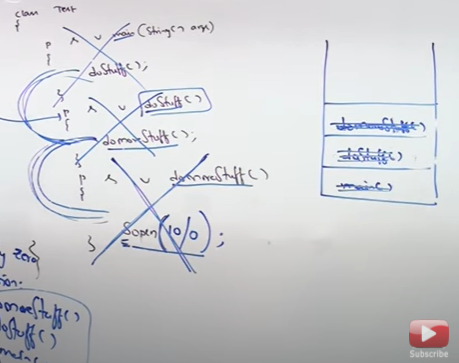
Exceptions by durga soft

Lecture2: Runtime stack

For every thread JVM creates one runtime stack. And this is going to be store in stack known as stack frame or Activation record.

What is default Exception Handling in Java:

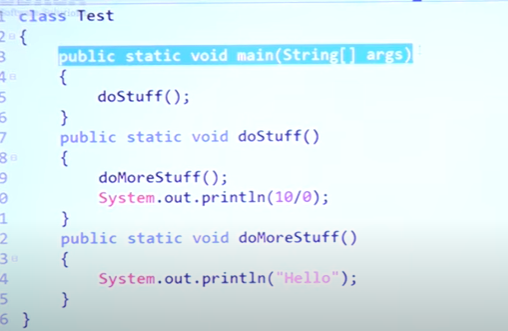


Here no method has implemented exception code to handle the divide by zero error , hence default exception will print the exception In the order which the program has called,

For example: jvm🡪main🡪doStuff()🡪doMoreStuff()🡪sop(10/0);

The default exception handler is maintained by JVM. It will print the error by terminating the program abruptly.

Exception handling part2:



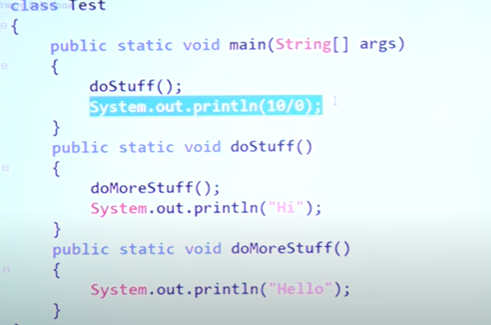
Will this program will run?

No, it will terminate abnormally after printing

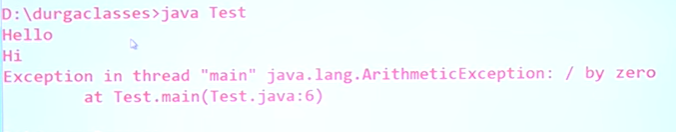
Hello

Java.lang.AE

Q

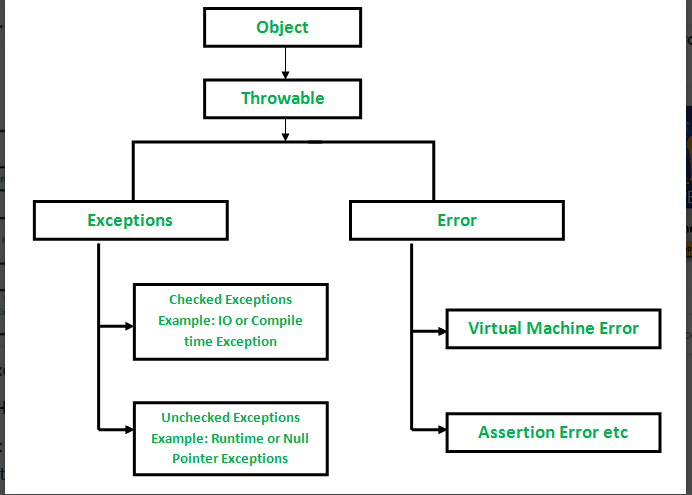


Output : abnormal termination



Exception hierarchy

Throwable [not an interface] class is the root of the exception class



Exceptions are recoverable whereas errors aren’t recoverable.

Error caused by lack of system resources.

Interview: Checked vs Unchecked Exception:

Checked Exception occurred at compile time and unchecked exception occurs at run time

It is a wrong statement because every exception happens only at run time. At compile we get syntactical error.

Example: Hall ticket is being carried or not before going to take exam is best example of compile time exception here there will be no problem even if you leave your hallticket but issue will happen at run time (the moment when you are going to take an exam). So parents will alert to take the hall ticket for the smooth process in exam hall.

another example is alert for carrying extra pen for exam or not. [HallTicketException][PenIsNotWorkingException][FileNotFoundException]

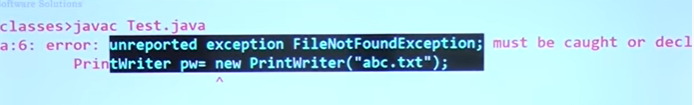
Checked: The exceptions that are checked by compiler whether the programmer is handling the exception or not for smooth run of program.

Exception which can occurs most commonly is checked by compiler is known as checked exception.

Example: opening a file : file not found exception.

At compile time these error will occur which it can be a candidate of runtime exception.

PrintWriter pw = new PrintWriter(“open.txt”);



Difference between Checked and Unchecked Exceptions part-2,

Unchecked Exception: exception like blast can happens in a school which is very rare hence won’t be alerted or compiler won’t raise an error for uncommon exceptions.

Example : sop(10/0) [no error at compile time]

Exceptions that are not checked by compiler is known as unchecked exceptions.

Class extends for runtime exception where as checked(compile time) extends Exception

Except Runtime Exception and Error are Checked Exception

**Difference between Fully - Checked and Partially- Checked Exceptions?**

If parent and child both are checked Exception then it is Fully Checked

Where as if child is unchecked and parent is checked is partly checked Exception

Only Throwable , Exception ---party checked

Example

IOException -🡪checked(Fully Chekced)

RuntimeException 🡪 unchecked

InterruptedException 🡪 Fully Checked Exception

Error 🡪 unchecked

Throwable 🡪 checked (partiallly checked) because

ArithmeticException 🡪 unchecked

Null pointer Exception🡪 unchecked

Exception 🡪 not part of runtime and error hence checked (partially checked)

**Customized Exception Handling by using Try -catch?**

Real use cases : Database connection can be open if program stops abnormally.

Try{

Risky code should be taken under this block

}catch{

Handling Code

}

############## Control Flow Inside TRY-Catch ################

Try{

Statement-1

Statement -2

Statement-3

}catch(exception e){

Statement -4

}

Statement -5

Case1: if there is no exception : 1,2,3,5,NT

Case2: if an exception raised at statement 2 and corresponding catch block matched

1,4,5,NT

Here the statement 3 will not be executed because if any problem happens inside a try block

The rest of the code of inside try block will not be executed.

Case3: if an exception raised at statement-2 and corresponding catch block not matched.

1,AT(ab normal termination)

If any exception rises outside of try block it is always abnormal termination.

Case4: if exception raised at statement 4 or 5?

AT

If any exception rises outside of try block it is always abnormal termination.

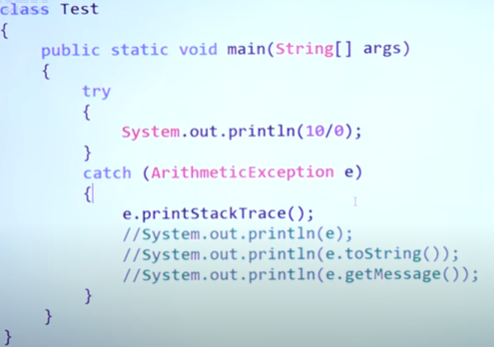
Methods to print Exception Information

1. e.printStackTrace() : for printing complete information
2. e.toString()

sop(e) or sop(e.toString()); : for printing name of exception : description

1. sop(e.getMessage()) : only for printing description

all above methods are present inside throwable class



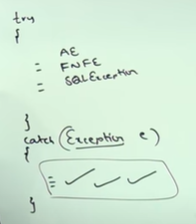
[3:09:40](https://www.youtube.com/watch?v=VHi9PedZCq8&t=11380s) Try with multiple catch

Need of the concept :

Interesting example: let’s say a student is asking 4-5 question and for every question the teacher is providing same answers which is abnormal.

In similar fashion there our code can look abnormal if we will have single catch block.

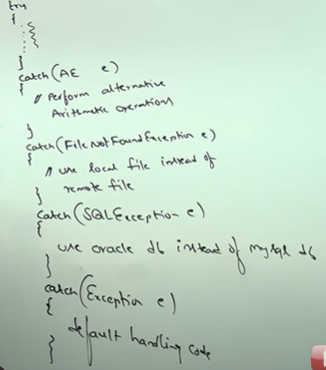
Not recommended



Here we are trying to handle every exception with single statement like giving answers of different question same is worst programing practice.

Example of :

Highly recommended to use this approach



But here is one loophole: catch block happens top to bottom manner ,

So if child exception is already present in parent exception , hence order should be maintained in proper

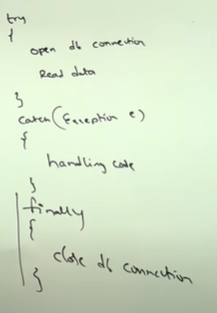
Child🡪parent --runs perfectly

Parent🡪 -- compile time error saying exception is already caught

Order of catch block is very important

:Finally Block:

Need of finally block: Resource deallocation code or cleanup code.



Specialty: Finally block will be executed irrespective of exception occurred or not.

Case1: if there is no exception

Try, Finally , block will be executed

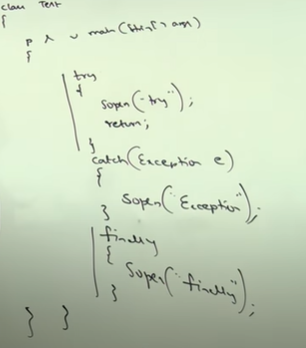
Case2: if an Exception raised and handled

Op/ try,catch,finally

Case3: if an exception raised but not handled

############### finally block vs Return Statement #######################

if there is return statement inside try catch block will finally block will be executed?



**Ans: Irrespective of return statement finally block will be executed.**

**Finally block dominates return statement.**

**###### Finally vs system.exit(0) #################**

Scenario.

Try{

Executing…….

Executing…………

Executing……….

System.exit(0)

}catch(Exception e){

}

Finally {

}

System.exit(0) is going to dominate finally block and jvm will stop as soon as system.exit(0) happens.

2nd scenario : When the return statement is defined in the finally block

finally

{

System.out.println("I am in finally block");

return 50;

System.out.println("Statement after return statement"); // Unreachable code.

}

Control Flow to try -catch-finally:

Try{

Statement-1;

Stamenet -2;

Statement-3;

}catch(x e){

Statement -4

}finally{

Statement -5

}

Statement -6

Case1: if there is no exception;

1,2,3,5,6,NT

Case2: if an exception raised statement2 and corresponding catch block

1,4,5,6,NT

Case-3 if an exception raised at statement-2 and corresponding catch block not matched

1,5,AT ----since no corresponding block is present so AT

Case 4: If an exception raised at statement -4;

AT, but before that finally block will be executed

Case5: if an exception raised at statement-5 or statement6

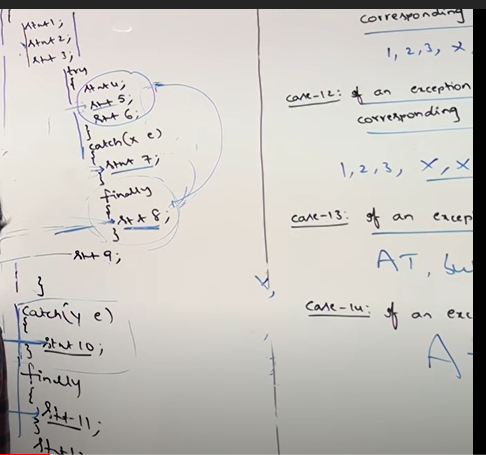
AT

Nested Try catch finally

try catch can be part of try block, catch block, finally block

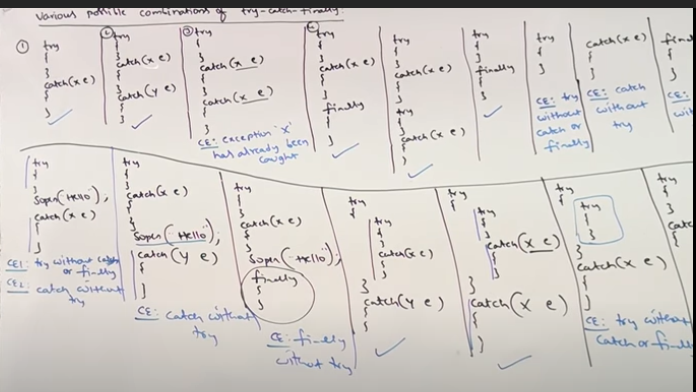
Control flow in nested

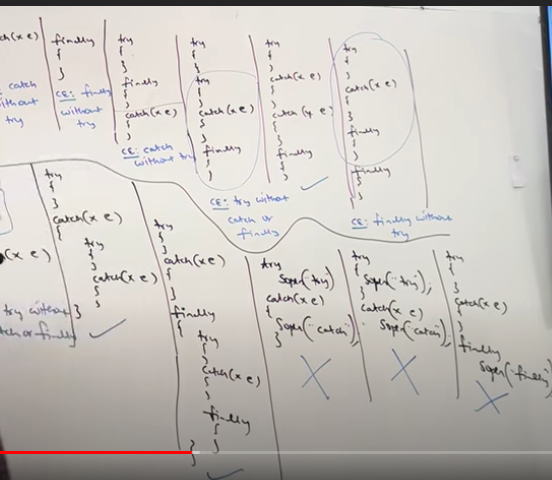
Point—once you entered in try block then only finally block will be executed .



[5:07:45](https://www.youtube.com/watch?v=VHi9PedZCq8&t=18465s) Various Possible combinations of try-catch-finally

* With try either catch of finally block should be present.
* There won’t be any catch without any try
* There won’t be any finally without try
* Order should be maintained try🡪catch🡪 finally
* Only one finally is allowed with one try block
* There can’t be any statement between try catch block.
* For try catch and finally block curly braces are important even if there is only one statement

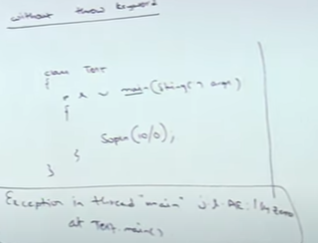




######### Throw Keyword ####################

Need of Throw key word: to handover our customize exception object to jvm manually

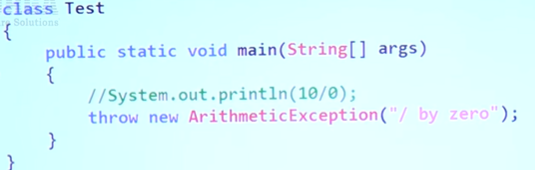
**Without throw keyword**



For this program exception will be handled by jvm internally via main method automatically

The same thing we want explicitly

**With throw keyword**



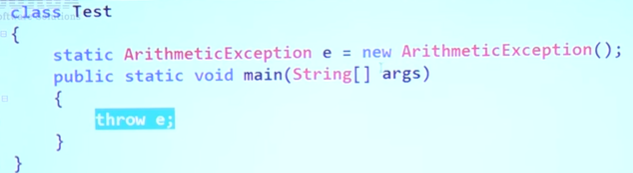
**With same exception as above**

[5:50:16](https://www.youtube.com/watch?v=VHi9PedZCq8&t=21016s) Imp cases related to throw keyword

Throw e:

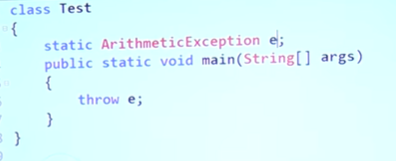
If e refer null then we will get NPE

Case1:





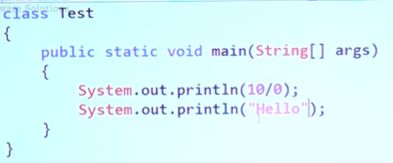
Case



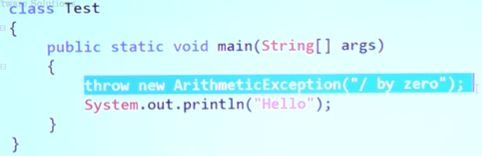
NPE

Case2:

After throw we can’t write any statement directly



Runtime exception : AE exception



Compile Error: Unreachable statement:

Reason : in SOP statement compiler doesn’t if there will be any RE or not but in case of THROW key word it is understood that there will be RE hence nothing is allowed after throw key word.

Case3:

Throw keywords are applicable only for throwable types like exception and error.

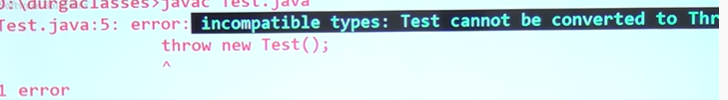
Class Test{

Psvm{

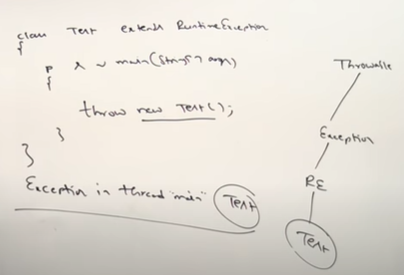
Throw new Test();

}

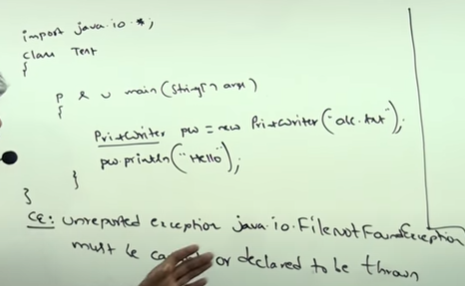
}

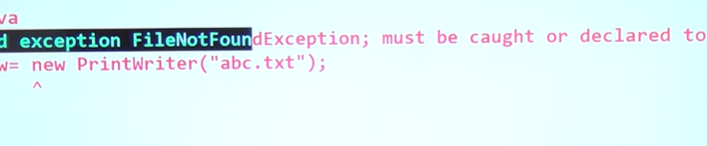


Where as this is allowed. Because Test class is now part of throwable

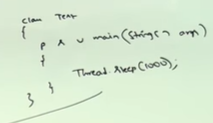


**Need and Usage of Throws Keyword:**





Example



This will throw compile time:

To solve the problem we have ,

1. Using try catch(manual handle )

Try{thread.sleep(1000)}

Catch(InterruptedException e){}

1. **Delegate the response to the caller (jvm or another method).**

**By using throws Keyword**

**Public class test{**

**Psvm() throws InterruptedException{**

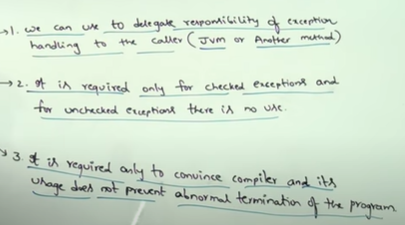
**Thread.sleep(1000)**

**}}**

**So if there is any chance of exception in our program we have to handle it.**

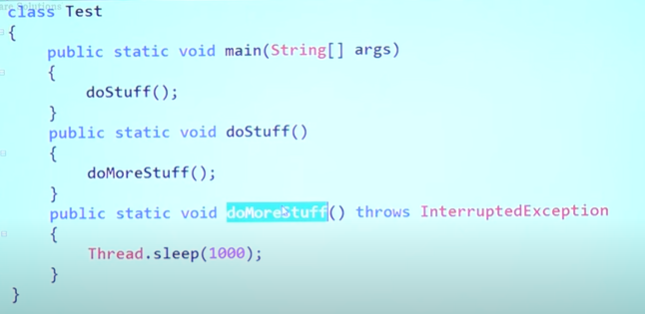
**Throws key word is used only for checked Exceptions.**

**Throws keyword important points:**



**Obviously the preferred approach is try catch as compare to throws keyword.**

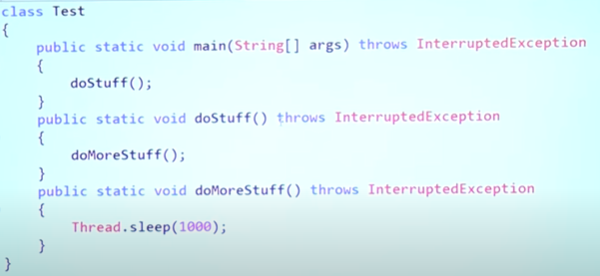
[6:34:01](https://www.youtube.com/watch?v=VHi9PedZCq8&t=23641s) throws keyword across multiple methods



Since doMoreStuff() is throws IE since it is called by doStuff() , since this is called by mby main method hence main also has to implement , above code will give the compile error.

Purpose of throws keyword: to delegates the exception to jvm or another method.

Correct code:



but lets say we remove

**throws key word from doMoreStuff() and doStuff() , will there will be compilation error?**

Yes. Because compiler is always going to ask the first method that is generating the exception.

Then latter subsequent function will be checked if they have implemented exception handling or not.

**Imp cases related to throws keyword**

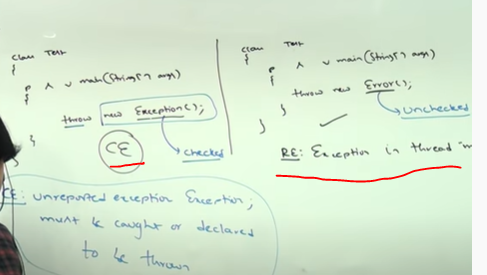
for classes we can not use throws where it is valid to use throws with method and constructor.

* **Throws is applicable only to method and constructor**
* **Throws keyword can be used only for throwable type(**error will be not convertible**)**
* **To make a class throwable we can extend Throwable, RuntimeException, Exception.**

**Special case:**

**Since Exception is checked type Exception hence the first case will not be able to compile where as second case compiles fine but RE.**

**Hecnce it is not advisable to use Throw(not throws) with checked exception.**



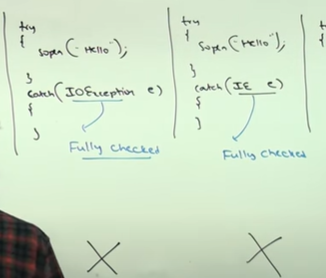
Important point:

If there is no chance of rising exception then compiler will not allow you to right try catch block,

Obviously still we can write try block for unchecked exception but for checked exception if there is no chance of rising compiler won’t allow to let you write a try catch,

Example

These code will not compile and it is for fully checked exception



CE: IOE is never raised in Corresponding block.

Exception Handling Keywords summary and various possible compile time errors

Try🡪 To maintain Risky code

Catch🡪 to maintain handling code

Finally🡪 to maintain clean up code

Throw 🡪 to handover our created exception object to the jvm manually

Throws 🡪 to delegate responsibilities of exception handling to the caller

Various possible compile time erros in exception handling:

* Try without catch or finally
* Catch without try
* Finally without try
* Unreported exception XXX must be caught or declared to be thrown
* Exception xxx is never thrown in body and corresponding try statement
* Exception XXX has already been caught (order of catch block)
* Unreachable statement
* Incompatible types: Test cannot be converted to throwable

Difference between final, finally and finalize()

1. Final : it is access modifier and applicable for classes,methods,variables
   1. If class is declared final then no once can extend this class ie inheritance is not possible
   2. If methods is declared final we can not override
   3. If variables declared as final we can not change the value.
2. Finally
   1. It is associated with try catch block
   2. To maintain cleanup code
   3. Finally block will be executed everytime except at system.exit(0)
3. Finalize() : it is method and associated with garbage collector
   1. Just before destroying the object the garbage collector will call the finalize method to perform the cleanup activities.

What is the difference between finally and finalize() cleanup,

Both works on different context

Finally is responsible for cleanup or close the resource which open at the time of try catch block where as any object related cleanup finalize() method is used.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Customized or User Defined Exceptions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The exceptions that are defined by programmer to meet the programming requirements is known as user defined Exceptions.

Example: trying to withdraw money from account and amount is greater than available balance : insufficient Balance

How to define user defined Exception

Every exception in java should be child to throwable class

**Class TooYoungException extends RuntimeException{**

**//constructor**

**TooYoungException(String message){**

**Super(message);**

**}**

**}**

**//another exception**

**Class TooOldException extends RuntimeException{**

**TooOldException(String message){**

**Supper(message);**

**}**

**}**

**Class main{**

**Psvm(String[] args){**

**Int age = Integer.parseInt(args[0]);**

**If(age>60){**

**Throw new TooOldException(“You have already crossed your marriage age”);**

**}else if(age<18){**

**Throw new TooYoungException(“please wait for some time you’re age isn’t valid”);**

**}else{**

**Sop(“thanks for registration,”);**

**}**

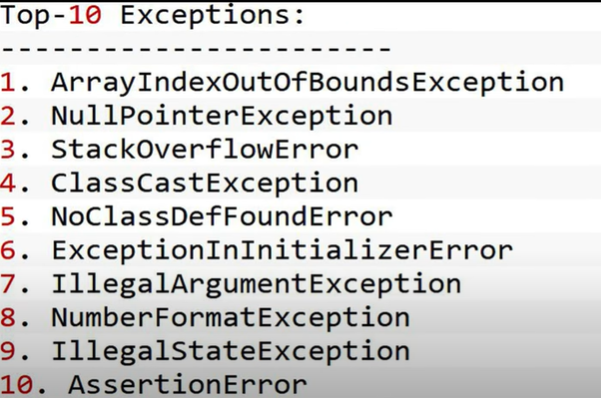
**}**

**}**

Loop holes :

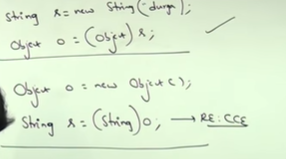
* Best use of throw keyword is used for customized exception not for inbuilt exception
* Need for supper(msg): TooYoungException 🡪 RuntimeException 🡪 Exception 🡪 Throwable, so to print the message we need supper keyword because the parent class has implemented the print statement, hence it will avail our message to default exception handler
* It is highly recommended to extend runtime exception : why : because RuntimeException is part of unchecked Exception and hence compiler will not rise any error for throw key word, otherwise if you use Exception(checked) then compiler will throw an error at throw key word.

**TOP – 10 Exceptions:**



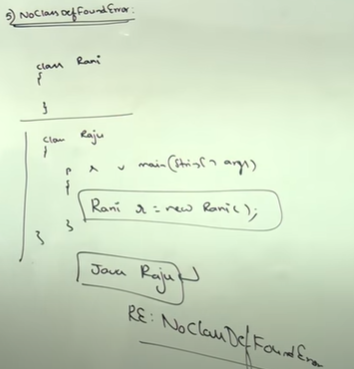
1. **AIOOBE🡪IOBE🡪RE🡪Exception🡪Throwable : hierarchy :**  it is unchecked exception
2. **ClasssCastException** 🡪 RE🡪Exception🡪throwable : while performing any typecasting if there is any issue arise it is kown as CCE

**Example :**



Child class can be typecased to parent where as parent class can’t be typecasted to child.

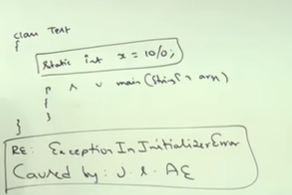
1. **NoClassDefFoundException :**



**While running the second class if first class is not present then we will get this exception**

1. **ExceptionInInitializerError🡪LinkeageError🡪Error🡪Throwable**

while performing static variable initialization and static block execution if JVM faces any issues then this exception arises.



1. **IllegalArgumentException 🡪 Exception 🡪 Throwable**

If we call any method with illegal argument

1. NumberFormatException🡪IAE🡪RE🡪E🡪Throwable

Int i = Integer.parseInt(“10);

Int I = Integer.parseInt(“ten”): //number format exception

1. IllegalStaeException 🡪 RE 🡪 Exception 🡪 Throwable

If we are calling any method at wrong time is known as illegalStateException

1. AssertionError: AE🡪Error

If assertion is fails then assertion error will occur.

[9:18:28](https://www.youtube.com/watch?v=VHi9PedZCq8&t=33508s) Try with resources

In try catch what ever resources we haven that needs to be closed with the help of finally block where as in try with resources no resource needs to be explicitly closed.

Advantage

* Readability
* Length of program is deceases
* Complexity decreases
* No finally block is required to close the resource

**Example:**

**Try(BR br = new BR(new FR(input.txt)){**

**Use br based on our requirement**

**Once control reaches end of the try block automatically br will be closed, we are not required to close explicitly**

**}(IOExcecption e){**

**//handling code**

**}**

Important conclusions

1. For try with resources we can take as many as resources we want provided they are separated with semicolon

Ex: try(R1;R2;R3){

==}

Try(FR fr = new FR(input.txt); Pw pw = new PW(“print.write”)){

==

}

1. What ever reference variable we are using in try resource are implicitly final

Example

Try(FR = fr = new FR(“input.txt”){

fr = new FR(abc..txt) ///**not allowed**

}

1. **From 1.7 version onwards try with resource is without catch and finally**

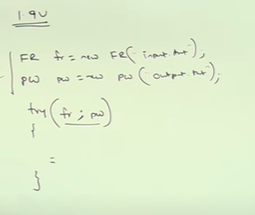
**Example:**

Try(R){

}

//no catch and finally block is required

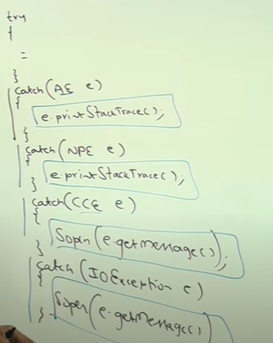
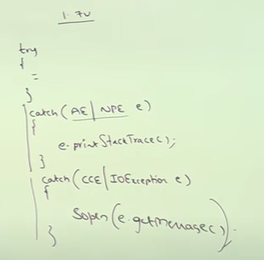
In 1.9 version we can assign the reference variable that is already used in try with resources.



**Multi Catch block**

Until 1.6 verison the problem is even though threre is only one handling code but we have to write multiple catch block like below

**Multiple catch block multi catch block(1.7 onwards)**

[10:00:54](https://www.youtube.com/watch?v=VHi9PedZCq8&t=36054s) Exception Propagation and rethrowing an exception

Show less