**Checked and Unchecked Exception**

Two types of Exception exist in Java. Though there is no difference in functionality and you can very achieve same thing with either checked Exception or Unchecked Exception, there is some difference on exception handling part. In this Java tutorial we will see **what is checked and Unchecked Exception in Java**, Examples of Checked and Unchecked Exception and most importantly we will learn when to use Checked Exception and when to use Unchecked Exception in Java and lastly we will see **difference between checked and unchecked exception** to understand things better. By the way this article is second in my post on Exception along with d[ifference between throw and throws in Java](http://javarevisited.blogspot.com/2012/02/difference-between-throw-and-throws-in.html) and improved Exception handling in Java 7 with [Automatic resource management ARM](http://javarevisited.blogspot.sg/2011/09/arm-automatic-resource-management-in.html) and multi-catch block in JDK 7.

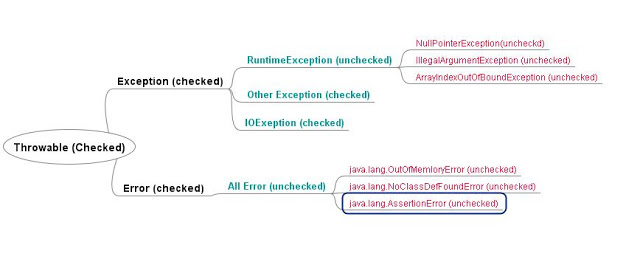
Difference between Checked vs Unchecked Exception in Java

What is Checked Exception in Java?

**Checked Exception in Java** is all those Exception which requires being catches and handled during [compile time](http://javarevisited.blogspot.com/2012/03/what-is-static-and-dynamic-binding-in.html). If Compiler doesn’t see try or catch block handling a Checked Exception, it throws Compilation error. Now Which Exception is checked Exception and Why Checked Exception are introduced in first place? All the Exception which are direct sub Class of Exception but not inherit Runtime Exception are Checked Exception.

While doing File Programming in C++ I found that most of the time programmer forgets to close file descriptors , which often result in locking of file on OS level. Since Java is introduced after C++, designers of Java thought to ensure such mistakes are not allowed and resources opened are closed properly. To ensure this they introduced Checked Exception. If you see most of [File IO related operation](http://javarevisited.blogspot.com/2011/12/read-and-write-text-file-java.html) comes under IOException which is checked one. Though is a special scenario related to Checked Exception but you can generalize this as, where Java sees an opportunity of failure more, they ensure that programmer provide recovery strategy or at least handle those scenario gracefully.

Since a picture is worth 1000 words I have put together Exception hierarchy in mind map which clearly says which Exceptions are checked and which Exceptions are unchecked.

[](http://4.bp.blogspot.com/-ocSXMYac3Eo/T9IGCdsbHQI/AAAAAAAAAYM/EjDyZa9ykik/s1600/checked+exception+vs+unchecked+exception+in+java.JPG)

When to use Checked Exception in Java

Knowing **Checked Exception** is not that useful until you know how to use Checked Exception in Java. Java has often been criticized for its Checked Exception strategy, arguments given are that checked Exception adds lot of boiler plate code and makes whole class or function unreadable. Somewhat I agree with this and java also recognize this by introducing [improved Exception handling mechanism in Java7](http://javarevisited.blogspot.com/2011/07/jdk7-multi-cache-block-example-tutorial.html) but Checked Exception does have its real purpose. Following are some scenarios where I would prefer to use Checked Exception to ensure that [Code is Robust and stable](http://javarevisited.blogspot.com/2011/09/how-to-write-production-quality-code.html):

1) All Operation where chances of failure is more e.g. IO Operation, Database Access or Networking operation can be handled with Checked Exception.

2) When you know what to do (i.e. you have alternative) when an Exception occurs, may be as part of Business Process.

3) **Checked Exception** is a reminder by compiler to programmer to handle failure scenario.

Example of checked Exception in Java APIFollowing Examples of Checked Exception in Java library:

IOException

[SQLException](http://javarevisited.blogspot.com/2012/01/javasqlsqlexception-invalid-column.html)

DataAccessException

[ClassNotFoundException](http://javarevisited.blogspot.com/2011/08/classnotfoundexception-in-java-example.html)

InvocationTargetException

What is Unchecked Exception in Java?

**Unchecked Exception in Java** is those Exceptions whose handling is not verified during Compile time. Unchecked Exceptions mostly arise due to programming errors like accessing method of a null object, accessing element outside an array bonding or invoking method with illegal arguments. In Java, Unchecked Exception is direct sub Class of RuntimeException. What is major benefit of Unchecked Exception is that it doesn't reduce code readability and keeps the client code clean.

When to use UnCheckedException in Java

A good strategy of **Exception handling in Java** is wrapping a checked Exception into **UnCheckedException**. Since most of [Database operation](http://javarevisited.blogspot.com/2011/11/database-transaction-tutorial-example.html) throws SQLException but it’s not good to let SQLException propagate from your DAO layer to up higher on business layer and client code provide exception handling you can handle SQLException in DAO layer and you can wrap the cause in a RuntimeException to propagate through client code. Also as I said earlier unchecked exceptions are mostly programming errors and to catch them is real hard until you do a load test with all possible input and scenario.

**Difference between Checked and Unchecked Exception in Java**

Now we have enough information to differentiate Checked Exception with Unchecked Exception:

1) Checked Exception is required to be handled by compile time while Unchecked Exception doesn't.

2) Checked Exception is direct sub-Class of Exception while Unchecked Exception are of RuntimeException.

3) CheckedException represent scenario with higher failure rate while UnCheckedException are mostly programming mistakes.

Example of unchecked Exception in Java API

Here are few **examples of Unchecked Exception in Java** library:

NullPointerException

ArrayIndexOutOfBound

IllegalArgumentException

IllegalStateException

**Summary:**

1. Both Checked and Unchecked Exception are handled using keyword try, catch and finally.

2. In terms of Functionality Checked and Unchecked Exception are same.

3. Checked Exception handling verified during compile time.

4. Unchecked Exception are mostly programming errors

5. JDK7 provides improved Exception handling code with c[atching multiple Exception in one catch block](http://javarevisited.blogspot.sg/2011/07/jdk7-multi-cache-block-example-tutorial.html) and reduce amount of boiler plate code required for exception handling in Java.

**Related Java Tutorials**

[How to Split String in Java with Example](http://javarevisited.blogspot.com/2011/09/string-split-example-in-java-tutorial.html)

[Static Method and Static Variable in Java](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html)

[How to override HashCode method in Java](http://javarevisited.blogspot.com/2011/10/override-hashcode-in-java-example.html)

[How SubString method works in Java](http://javarevisited.blogspot.com/2011/10/how-substring-in-java-works.html)

[How to avoid deadlock in Java](http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html)

[How to Convert Date to String in Java](http://javarevisited.blogspot.com/2011/09/convert-date-to-string-simpledateformat.html)

You might like:

* [Difference between Method Overloading and Overriding in Java?](http://javarevisited.blogspot.com/2011/12/method-overloading-vs-method-overriding.html)
* [ARM- Automatic resource management in Java7 an example tutorial](http://javarevisited.blogspot.com/2011/09/arm-automatic-resource-management-in.html)
* [Difference between Abstract class vs Interface in Java](http://javarevisited.blogspot.com/2013/05/difference-between-abstract-class-vs-interface-java-when-prefer-over-design-oops.html)
* [fail-safe vs fail-fast Iterator in Java](http://javarevisited.blogspot.com/2012/02/fail-safe-vs-fail-fast-iterator-in-java.html)

|  |  |
| --- | --- |
|  |  |

**5 comments :**

**[Mehmet Cirak](https://www.blogger.com/profile/17375733986162327174" \o "Click to open in a new window" \t "_blank) said...**

Good post. I use the same/similar rule of thumb when it comes to exceptions, and when in doubt go with runtime exception based on arguments presented by the Spring Framework team. Feels good to have someone else think the same way, almost like an affirmation :)

[January 12, 2012 at 11:14 AM](http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html?showComment=1326395685898#c5489550892754189035) [http://www.blogger.com/img/icon_delete13.gif](https://www.blogger.com/delete-comment.g?blogID=8712770457197348465&postID=5489550892754189035)

**Niharika said...**

One difference between Checked and UnChecked Exception is that checked Exception requires mandatory try catch or try finally block but unchecked Exception don't.  
  
Another difference between Checked and UnChecked Exception is in where to use them. checked Exception should be use if you know how to recover from Exception while Unchecked Exception should be used for programming errors.

[March 8, 2012 at 1:10 AM](http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html?showComment=1331197849235#c218639142346860400) [http://www.blogger.com/img/icon_delete13.gif](https://www.blogger.com/delete-comment.g?blogID=8712770457197348465&postID=218639142346860400)

**Gauri said...**

Hello Javin, Can you please provide some guidance on proper use of checked Exception and RuntimeException in Java. Can you give me a scenario, where you chose to throw Exception or sub class of java.lang.Exception instead of unchecked exception or subclass or java.lang.RuntimeException ?

[June 27, 2013 at 12:34 AM](http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html?showComment=1372318491847#c3698283945398944801) [http://www.blogger.com/img/icon_delete13.gif](https://www.blogger.com/delete-comment.g?blogID=8712770457197348465&postID=3698283945398944801)

**Anonymous said...**

There is some discrepancy in the diagram. Error should be unchecked

[June 12, 2014 at 12:39 PM](http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html?showComment=1402601953443#c6516468767366047211) [http://www.blogger.com/img/icon_delete13.gif](https://www.blogger.com/delete-comment.g?blogID=8712770457197348465&postID=6516468767366047211)

**Anonymous said...**

I do not understand why "Since Java is introduced after C++, designers of Java thought to ensure such mistakes are not allowed and resources opened are closed properly." Checked exceptions have nothing in common with clearing resources.

Read more: <http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html#ixzz3mZu64qPa>