Exception Handling

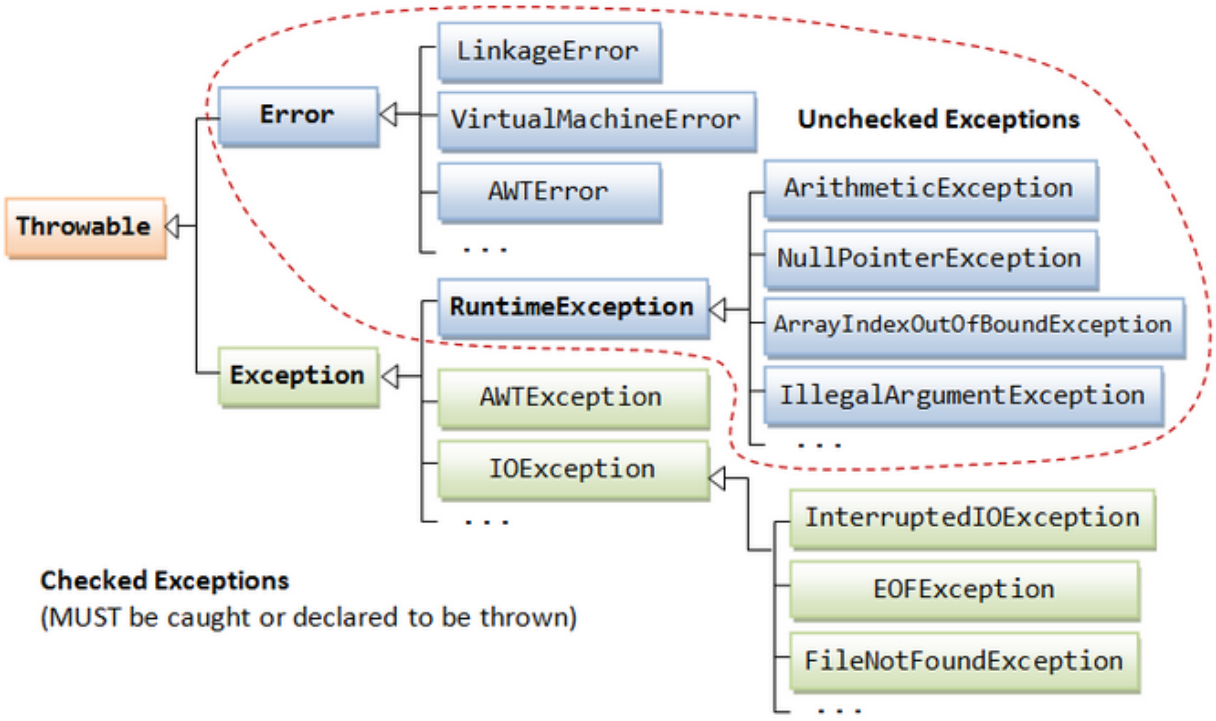
Information regarding Exception

- Dictionary meaning of the exception is **abnormal termination**.
- An un expected event that disturbs or terminates normal flow of execution called exception.
- If the application contains exception then the program terminated abnormally the rest of the application is not executed.
- To overcome above limitation in order to execute the rest of the application **must handle the exception**.

In java we are having two approaches to handle the exceptions.

1) By using try-catch block.

2) By using throws keyword.



Exception Handling:-

- The main objective of exception handling is to get normal termination of the application in order to execute rest of the application code.
- Exception handling means just we are providing **alternate** code to continue the execution of remaining code and to get normal termination of the application.
- Every Exception is a predefined class present in different packages.

- ✓ ***java.lang.ArithmeticException***
- ✓ ***java.io.IOException***
- ✓ ***java.sql.SQLException***
- ✓ ***javax.servlet.ServletException***

The exception are occurred due to two reasons

a. Developer mistakes

b. End-user mistakes.

i. While providing inputs to the application.

ii. Whenever user is entered invalid data then Exception is occur.

iii. A file that needs to be opened can't found then Exception is occurred.

iv. Exception is occurred when the network has disconnected at the middle of the communication.

Types of Exceptions:-

As per the sun micro systems standards The Exceptions are divided into three types

1) Checked Exception

2) Unchecked Exception

3) Error

checked Exception:-

- The Exceptions which are checked by the compiler at the time of compilation is called Checked Exceptions.

IOException,SQLException,InterruptedException……..etc

- If the application contains **checked exception the code is not compiled** so must handle the checked Exception in two ways
- *By using **try-catch** block.*
- *By using **throws** keyword.*

If the application contains checked Exception the compiler is able to check it and it will give intimation to developer regarding Exception in the form of compilation error.

Exception handling key words:-

1) try
2) catch
3) finally
4) throw
5) throws

Exception handling by using Try –catch block:-

Syntax:-

try
{
exceptional code;
}
catch (ExceptionName reference_variable)
{
Code to run if an exception is raised;
}

Example : code.

1) Whenever the exception is raised in the try block JVM **won't terminate the program immediately** , it will search **corresponding** catch block.
a. If the catch block is matched that will be executed then rest of the application executed and program is **terminated normally**.
b. If the catch **block is not matched** program is terminated **abnormally**.

Example code : //

- *If there is no exception in try block the catch blocks are not checked*
- *in Exception handling independent try blocks are not allowed must declare* ***try-catch*** *or* ***try-finally*** *or* ***try-catch-finally.***
- ***In between try-catch blocks it is not possible to declare any statements must declare try with immediate catch block.***
- If the exception raised in try block remaining code of try block won't be executed.
- Once the control is out of the try block the control never entered into try block once again.
- The way of handling the exception is varied from exception to the exception hence it is recommended to provide **try with multiple number of catch blocks.**
- By using Exception catch block we are able to hold any type of exceptions.

- if we are declaring **multiple catch blocks** at that situation the catch block order should be child to parent shouldn't be parent to the child.
- *It is possible to combine two exceptions in single catch block the syntax is catch(ArithmeticException | StringIndexOutOfBoundsException a).*

Finally block:-

1) Finally block is always executed irrespective of try and catch.
2) It is used to provide clean-up code

a. Database connection closing. **Connection.close();**
b. streams closing. **Scanner.close();**
c. Object destruction . **Test t = new Test();t=null;**

It is not possible to write finally alone.

a. try-catch-finally -- valid
b. try-catch -- valid
c. catch-finally -- invalid
d. try-catch-catch-finally -- valid
e. try-finally -- valid
f. catch-catch-finally -- invalid
g. Try -- invalid
h. Catch -- invalid
i. Finally - invalid.

Syntax:-

```
try
{ risky code;
}
catch (Exception obj)
{ handling code;
}
finally
{ Clean-up code;(database connection closing, streams closing……etc)
}
```

Example:-in only two cases finally block won't be executed

Case 1:- whenever we are giving chance to try block then only finally block will be executed otherwise it is not executed.
Case 2:-In your program whenever we are using **System.exit(0)** the JVM will be shutdown hence the rest of the code won't be executed.

throws

The main purpose of the throws keyword is **bypassing** the generated exception from **present method to caller method.**

- Use throws keyword at method declaration level.

It is possible to throws any number of exceptions at a time based on the programmer requirement.
If main method is throws the exception then JVm is responsible to handle the exception.

Example code : //

Throw:-

1) The main purpose of the throw keyword is to creation of Exception object explicitly either for predefined or user defined exception.
2) Throw keyword works like a try block. The difference is try block is automatically find the situation and creates an Exception object implicitly. Whereas throw keyword creates an Exception object explicitly.
3) Throws keyword is used to delegate the responsibilities to the caller method but throw is used to create the exception object.
4) If exception object created by JVM it will print predefined information **(/ by zero)** but if
exception Object created by user then user defined information is printed.
5) We are using throws keyword at method declaration level but throw keyword used at method implementation (body) level.

Different types of Exceptions in java:-

Checked Exception ---> Description
ClassNotFoundException --> If the loaded class is not available
CloneNotSupportedException-->
Attempt to clone an object that does not implement the Cloneable interface.

- **IllegalAccessException** --> Access to a class is denied.
- **InstantiationException** --> Attempt to create an object of an abstract class or interface.
- **InterruptedException** --> One thread has been interrupted by another thread.
- **NoSuchFieldException** --> A requested field does not exist.
- **NoSuchMethodException** --> If the requested method is not available.

UncheckedException Description

- **ArithmeticException** Arithmetic error, such as divide-by-zero.
- **ArrayIndexOutOfBoundsException** Array index is out-of-bounds.(out of range)
- **InputMismatchException** If we are giving input is not matched for storing input.
- **ClassCastException** If the conversion is Invalid.
- **IllegalArgumentException** Illegal argument used to invoke a method.
- **IllegalThreadStateException** Requested operation not compatible with current thread state.
- **IndexOutOfBoundsException** Some type of index is out-of-bounds.
- **NegativeArraySizeException** Array created with a negative size.
- **NullPointerException** Invalid use of a null reference.
- **NumberFormatException** Invalid conversion of a string to a numeric format.
- **StringIndexOutOfBoundsException** Attempt to index outside the bounds of a string.