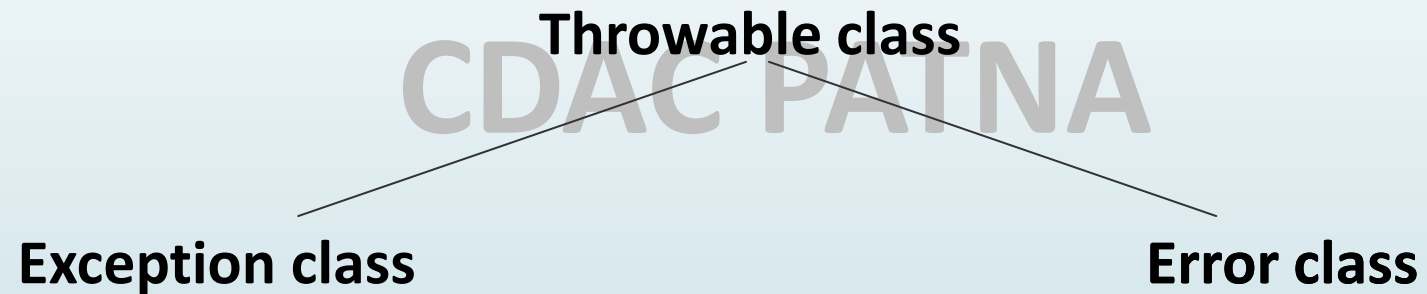




Exception Handling

Exception

- An unexpected / unwanted event that disturb normal flow of the program is called exception.
- For all exception and errors (Throwable class) is the root class.





➡ **Exception –**

Most of the time exception are caused by program and these are recoverable

➡ **Error –**

Most of the time , errors are not caused by our program i.e they are due to lack of system resources. Errors are non-recoverable

CDAC PATNA

Exception Hierarchy—

Exception class

IOException

FileNotFoundException

EOFException

RemoteException

SQLException

InterruptedException

RuntimeException

IndexOutOfBoundsException

NullPointerException

ArithmeticException

Error class

Assertion Error

VMError

StackOutOfFlowError

OutOfMemory Error

Linkage class

Verify Error

Checked & Unchecked Exception

➤ Checked Exception –

Compiler is responsible to check the checked exception

Ex- FileNotFoundException, InterruptedException

➤ Unchecked Exception –

There are some exception i.e not checked by compiler such type pf exception is called unchecked exception

Ex- ArithmeticException

Fully Checked & Partially Checked Exception



- Fully checked-

When parent class and child class both are checked

- Partially checked –

When parent class is checked but child need not to be checked

**NOTE – In java there are only two partially checked exception
“Throwable” & “Exception”**

- 
- 
- IOException-->Checked (Fully Checked)
 - RuntimeException-->Unchecked
 - InterruptedException-->Checked (Fully Checked)
 - Error-->Unchecked
 - Throwable-->Checked (Partially Checked)
 - ArithmeticException-->Unchecked
 - NullPointerException-->Unchecked
 - Exception-->Checked (Partially Checked)
 - FileNotFoundException-->Checked (Fully Checked)

➤ Exception Handling by using try catch–

Syntax -

```
try {  
    // risky code  
}  
catch (Exception e )  
{  
    // Handling code  
}
```

CDAC PATNA

- Code in which may chance of raised an exception is called risky code and we define that code inside try block and corresponding handling code we have to define inside catch block



➡ Methods (provide the exception information):-

i. `printStackTrace()`

ii. `toString()`

iii. `getMessage()`

CDAC PATNA



Try with multiple catch blocks

- For every exception type, a separate catch block is required
 - If multiple catch block is present then order of catch block is very important we have to take child first & then parent otherwise we get compile time error.
- CDAC PATNA

► Finally block :-

finally block is always executed either exception is raised or not or, whether handled or not .

Syntax-

```
try{  
    // risky code  
}  
catch(Exception e)  
{  
    // handling code  
}  
finally  
{  
    // cleanup code  
}
```

CDAC PATNA

Control flow in try catch finally

➤ Example-

```
try{  
    stat 1;  
    stat 2;  
    stat3;  
}  
catch(---)  
{  
    stat 4 ;  
}  
finally  
{  
    stat 5;  
}  
statement 6;
```

CDAC PATNA



Nested try-catch-finally

- We can keep try-catch-finally block with in try block
- We can keep try-catch-finally block with in catch block
- We can keep try-catch-finally block with in finally block

CDAC PATNA



Various possible combinations of try catch finally

- Curly braces are compulsory
- Order is important (first try ,second catch and then finally)
- try without catch or finally is invalid
- catch without try is invalid
- finally without try is invalid



Throw exception

- We can use throws keyword to delegate responsibility of exception to the caller method(it may be another method or JVM) then caller method is responsible to handle that exception .
- When JVM handle exception then only abnormal termination possible .
- **throws** keyword required for only checked exception
- We can use throws keyword for methods and constructors but not for classes

Difference between final, finally ,finalize()

- **final** is access modifier i.e applicable for classes, methods and variables
- **finally** is a block associated with try-catch ,so inside the finally block we keep cleanup code .Finally block will be executed always ,whether exception raised or not , handle or not handled .
- **finalize() method** - This method is associated with garbage collector .

Just before destroying an object garbage collector calls finalize() method to perform cleanup activity .

Once finalize() method completed immediately the GC destroy that object.



THANK YOU

CDAC PATNA