Java Catch Multiple Exceptions, Rethrow Exception

APRIL 2, 2018 BY PANKAJ — 11 COMMENTS

In Java 7, catch block has been improved to handle multiple exceptions in a single catch block. If you are catching multiple exceptions and they have similar code, then using this feature will reduce code duplication. Let's understand java catch multiple exceptions feature with an example.

Java catch multiple exceptions

Before Java 7, we used to catch multiple exceptions one by one as shown below.

```
catch (IOException ex) {
     logger.error(ex);
     throw new MyException(ex.getMessage());
catch (SQLException ex) {
     logger.error(ex);
     throw new MyException(ex.getMessage());
}
```

In Java 7, we can catch both these exceptions in a single catch block as:

```
catch(IOException | SQLException ex){
     logger.error(ex);
     throw new MyException(ex.getMessage());
}
```

If a catch block handles multiple exception, you can separate them using a pipe (|) and in this case exception parameter (ex) is final, so you can't change it. The byte code generated by this feature is smaller and reduce code redundancy.

Java rethrow exception

Another improvement is done in Compiler analysis of rethrown exceptions. Java rethrow exception allows you to specify more specific exception types in the throws clause of a method declaration.

Let's see this with a small example:

```
package com.journaldev.util;

public class Java7MultipleExceptions {

        public static void main(String[] args) {
                try{
                        rethrow("abc");
                }catch(FirstException | SecondException | ThirdException e){
                        //below assignment will throw compile time exception since e
is final
                        //e = new Exception();
                        System.out.println(e.getMessage());
                }
        }

        static void rethrow(String s) throws FirstException, SecondException,
                        ThirdException {
                try {
                        if (s.equals("First"))
                                throw new FirstException("First");
                        else if (s.equals("Second"))
                                throw new SecondException("Second");
```

As you can see that in `rethrow` method, catch block is catching Exception but it's not part of throws clause. Java 7 compiler analyze the complete try block to check what types of exceptions are thrown and then rethrown from the catch block.

Note that this analysis is disabled if you change the catch block argument.

Further Reading: Exception Handling in Java.

« PREVIOUS
Java Resize Image

NEXT »
Regular Expression in Java – Java Regex Example

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: JAVA

Comments

Dileep says

JUNE 18, 2018 AT 11:54 PM

what is the use again we have to re catch exceptions from where we called the method

Reply

Tanu says

JANUARY 11, 2016 AT 3:01 PM

Hi Pankaj,
Note to add here if there is hierarchy of exceptions in catch block then we need to catch base exception only, can't write child and parent exception in same catch with pipe, it will give compilation error.
e.g.
catch (IOException | Exception e){}
will say IOException already caught by the alternative Exception.

Reply

phintsho says

APRIL 22, 2014 AT 12:32 AM

hellow pankaj ,
can u please explain more abt overloading and overriding… because i always use to get confuse between these two

Reply

phintsho says

APRIL 22, 2014 AT 12:28 AM

i dont understand the throws exception so please pankaj can u explain …..

Reply

indra says

APRIL 5, 2014 AT 6:40 AM

Hi pankaj,
i really dont understand the code for rethrowing an exception in java7.could you please explain the code in detail.

Reply

Ashok says

JANUARY 7, 2014 AT 4:06 AM

Hi Pankaj,
if i want to throw different exception message based on exception catched then how to do that?
ex: if sql exception then i want send message as failed to insert into table
and number excpetion then input provided is not a valid number
then how to use single catch block?

Reply

Pankaj says

JANUARY 7, 2014 AT 6:21 AM

If you have different logic for different catch blocks, then it doesn't make much sense to have them in a single catch block.

Reply

Vivek says

JANUARY 29, 2014 AT 6:45 PM

Hi
You can retrieve the exception object inside catch block and based upon the type of that exception you can print the exception message.

Reply

subrat panda says

JULY 23, 2013 AT 12:07 PM

Hi Pankaj,
Using single catch block how can I differentiate from where my code is getting exception?

```
try{}
catch(IOException ie){
ie.printStackTrace();}
catch(SQLException se){
se.printStackTrace();}
catch(Exception e){
e.printStackTrace();}
```

now it is replaced with

```
catch(IOException| SQLException |Exception e){
e.printStackTrace();}
```

In the above code my doubt is e.printStackTrace() will print which exception class message whether (IOException / SQLException/Exception)?

Reply

Pankaj says

JULY 24, 2013 AT 3:38 AM

Exception stack trace provide very detailed message for the type of exception, methods invocation hierarchy and even the line number that caused the exception.

Reply

Ram says

JUNE 26, 2018 AT 11:45 AM

We can get the class name by writing the statement below inside the catch block.
ex.getClass().getCannonicalName();//this gives us the exact class name of the exception. It's short to identify the exact exception class type.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

□

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

CORE JAVA TUTORIAL

Java 10 Tutorials Java 9 Tutorials Java 8 Tutorials Java 7 Tutorials Core Java Basics OOPS Concepts Data Types and Operators String Manipulation Java Arrays Annotation and Enum Java Collections Java IO Operations Java Exception Handling MultiThreading and Concurrency Regular Expressions Advanced Java Concepts

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial