Exception handling

Exception:

* Due to some problem if program execution is stopped/ terminated/ disrupted then that problem is called exception.

Exception handling:

* If an exception occurs that exception we are handling that process is called exception handling.

Why do we need exception handling?

* Because by default java stop the program execution if there is any exception. To avoid that as a user I need to handle that exception.
* Means we are telling to java what to be perform.

Different types of exceptions?

There are two types of exceptions

* Checked Exception (Compile-time exception)
* If we identify at the time of compiling program is called compile time exception.
* 2. Unchecked Exception (runtime exception)
* If we can't identify the exception at the time of compiling is called run-time exception.
* If we want to handle exceptions java provides try, catch, finally blocks and also throw and throws keyword also we will use.
* If we are using this blocks means, we are taking control from java and we are handling those exceptions.

How to understand exception:

* In console we will get exception error is called Stack trace. Now let’s understand the error.

Stack trace

* Exception name
* Exception message
* Which line number
* Method info
* E.g.

Try

* After finding the exception line we need to keep in try block.

Try

{

result = fNumber/sNumber;

}

Catch

* After keeping exception in try block issue is not resolved. We need to tell to java what to do.
* Means we are throwing an exception, it need to be catch. So that’s why we are using catch block.
* In catch block we giving exception as a parameter.

Try

{

result = fNumber/sNumber;

}

catch (ArthmeticException ae)

{

system.out.println (ae.toString);

}

* Here exception we are storing in ae variable and inside the block we are telling what to perform.
* If there is no exception then catch block won't be executed.

Finally:

* If exception occur or not occur we have to execute a peace of code. Then we use finally keyword.

Try

{

result = fNumber/sNumber;

}

catch (ArthmeticException ae) {

system.out.println (ae.toString);

}

Finally

{

system.out.println (Finally block);

}

Throw:

* If we are throwing an exception by default java will catch.
* Then java will stop program.

Try

{

result = fNumber/sNumber;

}

catch (ArthmeticException ae) {

system.out.println (ae.toString);

throw ae;

}

Finally

{

system.out.println (Finally block);

}

Throws:

Throws keyword is used after main function, before compilation only we are telling to java that I am throwing an this type of exception. Don't give any error.

Advance level Exception handling

Try with resources:

* It is used to not leak any memory.
* If we are creating a steam instance/object we need to close that instance. If we forgot to close than memory get leak.
* To avoid that problem in java, Java itself close the instance by using try with resource concept.
* E.g.

try (resources) {

//statements

}

catch (Exception) {

//statement

}

* Here resource means stream instances.

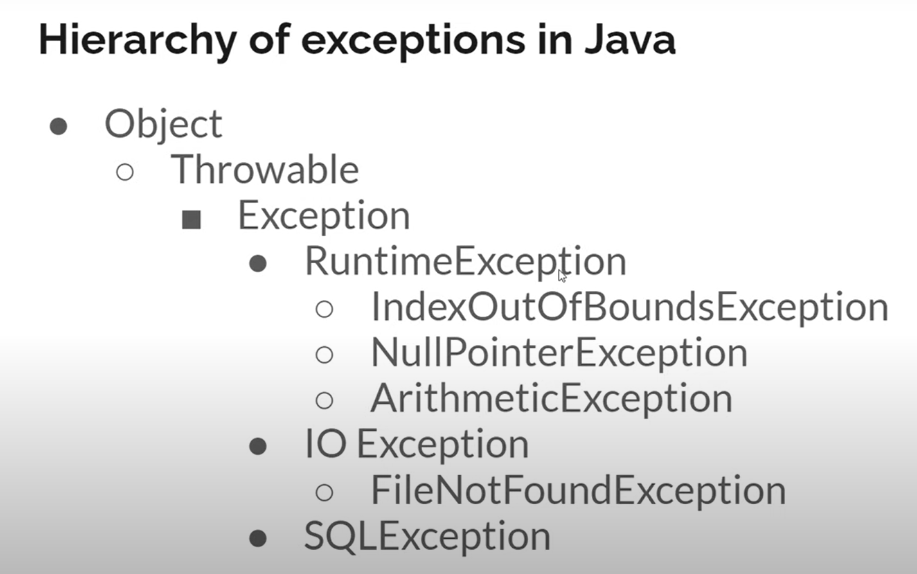
Different between parameters and resources?

* In parameters we only define variables buy using comma. But in resource we do initiation and multiple instance we write.
* Java throws some compile time exceptions and we can handle them in two ways

We can handle by using try catch block.

We can tell java to handle by using throws keyword.

* E.g. throws IOException, FileNotFoundException, etc...



* But in real time we will see many compile time exceptions and those many exceptions we can't write. To avoid that we need to write "Throwable" or "Exception" keyword.
* If we use those keywords java won't throw any compile time exceptions. Because those two keywords are super class for exceptions and can handle compile time exceptions.

Multiple catch block?

* In java we can get multiple exceptions.
* E.g.: ArrayOutOfbond exception, IO exception, filenotfound, arithmetic exception, etc...

And based on that exception we need to write multiple catch block.

try (resources) {

//statements

}

catch (ArrayOutOfBondException a) {

//statement

}

catch (ArthemeticException b) {

//statement

}

catch (FileNotFoundException c) {

//statement

}

* But we need to use only one try block(means inside try block one more try block not allowed) in that we can write multiple catch block.
* But in class we can have multiple try blocks.
* And we can handle multiple exceptions in one catch block by giving one variable name and in that block we are telling to execute that code if that exception found.

try (resources) {

//statements

}

catch (ArrayOutOfBondException | FileNotFoundException af) {

//statement

}

* Here we are separating exceptions by using pipe symbol.
* But in real time we won't use this combine exceptions concept.

Here instead in declaring multiple exceptions we can simple keep Exception keyword.

try (resources) {

//statements

}

catch (Exception a) {

//statement

}

* After keeping Exception, any other exception code won't be executed. Because for all the compile time exceptions super class is Exception so that particular code only execute.
* This is also we won't use in real time because for every exception there is different code.

Finally block:

* If we are using “System. Exit (0)" in catch block then up to catch block code execute and finally block code / after writing any code won't be executed.

Shortcut in eclipse for exception handling?

* Select peace of code and right click and click on surround with and select try/catch block option.

Or

* Write try and press ctrl + space

Throw custom/ user defined exception:

* If once exception block is executed and that block message is not good than we can write our own message.

try {

//statements

}

catch (ArthemeticException ae) {

throw new ArthemeticException ("Stupid fellow, don’t divide with zero")

//statement

}

* Here we are adding new exception with String.
* After we are throwing it.

User defined:

* If we want to throw an exception by application relatable.
* First create new class for exception and add super class as Exception by using extends keyword.
* If we want to pass a message, then we need to create a parameterized constructor.
* And in constructor we need to called super class message which is present in Trowable class.

