What are the most commonly used runtime exceptions in java? [closed]

Asked 15 years, 2 months ago Modified 7 years ago Viewed 42k times

46

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 6 years ago.

Improve this question

As a java programmer who wishes to perfect his programming skills, I often come across the situations that I have to create a runtime exception. I know it's a good practice if one use wisely.

Personally, **NullPointerException** and **IllegalStateException** are the most commonly used in the softwares that I have created. How about you?

What runtime exceptions do you often use? In what situations do you use them?

java exception runtime

Share

Improve this question

Follow

asked Oct 1, 2009 at 9:19

Winston Chen

6,879 ● 12 ● 55 ● 82

6 Answers

Sorted by: Highest score (default)

68

I never throw **NullPointerException**. For me, it is one that appears naturally in the code when something goes wrong and that requires a developer to look at what happens. Then (s)he fixes the cause and it doesn't happen again.

I use **IllegalStateException** to signal that an object is incorrectly configured or that calls are in an incorrect order. However, we all know that ideally, an object should ensure it can't be in a bad state and that you can't call it in incorrect order (make a builder and a resulting object ...).

I use a lot of **IllegalArgumentException** when a method detects that its parameters are incorrect. This is the responsibility of any public method, to stop processing (to avoid indirect errors that are more difficult to understand). Also, a few `if`s in the beginning of a method serve a documentation purpose (documentation that never diverge from the code because it is the code :-)).

```
     public void myMethod(String message, Long id) {
       if (message == null) {
          throw new IllegalArgumentException("myMethod
null");
          // The message doesn't log the argument beca
is null.
       }
       if (id == null) {
          throw new IllegalArgumentException("myMethod
          // This case is separated from the previous
          // 1. to output a precise message
          // 2. to document clearly in the code the re
       }
       if (message.length()<12) {
          throw new IllegalArgumentException("myMethod
was '" + message + "'");
          // here, we need to output the message itsel
          // because it is a useful debug information.
       }
     }
```

I also use **specific Runtime Exceptions** to signal higher level exceptional conditions.

> For example, if a module of my application couldn't start, I might have a *ModuleNotOperationalException* thrown (ideally by a generic code like an interceptor, otherwise by a specific code) when another module calls it. After that architectural decision, each module has to deal with this exception on operations that call other modules...

Share Improve this answer edited Dec 30, 2013 at 13:19

Follow

Steve Chambers
39.3k ● 29 ● 172 ● 220

answered Oct 1, 2009 at 9:26

KLE
24.1k ● 4 ● 57 ● 62

8 Great list; I'd add UnsupportedOperationException, which I use all the time with specific implementations of collection interfaces. – Carl Oct 1, 2009 at 11:30

Great answer. I'll pay more attention to the IllegalStateException in the future! – reef Oct 1, 2009 at 12:09

1 Great answer. I can only +1 once, but I'd do it twice: once for the good explanation, and once for the self-documenting precondition enforcement. I often use assert for those, but illegal argument is an exact fit for the situation. – CPerkins Oct 1, 2009 at 15:10

12

I've always considered that runtime exceptions should represent programming errors (e.g. `null` reference passed in when not expected, array index out of bounds, etc.) while checked exceptions should represent exceptional conditions in the environment that cannot be "coded away" (e.g. `IOException`, `SQLException`).

One violation of this is that sometimes you'll need to wrap what ought to be a checked exception in a RuntimeException, in order to satisfy the definition of an interface. As a brief example, you might have some snazzy implementation of `java.util.List` that manages

a distributed list between multiple machines. Clearly this would throw checked exceptions (probably some subclass of `IOException`) if defined on its own, but the benefits of making this class implement `List` is that clients can use it almost transparently, anywhere they use another list.

This can lead to what Joel terms a leaky abstraction, though, so it's important that your documentation is clear what exceptions can be thrown and what they mean! In this case I find a custom subclass of `RuntimeException` to generally be clearer at communicating the root cause rather than trying to shoehorn it into an existing runtime exception class.

Share Improve this answer

Follow

answered Oct 1, 2009 at 9:44

Andrzej Doyle

104k ● 33 ● 191 ● 231

> You are in line with the original thinking for Java, concerning CheckedExceptions. Actually, this is known as the only original feature of Java (all others were taken from successful other languages). But there has been a lot of debate over this, and I feel the current trend is to consider the distinction as a mistake in Java. The exact problems would be off-topic here, you can google them... – KLE Oct 1, 2009 at 9:50

3

UnknownException, very usefull :P

i also like org.apache.commons.lang.NotImplementedException

Share Improve this answer

Follow

answered Oct 1, 2009 at 12:40

IAdapter

64.6k • 73 • 186 • 243

15 I use `throw new UnsupportedOperationException("Not yet implemented");` for parts of my app that are not yet implemented. – Jesper Oct 1, 2009 at 12:51

3

I use `IllegalArgumentException` relatively often. Most of the time, I will try to return the default value as soon as it is logical but some of the time it was not, and so I use this one.

Another one I use is `ArrayIndexOutOfBoundsException`.

Share Improve this answer

Follow

edited May 20, 2013 at 10:43

Matt Fenwick

49.1k • 24 • 129 • 198

answered Oct 1, 2009 at 9:28

NawaMan

25.7k • 11 • 53 • 77

3

According to **Joshua Bloch** in *Effective Java*,

The most commonly reused exceptions:

1. IllegalArgumentException

2. IllegalStateException
3. NullPointerException
4. IndexOutOfBoundsException
5. ConcurrentModificationException
6. UnsupportedOperationException

Share Improve this answer

Follow

answered Dec 14, 2017 at 4:32

vinS

1,455 ● 5 ● 25 ● 39

2

Mostly I don't throw runtime exception. Rather than after checking specific conditions throw user defined exceptions. But the few I have used are - IllegalAccessException , ArithmeticException, NumberFormatException and SecurityException.

Share Improve this answer

Follow

answered Jul 11, 2013 at 5:25

Shruti Rawat

697 ● 6 ● 11 ● 24