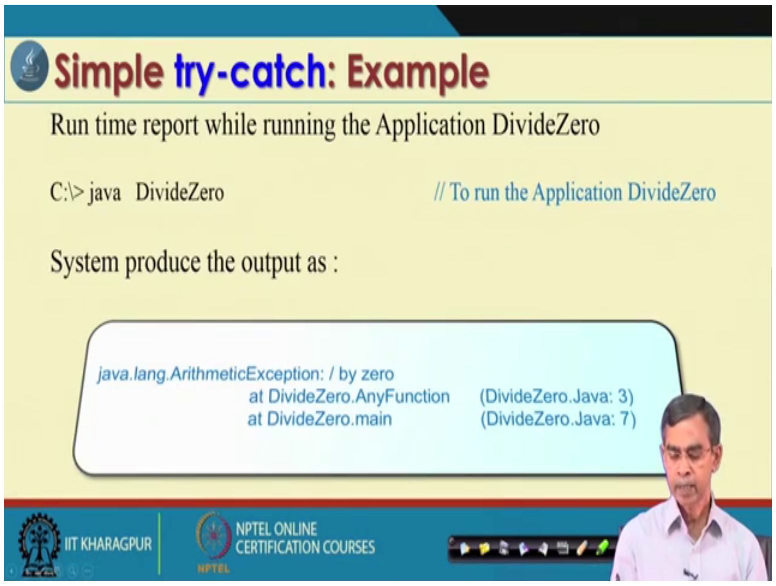
```
class DivideZero {
      static int anyFunction ( int x, int y ) {
          int a = x/y;
          return (a);
      }
      public  static void main (String args [  ] ) {
          int result = anyFunction (25, 0) ;
          // Exception occurs here as y = 0
          System.out.println ( " Result : " + result );
      }
}
```

Now, let us have a simple example to discuss it on. Now, here let us have a look about the simple class it is there, we have define one class the name of the class is called divide zero. And here one function is declared any function, this has two arguments integer x and integer y. And here is the code.

Now, and this is the main method it is declared there now ok. Now, you can see where is the source of error, if you look at this program? Now, here you can see there is a source of error why, because this method any function can be called for any integer values that is ok, it will work for any integer values.

However, it will not work properly, if the second integer is zero. Because, if second integer is zero, they are is basically divide by 0. And no processing unit can deal with the divide by zero error in a program. So, whenever y equals to zero, then if we put all this things under a try-catch. Then it will basically if that situation occurs that y zero, it will throw an exception, and then following catch will catch it, and then do it whatever it is there.

(Refer Slide Time: 21:47)



Simple try-catch: Example

Run time report while running the Application DivideZero

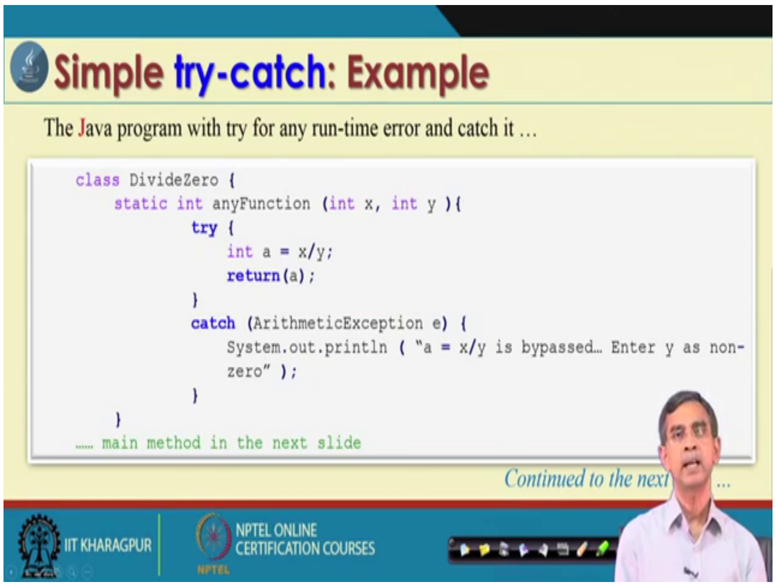
```
C:\> java  DivideZero                    // To run the Application DivideZero
```

System produce the output as :

```
java.lang.ArithmeticException: / by zero
                 at DivideZero.AnyFunction     (DivideZero.Java: 3)
                 at DivideZero.main            (DivideZero.Java: 7)
```

Now, see this program is therefore not a robust program. So, what is the weather this program can be make robust. And here is the one possible input that it will work easily always, but sometimes it may not work.

(Refer Slide Time: 21:53)



Simple try-catch: Example

The Java program with try for any run-time error and catch it …

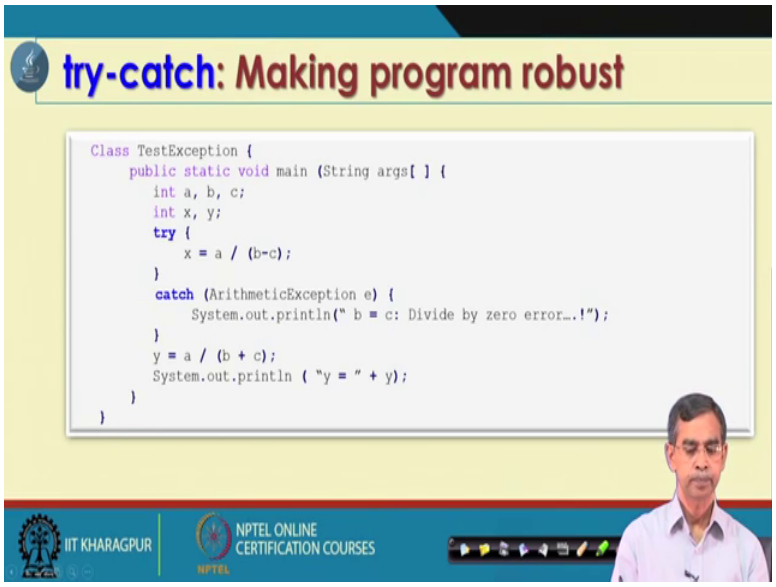
```
class DivideZero {
    static int anyFunction (int x, int y ){
            try {
                int a = x/y;
                return(a);
            }
            catch (ArithmeticException e) {
                System.out.println ( "a = x/y is bypassed… Enter y as non-
                zero" );
            }
    }
…… main method in the next slide
```

Continued to the next …

And here is basically is an attempt to make the program robust, how you have done it if you see, this is the code which is basically very critical code, so we put this code in the try-catch block. So, this is basically under try. And if you see, this is the catch is basically what is basically tell that x y is bypassed, enter y as non-zero. So, you should passed the input y as non-zero like. So, this basically may send a message or an output to the use at if this kind of error occurs.

So, here if you see, this code will not execute always. Only if error occurs here, then only it catch and then accordingly it will execute. So, this is the way that we can build the program, so that it can in a robust way that means, it will take any input including zero the second argument as an input also. So, this is the idea about making the program robust is a very simple example here.

(Refer Slide Time: 22:59)

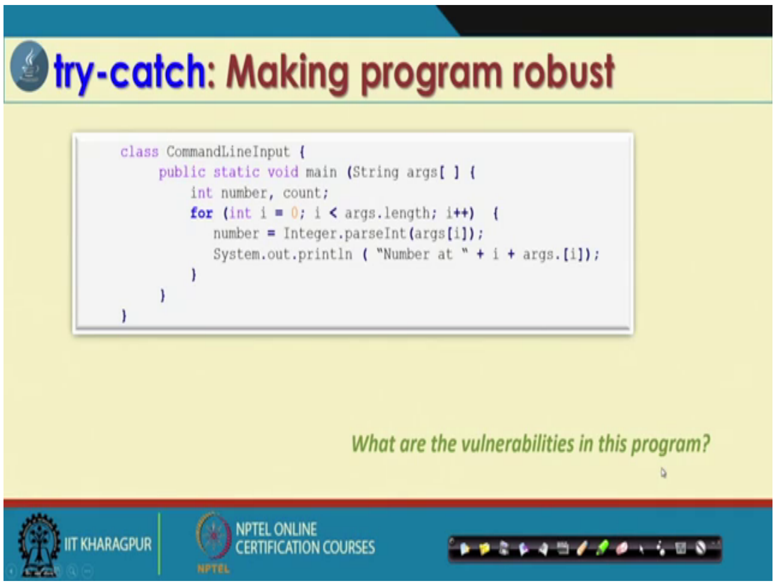


try-catch: Making program robust

```
Class TestException {
     public static void main (String args[ ] {
        int a, b, c;
        int x, y;
        try {
            x = a / (b-c);
        }
        catch (ArithmeticException e) {
             System.out.println(" b = c: Divide by zero error….!");
        }
        y = a / (b + c);
        System.out.println ( "y = " + y);
     }
 }
```

And here is another example as you can see here what you have done here is that we put these things in a try-catch block, you know why you have done it. It definitely because this code will again leads to divide by zero or occur for some input if b equals to c. So, here after if in case suppose this error occurred for b equals to c is basically report that this occur there, and then it basically bypassed and this. So, this is the idea about the try-catch or I can say that simple try catch mechanism in java to handle the errors.

(Refer Slide Time: 23:45)



try-catch: Making program robust

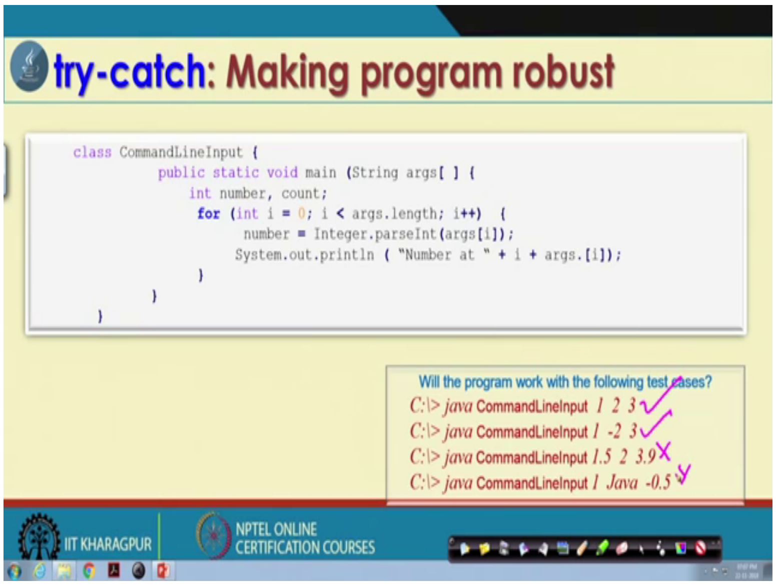
```
class CommandLineInput {
     public static void main (String args[ ] {
          int number, count;
          for (int i = 0; i < args.length; i++)  {
             number = Integer.parseInt(args[i]);
             System.out.println ( "Number at " + i + args.[i]);
          }
     }
}
```

What are the vulnerabilities in this program?

Now, here again let us have the look, this program looks very innocent apparently if you see that there is no errors. Now, can you tell me what are the possible errors are there in this code is basically common line input that means, here if you see args dot length ok. Args dot length is basically if it basically say that number of inputs, when you have passed as a common line ok.

Now, so here is a possibility that user can run this program without passing any input that means, this value will be zero. Now, if this value is zero or others or it passes suppose two value, but argument of this array is much more, then there may be what is called the is called the outer bound error there.

(Refer Slide Time: 24:51)



try-catch: Making program robust

```
class CommandLineInput {
            public static void main (String args[ ] {
                int number, count;
                 for (int i = 0; i < args.length; i++)  {
                       number = Integer.parseInt(args[i]);
                     System.out.println ( "Number at " + i + args.[i]);
                 }
            }
    }
```

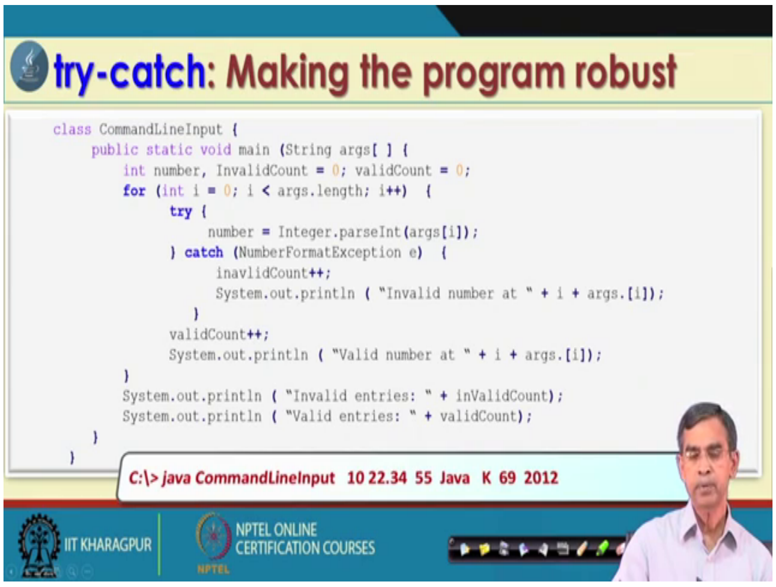
Will the program work with the following test cases?

```
C:\> java CommandLineInput 1 2 3 ✓
C:\> java CommandLineInput 1 -2 3 ✓
C:\> java CommandLineInput 1.5 2 3.9 X
C:\> java CommandLineInput 1 Java -0.5 ✓
```

So, this program if you run it for many inputs, you may check that this program may run for some input, but not truly for all input is there. Here, is a some instances I have mentioned here that, this program will work for this as an input, this as an input, but not this, but not this, now you can understand why this program will fail to run for this kind of input, but not this one.

Anyways, so as a exception handling mechanism it is our duty responsibility, so that all the inputs if user gives, then this program still will work that means, for the correct input it will definitely work correctly. But, for the wrong input passed to the program, it should take care them and then report to the user accordingly. So, this is the concept of the try-catch block in your java program. The simple try-catch block in the java program and here is the ok.

(Refer Slide Time: 25:43)



try-catch: Making the program robust

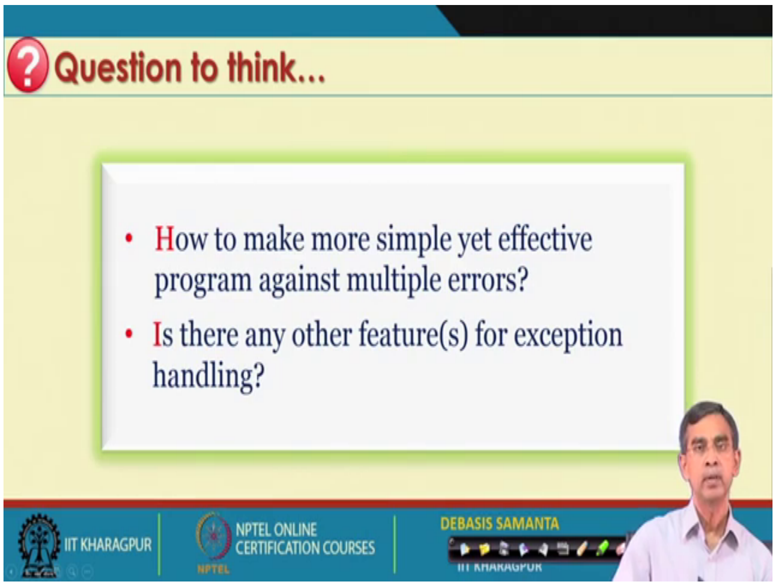
```
class CommandLineInput {
    public static void main (String args[ ] {
        int number, InvalidCount = 0; validCount = 0;
        for (int i = 0; i < args.length; i++)  {
            try {
                number = Integer.parseInt(args[i]);
            } catch (NumberFormatException e)  {
                inavlidCount++;
                System.out.println ( "Invalid number at " + i + args.[i]);
              }
            validCount++;
            System.out.println ( "Valid number at " + i + args.[i]);
        }
        System.out.println ( "Invalid entries: " + inValidCount);
        System.out.println ( "Valid entries: " + validCount);
    }
  }
```

```
C:\> java CommandLineInput  10 22.34  55  Java  K  69  2012
```

So, earlier program was not robust, because we see we check that this program does not work for all. And this is an atom, how we can make the program as a robust using try-catch block here. So, it is basically try catch block to take the full (Refer Time: 26:04) of the errors that may occur.

And whatever the input we have mentioned in the last execution, if we run this program with all the input, we will see this program will run. It will not give any abnormal termination of the program, and then we can say the program is robust. So, this is the idea about handling exception in a java program. And as an example you can see this program if you run, it will work for you correctly successfully.

(Refer Slide Time: 26:31)



Question to think...

- How to make more simple yet effective program against multiple errors?
- Is there any other feature(s) for exception handling?

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
IIT KHARAGPUR

So, this is a basic introduction about the exception handling. There are many more features many more things to be learned in so for the exception handling is concerned. So, basically the idea is that we have learn about try-catch, but there are few more features basically there, which makes more effective way of writing the program. And there are many more features, so for exception handling is concerned. So, all this things will be discussed in our next module, and thank you for your attention.

Thank you very much.