Difference between RuntimeException and checked Exception in Java

RuntimeException vs Checked Exception in Java

Java Exceptions are divided into two categories `RuntimeException` also known as unchecked Exception and checked Exception. *Main difference between RuntimeException and checked Exception* is that It is mandatory to provide try-catch or try finally block to handle checked Exception and failure to do so will result in a compile-time error, while in the case of `RuntimeException` this is not mandatory. The difference between checked and unchecked exception is one of the a most popular question on Java interview for 2 to years experienced developer especially related to Exception concepts.

The answer to this question is rather similar as mentioned in previous lines and they are mostly asked along with other Java Exception interview questions like difference between throw and throws an Error vs Exception.

Any Exception which is a subclass of `RuntimeException` are called unchecked and mandatory exception handling is no requirement for them.

Some of the most common Exception like NullPointerException, `ArrayIndexOutOfBoundException` are `unchecked` and they are descended from `java.lang.RuntimeException`. Popular example of checked Exceptions are ClassNotFoundException and `IOException` and that's the reason you need to provide a try catch finally block while performing file operations in Java as many of them throws `IOException`.

Similarly many utilities of Reflection API throws `java.lang.ClassNotFoundException`. In this Java tutorial we will see some more difference between `RuntimeException` and checked Exception in Java.

Runtime Exception vs Checked Exception in Java

Apart from the fundamental difference between Runtime and checked exception, another burning question is while creating custom Exception should you make them unchecked by deriving from `java.lang.RuntimeException` or checked? well, this decision is purely yours though some thoughts are available in the Java community. I mostly see JDK when in doubt and try to follow practices available in JDK.

If a method is likely to fail and the chances of failure are more than 50% it should throw Checked Exception to ensure alternate processing in case it failed. Another thought is that programming errors should be unchecked and derived from `RuntimeException` e.g. `java.lang.NullPointerException`.

Checked Exception also enforces proper handling of the error condition, though it's theoretical in nature and many programs simply appease compiler by providing try catch block instead of correctly handling exceptions in the catch block.

One of the disadvantage of checked exception over runtime exception is that it makes your

code ugly by adding boilerplate code in form of try-catch-finally block. Though this issue is addressed to some extent by improved Exception handling in JDK 7 by introducing automatic resource management or ARM blocks and allowing to catch multiple Exception in same catch block.

That's all on **the difference between runtime exception and checked in Java**. this question can also be asked as checked vs unchecked exception. Unchecked means compiler doesn't check and Checked means compiler checks for exception handling.