Best Practices for Exception Handling
Pages: 1, **2**

3. Try not to create new custom exceptions if they do not have useful information for client code.

What is wrong with following code?

```
public class DuplicateUsernameException
    extends Exception {}
```

It is not giving any useful information to the client code, other than an indicative exception name. Do not forget that Java `Exception` classes are like other classes, wherein you can add methods that you think the client code will invoke to get more information.

We could add useful methods to `DuplicateUsernameException`, such as:

```
public class DuplicateUsernameException
    extends Exception {
    public DuplicateUsernameException
        (String username){....}
    public String requestedUsername(){...}
    public String[] availableNames(){...}
}
```

The new version provides two useful methods: `requestedUsername()`, which returns the requested name, and `availableNames()`, which returns an array of available usernames similar to the one requested. The client could use these methods to inform that the requested username is not available and that other usernames are available. But if you are not going to add extra information, then just throw a standard exception:

```
throw new Exception("Username already taken");
```

Even better, if you think the client code is not going to take any action other than logging if the username is already taken, throw a unchecked exception:

```
throw new RuntimeException("Username already taken");
```

Alternatively, you can even provide a method that checks if the username is already taken.

It is worth repeating that checked exceptions are to be used in situations where the client API can take some productive action based on the information in the exception. *Prefer unchecked exceptions for all programmatic errors*. They make your code more readable.

4. Document exceptions.

You can use Javadoc's `@throws` tag to document both checked and unchecked exceptions that your API throws. However, I prefer to write unit tests to document exceptions. Tests allow me to see the exceptions in action and hence serve as documentation that can be executed. Whatever you do, have some way by which the client code can learn of the exceptions that your API throws. Here is a sample unit test that tests for `IndexOutOfBoundsException`:

```
public void testIndexOutOfBoundsException() {
    ArrayList blankList = new ArrayList();
    try {
        blankList.get(10);
        fail("Should raise an IndexOutOfBoundsException");
    } catch (IndexOutOfBoundsException success) {}
}
```

The code above should throw an `IndexOutOfBoundsException` when `blankList.get(10)` is invoked. If it does not, the `fail("Should raise an IndexOutOfBoundsException")` statement explicitly fails the test. By writing unit tests for exceptions, you not only document how the exceptions work, but also make your code robust by testing for exceptional scenarios.

Best Practices for Using Exceptions

The next set of best practices show how the client code should deal with an API that throws checked exceptions.

1. Always clean up after yourself

If you are using resources like database connections or network connections, make sure you clean them up. If the API you are invoking uses only unchecked exceptions, you should still clean up resources after use, with `try - finally` blocks.

```
public void dataAccessCode(){
    Connection conn = null;
    try{
        conn = getConnection();
        ..some code that throws SQLException
    }catch(SQLException ex){
        ex.printStacktrace();
    } finally{
        DBUtil.closeConnection(conn);
    }
}

class DBUtil{
    public static void closeConnection
        (Connection conn){
```

```
        try{
            conn.close();
        } catch(SQLException ex){
            logger.error("Cannot close connection");
            throw new RuntimeException(ex);
        }
    }
}
```

`DBUtil` is a utility class that closes the `Connection`. The important point is the use of `finally` block, which executes whether or not an exception is caught. In this example, the `finally` closes the connection and throws a `RuntimeException` if there is problem with closing the connection.

2. Never use exceptions for flow control

Generating stack traces is expensive and the value of a stack trace is in debugging. In a flow-control situation, the stack trace would be ignored, since the client just wants to know how to proceed.

In the code below, a custom exception, `MaximumCountReachedException`, is used to control the flow.

```
public void useExceptionsForFlowControl() {
    try {
        while (true) {
            increaseCount();
        }
    } catch (MaximumCountReachedException ex) {
    }
    //Continue execution
}

public void increaseCount()
    throws MaximumCountReachedException {
    if (count >= 5000)
        throw new MaximumCountReachedException();
}
```

The `useExceptionsForFlowControl()` uses an infinite loop to increase the count until the exception is thrown. This not only makes the code difficult to read, but also makes it slower. Use exception handling only in exceptional situations.

3. Do not suppress or ignore exceptions

When a method from an API throws a checked exception, it is trying to tell you that you should take some counter action. If the checked exception does not make sense to you, do not hesitate to convert it into an unchecked exception and throw it again, but do not ignore it by catching it with `{}` and then continue as if nothing had happened.

4. Do not catch top-level exceptions

Unchecked exceptions inherit from the `RuntimeException` class, which in turn inherits from `Exception`. By catching the `Exception` class, you are also catching `RuntimeException` as in the following code:

```
try{
..
}catch(Exception ex){
}
```

The code above ignores unchecked exceptions, as well.

5. Log exceptions just once

Logging the same exception stack trace more than once can confuse the programmer examining the stack trace about the original source of exception. So just log it once.

Summary

These are some suggestions for exception-handling best practices. I have no intention of staring a religious war on checked exceptions vs. unchecked exceptions. You will have to customize the design and usage according to your requirements. I am confident that over time, we will find better ways to code with exceptions.

I would like to thank Bruce Eckel, Joshua Kerievsky, and Somik Raha for their support in writing this article.

Related Resources

- "Does Java need Checked Exceptions?" by Bruce Eckel
- "Exceptional Java," by Alan Griffiths
- "The trouble with checked exceptions: A conversation with Anders Hejlsberg, Part II" on Artima.com
- "Checked exceptions are of dubious value," on C2.com
- Conversation with James Gosling by Bill Venners

Gunjan Doshi works with agile methodologies and its practices and is a Sun certified Java programmer.

Return to ONJava.com.

Comments on this article

1 to 37 of 37

Runtime Exception is not informative
2007-03-07 19:48:45 johnydep [View]

Runtime Exception is not informative
2007-08-16 22:11:24 remotec [View]

Runtime Exception is not informative
2007-03-06 22:31:05 johnydep [View]

Correction of the try/finally code
2006-11-16 15:34:08 rillig [View]

Correction of the try/finally code
2008-01-21 01:00:38 venkataramanam [View]

Comment on Exception Handling Article
2006-11-15 06:13:05 JayaNagdev [View]

Good article
2006-07-04 04:48:40 vjavatech [View]

Just crap, read http://today.java.net/pub/a/today/2006/04/06/exception-handling-antipatterns.html instead
2006-06-22 01:20:18 pgrange [View]

Just crap, read http://today.java.net/pub/a/today/2006/04/06/exception-handling-antipatterns.html instead
2008-07-31 00:10:57 kdgar [View]

Thank you , and suggestion
2005-11-28 10:49:00 ionia23 [View]

very useful
2005-07-11 05:09:01 JavaRambab [View]

very useful
2005-07-11 03:54:47 JavaRambab [View]

Effective Exception Handling and Logging
2005-07-08 07:51:27 celo [View]

Trackback from http://blog.csdn.net/jwbecalm/archive/2005/06/16/395512.aspx
Exception å¤„ç†ä¹‹æœŒä½³å®žè·µ
2005-06-15 19:46:47 [View]

Trackback from http://jroller.com/page/vivekv/20040621#h3_best_practices_for_exception
Best Practices for Exception handling in Java
2004-06-22 00:30:42 [View]

Trackback from http://www.realityinteractive.com/rgrzywinski/archives/000028.html
Coding Defensively
2004-04-27 08:08:12 [View]

Trackback from http://baby.homeip.net/patrick/archives/000154.php
Bruce Eckel's Weblog
2004-02-08 02:43:07 [View]

Modification proposed to java exception handling
2003-12-05 14:26:20 anonymous2 [View]

A few things
2003-12-05 08:17:50 javid [View]

Custom exceptions always have information for client code!
2003-12-23 03:00:26 sebastien.couturiaux [View]

Exceptions in unit tests
2003-12-09 10:14:00 zipwow [View]

Trackback from http://www.redwolf.pe.kr/myweblog/archives/000263.html
Best Practices for Exception Handling
2003-11-30 17:57:51 [View]

Thanks for a most excellent article!
2003-11-28 12:49:09 dashmore [View]

Disagree "new custom exceptions if no usefull info"
2003-11-27 08:04:36 anonymous2 [View]

Don't wrap exceptions!
2003-11-26 07:54:51 fjalvingh [View]

Don't wrap exceptions!
2003-11-27 05:39:31 anonymous2 [View]

Don't wrap exceptions!
2003-12-01 08:14:43 anonymous2 [View]

Don't wrap exceptions!
2003-12-03 13:58:50 anonymous2 [View]

Don't wrap exceptions!
2008-07-31 00:20:09 kdgar [View]

I suppose I'm in the minority
2003-11-25 20:00:50 ljnelson [View]

I suppose I'm in the minority
2003-11-27 05:43:30 ipreuss [View]

I suppose I'm in the minority
2003-11-27 05:44:25 ipreuss [View]

Ultimately, what to do with the rascals
2003-11-25 16:50:19 xyphrnld0x [View]

"Alternatively, you can even provide a method that checks if the username is already taken." not working
2003-11-24 00:31:32 anonymous2 [View]

"Alternatively, you can even provide a method that checks if the username is already taken." not working
2003-12-03 14:07:00 anonymous2 [View]

declaring impossible exceptions
2003-11-21 09:19:42 anonymous2 [View]

declaring impossible exceptions
2003-11-21 16:58:56 anonymous2 [View]

Exception messages best practices
2003-11-21 04:22:14 anonymous2 [View]

Trackback from http://www.magpiebrain.com/archives/000138.html
Proper Exception Handling
2003-11-21 02:41:13 [View]

Nested Exceptions in distributed systems
2003-11-20 18:25:55 anonymous2 [View]

Logging APIs use Exceptions to get info
2003-11-20 17:13:42 anonymous2 [View]

Best Practices
2003-11-20 12:42:24 anonymous2 [View]

Good article
2003-11-20 12:13:48 anonymous2 [View]

Worst Practices
2003-11-20 11:53:11 anonymous2 [View]

Worst Practices
2003-11-20 12:19:26 anonymous2 [View]

Very nice article.
2003-11-20 08:18:58 joshy1 [View]

Very nice article.
2003-11-20 09:54:23 gunjandoshi1 [View]

I18N and Exceptions?
2003-11-20 03:55:15 bazzargh [View]

I18N and Exceptions?
2004-03-31 11:18:56 vsonnathi [View]

Trackback from http://davew.typepad.com/42/2003/11/onjavacom_best_.html
ONJava.com: Best Practices for Exception Handling [Nov. 19, 2003]
2003-11-20 02:04:11 [View]

throw new Exception("...")
2003-11-19 22:48:53 anonymous2 [View]

throw new Exception("...")
2003-11-20 09:46:28 gunjandoshi1 [View]

throw new Exception("...")
2003-11-20 01:59:31 anonymous2 [View]

J2EE and RuntimeException
2003-11-19 22:30:54 schaefera [View]

J2EE and RuntimeException
2003-11-20 07:54:41 anonymous2 [View]

Good summary
2003-11-19 18:58:01 anonymous2 [View]

1 to 37 of 37