CYBAGE
Delivering Value. Scientifically.

Exception Handling

Author & Presenter -Asfiya Khan
(Senior Technical Trainer)

www.cybage.com

Agenda

- What are Exceptions
- OOP way of Exception Handling
- Types of Exceptions
- ARM – Automatic Resource Management
- User Defined Exceptions

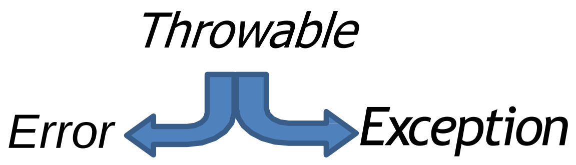
What is Exception

- Exceptions are nothing but some anomalous conditions that occur during the execution of the program.

- Exceptions are the conditions or typically an event which may either cause a running program to terminate or change its normal flow of execution

Hierarchy Of Exception Classes

- Throwable is the super class in Exception hierarchy
- It is in java.lang package.
- The Throwable class is further divided into two subclasses :-



Error Class

Errors :

- These are the situations which can not be handled or can not be recovered from.
- If any such situation occurs, program will terminate.
- Those situations are rare & usually fatal

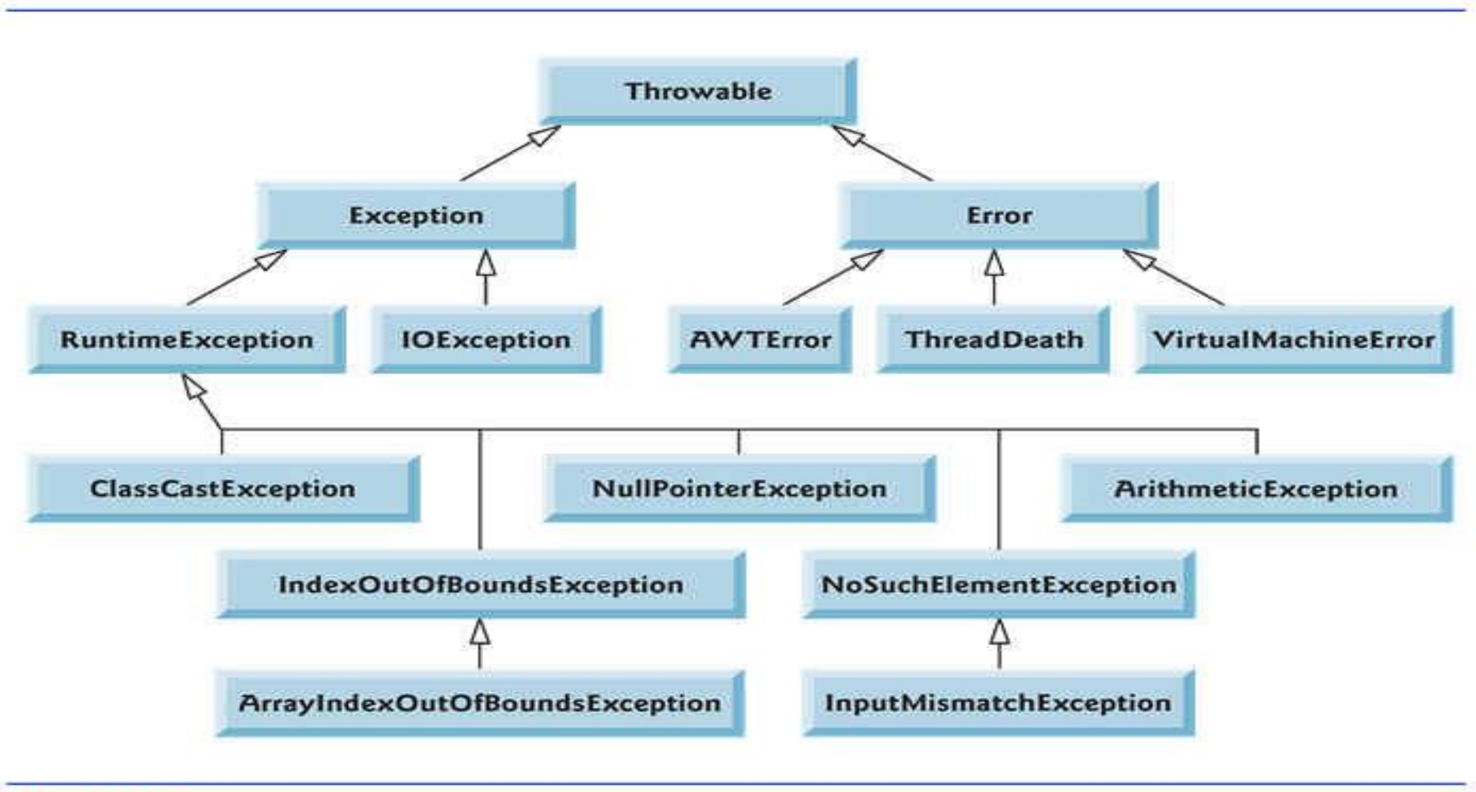
Example: *OutOfMemoryError*
some internal error in JVM

Exception Class

Exceptions:

- These are the situations which can be handled.
- It is possible to recover from such situations & continue with program execution further.
- However normal flow of execution may change in such situations.

Exception Hierarchy



OOP Way of Exception Handling

- The various keywords for handling exceptions are below.

  try
  catch
  finally
  throw
  throws

- The three exception handler components are used to catch and handle the exceptions. These are try, catch and finally clause.

Syntax of Exception Handling

- Using try and catch:

- The syntax for the usage of try, catch and finally block is given below.

```
try  {
        .........
   }
catch(<exceptionclass1> <obj1>){
        .........
   }
 finally{
        .........
     }
```

Try Block

- The code which is capable of generating some kind of
- exception (Checked Exception) must be written in Try block.

- Try block should be immediately followed by either a catch or finally block.

- A try block can have multiple catch blocks

Catch Block

The Catch Block is used as exceptional-handler. The Exception that arises in the try block is handled by the Catch -Block.

What is the problem with following code?

```
try {
        ------
       ------
       ------
   }
catch (RuntimeException re){   ---}
catch(NullPointerException      ne){---}
catch (Exception e) {-----}
catch (Throwable t){----}
```

Finally Block

- Finally block gets executed in either of the cases :

  if Exception occurs or it does not occur

  advantage : write clean up code in “finally”

Types Of Exceptions – Checked Exception

- Checked Exceptions: Those are the exceptions where compiler will ensure that programmer has handled those situations.

- The code capable of generating such kind of exception has   to be embedded in Try- catch block or compiler will complain (or method where this code resides should declare it in "throws" clause)

- Example : *IOException , SqlException*

Types Of Exceptions – Unchecked Exception

Unchecked Exceptions:

- Compiler will not check whether programmer has handled those situations or not.

- Example: all RunTimeExceptions

How to throw Exception in Java

Whenever an exceptional situation occurs, an object of that particular Exception is created & thrown.

We can also create an Exception object & explicitly throw it using keyword *"throw"*

Example:

```
try{   if(   …some condition)
           throw new Exception();
            -----
         }
              catch(Exception e)
        {
                 …….some handling code       here
         }
```

Important points

- If the method throwing a checked exception does not catch it then method must declare it as throws that exception
- Exceptions can cascade from a method to its caller & so on till main() method throws that exception.
- An overridden method in subclass can not throw more checked exception than the original method in super class

ARM – Automatic Resource Management

- Resources such as Connections, Files, Input/OutStreams , etc. should be closed manually by the developer by writing bog-standard code.

- Usually we use a try-finally block to close the respective resources.

- See the current practice of creating a resource, using it and finally closing it:

```
try(resources_to_be_cleant) {
            // your code
}
```

Multi-Catch

- There are a couple of improvements in the exception handling area.
- Java 7 introduced multi-catch functionality to catch multiple exception types using a single catch block.

```
 public voidnewMultiCatch()
{
     try{
              methodThatThrowsThreeExceptions();
        }
        catch(ExceptionOne | ExceptionTwo | ExceptionThree e) {
              // log and deal with all Exceptions }
        }
}
```

User Defined Exception

- Sometimes you may need to handle certain situations which are specific to your application.
- In such cases, you can create your own User Defined Exceptions.

Example:

```
class InsufficientBalanceException extends Exception {
        ------------
        ------------
}
```

User Defined Exception

```
public void withdraw(int amt)
 {
     try{
              if (balance < amt)
               throw new InsufficientBalanceException();
              ----
          }
     catch(InsufficientBalanceException e)
          {    handling code// some msg…}
 }
```

ANY
QUESTIONS
?

CYBAGE
Delivering Value. Scientifically.

Thank You!

Any Questions ?

www.cybage.com