How to create a custom exception type in Java? [duplicate]

Possible Duplicate:
How can I write an Exception by myself?

I would like to create a custom exception in Java, how do I do it?

```
...

try{

...

String word=reader.readLine();

if(word.contains(" "))
  /*create custom exeception*/

}
catch(){
```

when create my custom exception with `throw new...` I obtain the error `unreported exception...must be caught or declared to be thrown`

java exception

edited Feb 13 '13 at 22:48
Eric Leschinski
24.7k 14 129 132

asked Dec 7 '11 at 22:44
Mazzy
1,951 12 32 65

marked as duplicate by pst, Ash Burlaczenko, Bhesh Gurung, Toon Krijthe, C. A. McCann Dec 9 '11 at 16:00

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

1 Can some of the answers *please* discuss checked vs. unchecked exceptions? What does extending Exception imply? RuntimeException? Throwable? – user166390 Dec 7 '11 at 22:50

stackoverflow.com/questions/1070590/… , stackoverflow.com/questions/3505111/… , stackoverflow.com/questions/1176130/… – user166390 Dec 7 '11 at 22:52

There is 1) creating a custom exception type/class (as shown so many times) and 2) raising the exception. To raise an exception, simply pass the appropriate instance to `throw` , normally: `throw new MyFormatExpcetion("spaces are not allowed");` -- you could even use the *standard* ParseException, without "creating" a custom exception type. – user166390 Dec 7 '11 at 22:55

@pst Good point. I've added some information on checked vs unchecked exceptions in my answer. – Laf Dec 7 '11 at 23:31

See video tutorials about exception class creation and handling in Java. bitspedia.com/2013/11/exception-handling-in-java-video.html – Asif Shahzad Nov 18 '13 at 18:36

9 Answers

You should be able to create a custom exception class that extends the `Exception` class, for example:

```
class WordContainsException extends Exception
{
      //Parameterless Constructor
      public WordContainsException() {}

      //Constructor that accepts a message
      public WordContainsException(String message)
      {
         super(message);
      }
 }
```

Usage:

```
try
{
    if(word.contains(" "))
    {
         throw new WordContainsException();
    }
}
catch(WordContainsException ex)
{
     //Process message however you would like
}
```

edited Dec 7 '11 at 23:15

answered Dec 7 '11 at 22:46
Rion Williams
25.5k 8 88 162

2 Without adding any `catch` ? – Mazzy Dec 7 '11 at 22:49

This was just the declaration for the exception, it can be thrown with a simple throw new WordContainsException(); Updated and added usage. – Rion Williams Dec 7 '11 at 22:52

What should contain `catch()` ? – Mazzy Dec 7 '11 at 23:07

You seem to already have the catch within your code, I'm not sure exactly where you want the exception to be thrown. You could put it before the catch section, or inside of it. I edited it assuming how you might want to use the exception – Rion Williams Dec 7 '11 at 23:13

You should add two more constructors (`WordContainsException(Throwable)` and `WordContainsException(String, Throwable)`) to properly support exceptions chaining – Danilo Piazzalunga Sep 18 '13 at 13:11

You need to create a class that extends from `Exception` . It should look like this:

```
public class MyOwnException extends Exception {
    public MyOwnException () {

    }

    public MyOwnException (String message) {
        super (message);
    }

    public MyOwnException (Throwable cause) {
        super (cause);
    }

    public MyOwnException (String message, Throwable cause) {
        super (message, cause);
    }
}
```

Your question does not specify if this new exception should be checked or unchecked.

As you can see here, the two types are different:

- Checked exceptions are meant to flag a problematic situation that should be handled by the developer who calls your method. It should be possible to recover from such an exception. A good example of this is a FileNotFoundException. Those exceptions are subclasses of Exception.

- Unchecked exceptions are meant to represent a bug in your code, an unexpected situation that you might not be able to recover from. A NullPointerException is a classical example. Those exceptions are subclasses of RuntimeException

Checked exception **must** be handled by the calling method, either by catching it and acting accordingly, or by throwing it to the calling method. Unchecked exceptions are not meant to be caught, even though it is possible to do so.

edited Sep 3 '14 at 7:47
Daniel Kutik
4,637 2 12 28

answered Dec 7 '11 at 22:48
Laf
3,516 12 34

Why does it have to look like that? (1) If I am creating my own proprietary exception, I will make it package-private (or private if it is not top-level). (2) Same thing wrt to the constructors. (3) I won't define 4 constructors unless I need them. – emory Dec 7 '11 at 23:17

(1) You should make your new exception a public one, so that methods calling your method may catch this new exception and do something with it (if you mean this new exception to be a checked exception, see my edit. (2) Make the constructors public, no need to hide them. (3) You can get rid of the constructors you

don't need, all 4 are not mandatory. – Laf Dec 7 '11 at 23:30

Great answers about creating custom exception classes. If you intend to reuse the exception in question then I would follow their answers/advice. However, If you only need a quick exception thrown with a message then you can use the base exception class on the spot

```
String word=reader.readLine();

if(word.contains(" "))
  /*create custom exeception*/
  throw new Exception("My one time exception with some message!");
}
```

answered Dec 7 '11 at 22:49

Feisty Mango
7,528 3 28 56

Since you can just create and throw exceptions it could be as easy as

```
if ( word . contains ( " " ) )
{
     throw new RuntimeException ( "Word contains one or more spaces" ) ;
}
```

If you would like to be more formal, you can create an Exception class

```
class SpaceyWordException extends RuntimeException
{

}
```

Either way, if you use `RuntimeException`, your new `Exception` will be unchecked.

edited Apr 1 at 14:28
Connor Wright
35 11

answered Dec 7 '11 at 22:47
emory
6,878 1 17 29

As a careful programmer will often throw an exception for a special occurrence, it worth mentioning some general purpose exceptions like **IllegalArgumentException** and **IllegalStateException** and **UnsupportedOperationException**. IllegalArgumentException is my favorite: throw new IllegalArgumentException("Word contains blank: " + word);

answered Dec 7 '11 at 22:51

Joop Eggen
38.9k 2 21 47

An exception is a class like any other class, except that it extends from `Exception`. So if you create your own class

```
public class MyCustomException extends Exception
```

you can throw such an instance with

```
    throw new MyCustomException( ... );
    //using whatever constructor params you decide to use
```

And this might be an interesting read

answered Dec 7 '11 at 22:47

Robin
26.7k 3 23 52

You just need to create a class which extends Exception (for a checked exception) or any subclass of Exception, or RuntimeException (for a runtime exception) or any subclass of RuntimeException.

Then, in your code, just use

```
if (word.contains(" "))
    throw new MyException("some message");
}
```

Read the Java tutorial. This is basic stuff that every Java developer should know:
http://docs.oracle.com/javase/tutorial/essential/exceptions/

answered Dec 7 '11 at 22:48
JB Nizet
310k 20 291 471

You have to define your exception elsewhere as a new class

```
public class YourCustomException extends Exception{

//Required inherited methods here
}
```

Then you can throw and catch YourCustomException as much as you'd like.

answered Dec 7 '11 at 22:47
Thomas
3,554 7 12

You can create you own exception by inheriting from `java.lang.Exception` Here is an example that can help you do that.

answered Dec 7 '11 at 22:49
GETah
11.8k 2 24 58