{ +91 - 88474 33272
Slack }

Agenda:

1) Exception handling

2) Why to handle exceptions

3) Nested Exceptions.

* EXCEPTIONS / ERRORS

something unusual / unexpected happen

ArrayOutOfBoundException.

```
{
    List <Integer> a = _ _ __
    a.get(0)
}
```

Calculator

```
a   ← input()
b   ← input()
ope ← division
```

$\left(a/b\right) \longrightarrow$ [Application crashes]

$(b == 0) \rightarrow$ Division by zero

what we want instead, is to gracefully end the application and provide users with some actionable and meaningful message

$$\begin{tikzcd}
 & \text{Exception} \arrow[dl] \arrow[dr] & \\
\text{Handle} \arrow[d] & & \text{leave} \arrow[d] \\
\text{message} & & \text{crash is propagated to custome}
\end{tikzcd}$$

(Stack Overflow)

1. A.I.O.B

   list. get(0)

2. String s;
   s. length( )

   N.P.E

3. Arithmetic Exception
   $5/0$

Handling Exceptions

Why?
↓

1) Prog should not crash

2) clients should be informed about the same.

try... catch block.

1) Add the suspected code in try block.

2) Add a catch block that can handle exception being thrown in the try block

→ catch (Exception e)
↓
how to handle that exception

```
int y=0 , x=0
try {
        z =  y/x          ← Exception thrown
      | z = y+x |  ←      Any line after the exception has been thrown won't be executed.
} catch ( A10B   e) {
        sout ("Something went wrong")
}
```

AE. thrown
not graceful

error msg

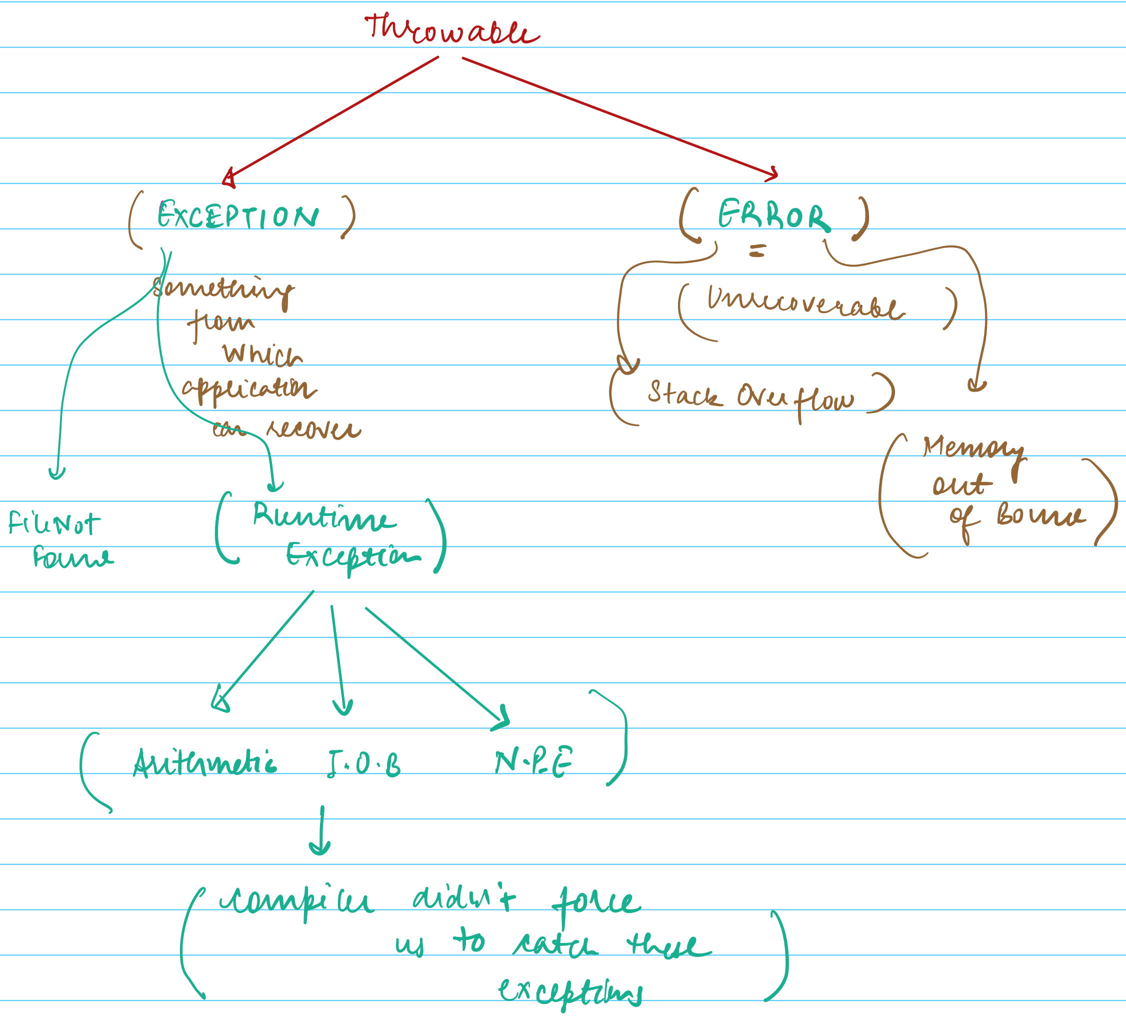
→ If something goes wrong in try block,

We try to handle the errors in the catch blocks.

finally( ) ← block.

```
try {
        z = y/x ;

} catch(            ) {



} finally( ) {
        db.shutdown()
        → Executed always

}
```

EXCEPTION CLASS HIERARCHY



CHECKED	UNCHECKED.
① some of the known issues	→ caused due to buggy faulty code
② [can be handled by the function or declared to be thrown further]	Arithmetic, NPE, IOB

10:24 pm

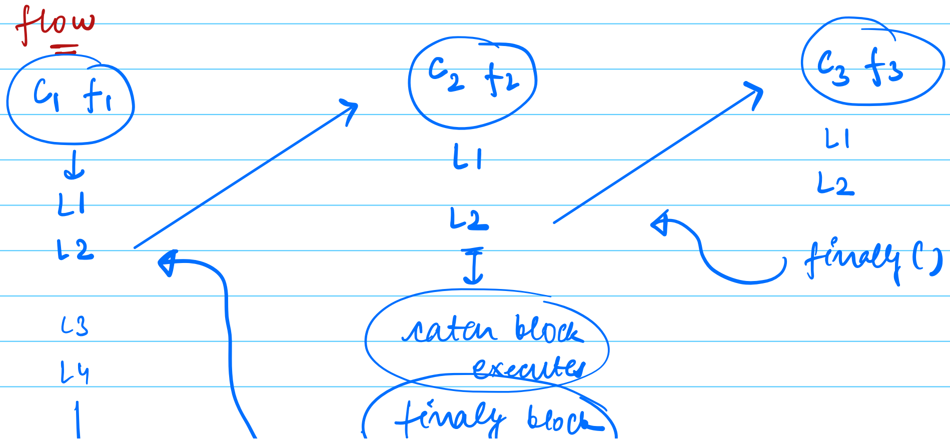
Nested try / catch :

```
c1
fn1

try {
    L1 →
    L2 →
    c2.fn2()
    L3
    L4
} catch (NPE e) {

} finally ( ) {

}
```

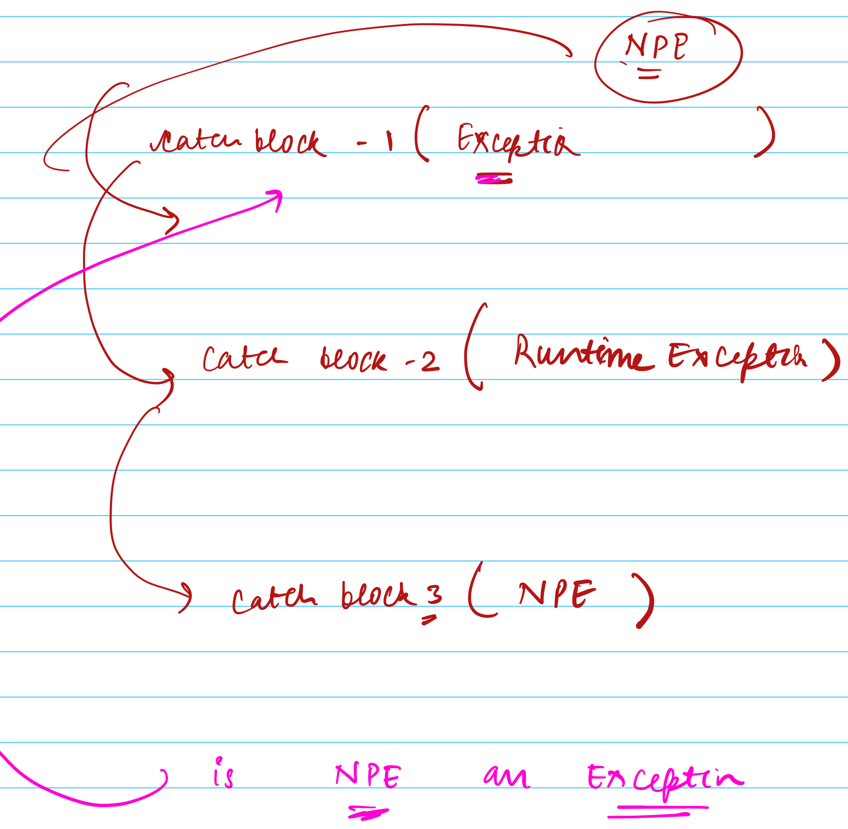
```
c2
fn2

try {
    L1
    L2
    (c3.fn3())
    ~~L3~~ x
    ~~L4~~ x
} catch (NPE e) {

} finally ( ) {

}
```

```
c3
fn3

try {
    L1
    L2
    throw NPE
    L3 x
    L4 x
} catch (IOB e) {

} finally ( ) {
    ?
}
```

flow



finally()


NPE
catch block - 1 (Exception)
Catch block - 2 (Runtime Exception)
Catch block 3 (NPE)
is NPE an Exception