捕获多个类型的异常并重新抛出改进的类型检查异常

Catching Multiple Exception Types and Rethrowing Exceptions with Improved Type Checking

This page covers the following topics:

- Handling More Than One Type of Exception
- Rethrowing Exceptions with More Inclusive Type Checking

处理多个异常的类型
重新抛出更具包容性的类型检查的异常

Handling More Than One Type of Exception

单个 catch 块可以处理多个类型的异常

In Java SE 7 and later, a single `catch` block can handle more than one type of exception. This feature can reduce code duplication and lessen the temptation to catch an overly broad exception.

减少代码重复、减少捕获一个过于宽泛的异常的诱惑

Consider the following example, which contains duplicate code in each of the `catch` blocks:

```
catch (IOException ex) {
     logger.log(ex);
     throw ex;
catch (SQLException ex) {
     logger.log(ex);
     throw ex;
}
```

In releases prior to Java SE 7, it is difficult to create a common method to eliminate the duplicated code because the variable `ex` has different types.

The following example, which is valid in Java SE 7 and later, eliminates the duplicated code:

```
catch (IOException|SQLException ex) {
    logger.log(ex);
    throw ex;
}
```

catch 子句指定代码块可以处理的异常的类型

The `catch` clause specifies the types of exceptions that the block can handle, and each exception type is separated with a vertical bar (`|`).

Note: If a `catch` block handles more than one exception type, then the `catch` parameter is implicitly `final`. In this example, the `catch` parameter `ex` is `final` and therefore you cannot assign any values to it within the `catch` block.

Bytecode generated by compiling a `catch` block that handles multiple exception types will be smaller (and thus superior) than compiling many `catch` blocks that handle only one exception type each. A `catch` block that handles multiple exception types creates no duplication in the bytecode generated by the compiler; the bytecode has no replication of exception handlers.

创建非重复的字节码，字节码没有异常处理程序的复制

Rethrowing Exceptions with More Inclusive Type Checking

The Java SE 7 compiler performs more precise analysis of rethrown exceptions than earlier releases of Java SE. This enables you to specify more specific exception types in the `throws` clause of a method declaration.

Consider the following example:

```
  static class FirstException extends Exception { }
  static class SecondException extends Exception { }

  public void rethrowException(String exceptionName) throws Exception {
    try {
      if (exceptionName.equals("First")) {
        throw new FirstException();
      } else {
        throw new SecondException();
      }
    } catch (Exception e) {
      throw e;
    }
  }
```

This examples's `try` block could throw either `FirstException` or `SecondException`. Suppose you want to specify these exception types in the `throws` clause of the `rethrowException` method declaration. In releases prior to Java SE 7, you cannot do so. Because the exception parameter of the `catch` clause, `e`, is type `Exception`, and the catch block rethrows the exception parameter `e`, you can only specify the exception type `Exception` in the `throws` clause of the `rethrowException` method declaration.

However, in Java SE 7, you can specify the exception types `FirstException` and `SecondException` in the `throws` clause in the `rethrowException` method declaration. The Java SE 7 compiler can determine that the exception thrown by the statement `throw e` must have come from the `try` block, and the only exceptions thrown by the `try` block can be `FirstException` and `SecondException`. Even though the exception parameter of the `catch` clause, `e`, is type `Exception`, the compiler can determine that it is an instance of either `FirstException` or `SecondException`:

```
  public void rethrowException(String exceptionName)
  throws FirstException, SecondException {
    try {
      // ...
    }
    catch (Exception e) {
      throw e;
    }
  }
```

This analysis is disabled if the `catch` parameter is assigned to another value in the `catch` block. However, if the catch parameter is assigned to another value, you must specify the exception type `Exception` in the `throws` clause of the method declaration.

In detail, in Java SE 7 and later, when you declare one or more exception types in a `catch` clause, and rethrow the exception handled by this `catch` block, the compiler verifies that the type of the rethrown exception meets the following conditions:

- The `try` block is able to throw it.
- There are no other preceding `catch` blocks that can handle it.
- It is a subtype or supertype of one of the `catch` clause's exception parameters.

The Java SE 7 compiler allows you to specify the exception types `FirstException` and `SecondException` in the `throws` clause in the `rethrowException` method declaration because you can rethrow an exception that is a supertype of any of the types declared in the `throws`.

In releases prior to Java SE 7, you cannot throw an exception that is a supertype of one of the `catch` clause's exception parameters. A compiler from a release prior to Java SE 7 generates the error, "unreported exception `Exception`; must be caught or declared to be thrown" at the statement `throw e`. The compiler checks if the type of the exception thrown is assignable to any of the types declared in the `throws` clause of the `rethrowException` method declaration. However, the type of the catch parameter `e` is `Exception`, which is a supertype, not a subtype, of `FirstException` and`SecondException`.