**Exception handling**

**What is exception?**

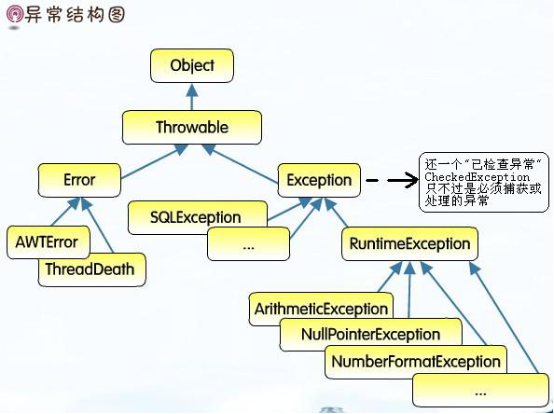
Exception may occur anytime during programme development, such as an invalid user input, an external system unable to be responded to or a simple application error. For these scenarios, we need applications to fix the problem, otherwise the whole application programme will break down.

Therefore, *an exception is an error event occurred while a programme is running, which interrupts the normal flow of the programme.*

*But it's important to know that an exception is not equivalent to an error:*

* Exception: Exceptions are problems existed in a programme, which can be predicted and recovered. Most exceptions indicate medium-to-small level problems. Exceptions often arise in certain situations, specially in certin methods and operations of codes. For example, such as an attempt to divide by zero, or anomalous database connection, etc.
* Error: Errors are serious problems confronted while running a programme. Most errors have nothing to do with the operations executed by a programmer, but indicate the problems of the JVM (Java virtual machine) while codes are running. For example, when JVM doesn't have enough memory resources to continue the execution, OutOfMemoryError will occur.

*Here is a diagram showinng the relationship between exception and error:*



**Exception types:**

* Checked exceptions (exceptions occurred not during runtime): A checked exception is an exception checked by a compiler. When a method is declared exception, the compiler will do forced execution or declares the rule. For example, SqlExecption is a checked exception. When connected to a database, a compiler won't get it passed and compile normally unless you fix the exception.
* Unchecked exceptions (runtime exceptions): No action is needed for this type as a compiler won't check if such exceptions have been fixed. On most cases, wrong codes logic is the cause to blame. A programmer should avoid this kind of exceptions by improving the logic of codes. For example, for an array with 3 units of length, if you set the index of 3, you will see the out of bounds exception. Such exceptions cannot be checked by JVM, but need to be detected by programmers.
* Custom exceptions: Custome exceptions are used to provide accurate definitions for one or more potential errors in the codes, as a way to indicate certain error types of applications. Their definitions are customized by software programmers pursuant to the needs of the system.

**How to handle exceptions?**

Exception handling is a process to improve the system stability where an exception is occurred by special methods, in a way to maintain the normal flow of the application. The usual methods include hanlding an exception and declaring an exception.

* Hanlding an exception: an exception can be handled on site, often by using the statement block of "try...catch...".
* Declaring an exception: an exception cannot be handled where it's occurred, so it should be thrown to others to handle. Meanwhile a declaration should be made in the method definitions.

*Now we will explain in detail:*

**try...catch...finally...**

*Grammar:*

**try** {

*//code*

} **catch** (ExceptionType typeName) {

*// handle exception code*

} **finally** {

*// handle exception code*

}

Description:

* *try block*: When putting one or more statements in a try block, the system may confont an exception. The compiler know an exception may occur, so it uses a special structure to assess all the statements in the block.
* *catch block*: When a problem arises, one solution is to defining the code block. That's what catch block is all about. It is the receiver of the exceptions produced by try blocks.
* *finally block*: You can also define finally blocks, whose codes will never fail to run whatever the outcome of the try block codes is. Finally blocks can run in all common environments and will be executed regardless of whether a try block has finished running, any exception is occurred, or whehter the exceptions have been handled in the catch block.

Rule:

* A try block must be followed by a catch or finally block. A catch and finally block can exist simultaneously after a try block.
* The blocks should be put in certain order. If a catch and finally block are used at the same time, catch blocks must be put after try blocks.
* One try block may have multiple catch block. If this is the case, the first matching block is executed.
* A try block can only have one finally block.

*For example*

**public** **int** **getNumberFromArray**(**int**[] array, **int** index) {

**try** {

**return** array[index];

} **catch** (ArrayIndexOutOfBoundsException e) {

System.out.println("IndexOutOfBoundsException:" + e);

}

}

*Running result:* When the length of index is over that of the array, you will catch the index out of bounds exception.

**throws/throw exceptions**

Declaring an exception are used to throw an exception to its caller under certain methods. The compiler will also be notified: any caller of this method should follow the handling or delcaration rule.

*Grammar:*

**public** returnType **methodName**(paramType param) **throws** ExceptionName {

*// other code*

**throw** **new** Exception...

}

The rule is as follows:

* Throw keywords should be put at the end of method signature;
* Throw keywords can be used to throw an exception, whether it is instantiated or newly caught.
* A method may delcare that multiple exceptions have been thrown, with each exception divided by a comma.

*For example:*

**public** **int** **getNumberFromArray**(**int**[] array, **int** index) **throws** ArrayIndexOutOfBoundsException {

**if** (index > array.length) {

**throw** **new** ArrayIndexOutOfBoundsException("数组越界异常");

}

**return** array[index];

}

Here we only introduce some basic knowledge of exceptions. Considering its importance in Java development, we provide below materials for your extra study:

1. [Exceptions](https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html)
2. [Java exception handling and its application](https://www.ibm.com/developerworks/cn/java/j-lo-exception/index.html)
3. [A brief introduction on Java exception](http://www.cnblogs.com/focusj/archive/2011/12/26/2301524.html)

**Common exceptions in Java**

There are many exception types in Java and each has its own features. Understanding certain exceptions may help a lot in development and testing. Below are some recommended learning material for your perusal.

1. [List of Java Exceptions](https://programming.guide/java/list-of-java-exceptions.html)
2. [Common exception types in Java](https://www.cnblogs.com/cvst/p/5822373.html)
3. [Detailed explanation on common exceptions in Java](https://www.jianshu.com/p/6a6549a6bebb)
4. [A summary of common exceptions in Java](https://www.jianshu.com/p/a03c8807bbbc)