10 SATURDAY DECEMBER

# Exception Hierarchy

(java.lang.Ex^n)

Object → Parent of all classes in Java.

Throwable

Error → caused by H/W Environment problems at runtime. (cannot be handled)

Exception → caused by S/W problem in appn

- SQL Excep^n
- ClassNotFound Ex.
- IOEx^n
  - FileNotFoundEx^n
  - EOFEx^n
- InterruptedExc^n
- RuntimeEx^n
  - IndexOutOfBoundsEx^n
    - String "
    - Array "
  - ArithmeticEx^n — e.g divide by 0
  - NumberFormatEx^n — converting a string into number
  - NullPEx — getting length of a null string
- ParseEx^n

m1()
{ m1() }
(eg. recursive call)

StackOverflowError, OutOfMemoryError, AssertionError, NoClassDefFoundError

→ Super class of all exception is Throwable.

→ Error is an excep^n that happens at runtime

**Unchecked Ex^ns:** RuntimeEx^n & all its subclasses including Error. are known as.
(RuntimeEx^n)
↳ These excep^ns are unknown to compiler & occurs at runtime

**Checked Ex^n:** Throwable & it subclasses except Error & RuntimeExcep^n.
(compile time Ex^n)
↳ are known to compiler & checked at compile time only

11 SUNDAY

2016

MONDAY DECEMBER 12

## Alternative to try-catch:

**throws:** → used with the method signature
→ can throw/declare multiple excep$^{n}$ sep. by ,
~~used to throw~~ checked exceptions only.
↓ mostly.

- Not Suitable for → Robust coding, as remaing code will not execute after the pt of excep$^{n}$ & program is terminated.
- Used by novices
- A bit simpler to write, increases code Readability

⇒ try-catch is more robust than throws & throw.
⇓ Not a robust way.

**throw:** → used within the method
→ can not throw multiple excep$^{n}$ at a time.
→ to throw user defined excep$^{n}$ as per the need of coding.
↳ Remaing code is not executed, prog. terminated

Mostly used to throw user defined excep$^{n}$.
↳ can throw system defined excep$^{n}$ as well. Java

### User Defined Excep$^{n}$

```
public class YesBank
{
  psvm(s...)
  { int bal=100, withdraw=1000;
    if(bal<withdraw)
    { throw new ArithmeticException("No Money Please");
    }
    else
    { sop("Draw & enjoy");
    }
  }
```

No connection with Bank balance.

To have meaningfull excep$^{n}$ name, we can create our own.

2016

→ try-catch, throw, throws can be used with both Checked & Unchecked.

~~→ Runtime excep~~

13 TUESDAY DECEMBER
348-018 • WK 51

```
class NoBalanceException extends Exception   // Checked excep^n
{
    public NoBalanceExcep^n(String problem)
    {   super(problem);
    }
}
```

```
public class YesBank
{
    PSVm(S...a) throws NoBalanceExcep^n   // hv to throw otherwise compile time error.
    {   int bal = 100, withdraw = 1000;
        if (bal < withdraw)
            throw new NoBalanceException("No balance");
        else
            sop("Draw & enjoy");
    }
}
```

→ Here superclass of NoBalanceEx is a checked Excep^n so we it is a must to throw in main() declare line. if superclass were RuntimeExcep^n, then no need.

O/P: Excep^n in thread "main" NoBalanceExcep^n: No balance
line java:15

2016

if we ~~throw~~ use 'throws' to throw a checked excep$^{n}$ we need to use throws with same Ex$^{n}$ or any of its superclass.

if Unchecked → we don't need to use either throws or try-catch
↳ code will compile.

WEDNESDAY 14 DECEMBER

printing excep$^{n}$ msg: 3 ways

- `sop(e);` → excep$^{n}$ obj thrown by JVM
- `e.getMessage()`
- `e.printStackTrace()`

} method of Throwable class

otherwise compilation failure

↓ or we need to surround throw with try-catch

L&T:

```
class ABC
{
    ABC abc = new ABC();
    psvm(s...a)
    {
        throw new Exception("abc");
    }
}
```

object can be created outside methods.

if you make it RuntimeExcep$^{n}$ it will compile.

compil$^{n}$ error: java.lang.Excep$^{n}$

cause Exception is a checked excep$^{n}$ so while throwing a checked excep$^{n}$ main declr$^{n}$ should be like:

psvm(s...a) throws Exception.

↓

then code will be compiled & excep$^{n}$ will be thrown at runtime.

Excep$^{n}$ in thread "main" java.lang.Excep$^{n}$: abc

in Unchecked case : → java.lang.RuntimeExcep$^{n}$: abc

→ User-defined excep$^{n}$ acts as Checked Ex$^{y}$ so we need throws or try-catch (surround throw by try-catch) ↓ it depends

15 THURSDAY DECEMBER
350-016 • WK 51

→ throw : can not throw multiple excep$^{n}$.

9 Checked excep$^{n}$ can not be propagated
10 with throw only

11 → throw a followed by an instance
12 → used within the method

throws → dealing can multiple excep$^{n}$

throws
Checked excep$^{n}$
"
can be

" — by a class
→ used with method signature

Sy → We

1 try-catch : System throw excep$^{n}$, we handle it
throws → Sy → Sy.
2 throw → We → Sy

3

→ Parent class for Error & Excep$^{n}$ : Throwable

↳ try should be followed by either catch or finally
we but should catch the excep$^{n}$ otherwise code will not execute

5

→ excep$^{n}$ can be rethrown : yes

→ can child class overriding method declare an excep$^{n}$ if parent class method doesn't throw an excep$^{n}$ ?
↳ Yes, but only unchecked excep$^{n}$, not checked.

↳ try { int a=10/0; } ⇒ in this case remaining code will not be executed as excep$^{n}$ is not caught anywhere

2016 finally { }

## Rules for Local variables :-

→ L.V. cannot use any of access modifier since its scope is only inside the method → final is the only key word that can be used with L.V.

→ Local var are not assigned a default value, hence they need to be initialized.

FRIDAY
DECEMBER
16
WK51 • 351-015
2016

Excep$^{n}$: an error event that can happen during exec$^{n}$ of prog. & stops the exec$^{n}$ flow.

08 THURSDAY DECEMBER 343-023 • WK 50

→ Exception Handling → so that remaining code can execute

→ Exception Handling makes a language Robust

programming errors → compile time (bugs): Checked Ex$^{n}$
→ Runtime (exceptions): Unchecked

Compile-time Error Causes:
typing errors

Runtime Error Causes
Due to input errors
— H/W errors → trying to copy a file & there is No space → No space on disk. → can not be handled
— Logical errors
(also called semantic errors)

↳ if file to be opened doesn't exist:
FileNotFoundExcep$^{n}$
↳ file to be opened in wrong format Demo.java? Demo.java.txt?
ClassNotFoundEx$^{n}$
↳ M/w is unable to connect as you might hv given wrong ip address → handled by MalformedURL Ex$^{n}$
↳ The sleep time (Inactive time) of a thread may be interrupted → InterruptedExcep$^{n}$

2016

FRIDAY
DECEMBER 09

When to use try-catch

identify which statement in the code may give problem to the user at run time.

e.g. whenever ur code ask for some input from user there may be some problem.

↓ handled by ArithmeticExⁿ

↳ catch block executes only when problem arises in try, otherwise catch is not executed but finally alway executes irrespective of excepⁿ in try.

finally Block → contains clean up code.

↳ There may be some statement which must be executed before program is terminated even if excepⁿ is not handled successfully

e.g. if appⁿ throw 3rd type of excepⁿ, progʳ is handling only two types.

e.g. { closing of streams in I/O operⁿ
JDBC Connections
Sockets in LAN programming

try {
}
finally {
}
↓
correct.

But remaing code will not execute as excⁿ was not caught

2016