ExceptionHandling with MethodOverriding in Java

|  |
| --- |
| There are many rules if we talk about methodoverriding with exception handling. The Rules are as follows:   * **If the superclass method does not declare an exception**   + If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but it can declare unchecked exception. * **If the superclass method declares an exception**   + If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception. |

If the superclass method does not declare an exception

#### 1) Rule: If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception.

1. **import** java.io.\*;
2. **class** Parent{
3. **void** msg(){System.out.println("parent");}
4. }
6. **class** TestExceptionChild **extends** Parent{
7. **void** msg()**throws** IOException{
8. System.out.println("TestExceptionChild");
9. }
10. **public** **static** **void** main(String args[]){
11. Parent p=**new** TestExceptionChild();
12. p.msg();
13. }
14. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionChild)

Output:Compile Time Error

#### 2) Rule: If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but can declare unchecked exception.

1. **import** java.io.\*;
2. **class** Parent{
3. **void** msg(){System.out.println("parent");}
4. }
6. **class** TestExceptionChild1 **extends** Parent{
7. **void** msg()**throws** ArithmeticException{
8. System.out.println("child");
9. }
10. **public** **static** **void** main(String args[]){
11. Parent p=**new** TestExceptionChild1();
12. p.msg();
13. }
14. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionChild1)

Output:child

### If the superclass method declares an exception

#### 1) Rule: If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

### Example in case subclass overridden method declares parent exception

1. **import** java.io.\*;
2. **class** Parent{
3. **void** msg()**throws** ArithmeticException{System.out.println("parent");}
4. }
6. **class** TestExceptionChild2 **extends** Parent{
7. **void** msg()**throws** Exception{System.out.println("child");}
9. **public** **static** **void** main(String args[]){
10. Parent p=**new** TestExceptionChild2();
11. **try**{
12. p.msg();
13. }**catch**(Exception e){}
14. }
15. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionChild2)

Output:Compile Time Error

### Example in case subclass overridden method declares same exception

1. **import** java.io.\*;
2. **class** Parent{
3. **void** msg()**throws** Exception{System.out.println("parent");}
4. }
6. **class** TestExceptionChild3 **extends** Parent{
7. **void** msg()**throws** Exception{System.out.println("child");}
9. **public** **static** **void** main(String args[]){
10. Parent p=**new** TestExceptionChild3();
11. **try**{
12. p.msg();
13. }**catch**(Exception e){}
14. }
15. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionChild3)

Output:child

### Example in case subclass overridden method declares subclass exception

1. **import** java.io.\*;
2. **class** Parent{
3. **void** msg()**throws** Exception{System.out.println("parent");}
4. }
6. **class** TestExceptionChild4 **extends** Parent{
7. **void** msg()**throws** ArithmeticException{System.out.println("child");}
9. **public** **static** **void** main(String args[]){
10. Parent p=**new** TestExceptionChild4();
11. **try**{
12. p.msg();
13. }**catch**(Exception e){}
14. }
15. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionChild4)

Output:child

### Example in case subclass overridden method declares no exception

1. **import** java.io.\*;
2. **class** Parent{
3. **void** msg()**throws** Exception{System.out.println("parent");}
4. }
6. **class** TestExceptionChild5 **extends** Parent{
7. **void** msg(){System.out.println("child");}
9. **public** **static** **void** main(String args[]){
10. Parent p=**new** TestExceptionChild5();
11. **try**{
12. p.msg();
13. }**catch**(Exception e){}
14. }
15. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestExceptionChild5)

Output:child

# Exception handling in Method overriding with example

BY CHAITANYA SINGH | FILED UNDER: [**OOPS CONCEPT**](http://beginnersbook.com/category/oops-concept/)

In the last post we discussed about [**method overriding**](http://beginnersbook.com/2014/01/method-overriding-in-java-with-example/). In this post we will see how to do exception handling for overriding and overridden methods.

**Rule**:  An overriding method (the method of child class) can throw any [**unchecked exceptions**](http://beginnersbook.com/2013/04/java-checked-unchecked-exceptions-with-examples/), regardless of whether the overridden method (method of base class) throws exceptions or not. However the overriding method should not throw [**checked exceptions**](http://beginnersbook.com/2013/04/java-checked-unchecked-exceptions-with-examples/) that are new or broader than the ones declared by the overridden method. The overriding method can throw those checked exceptions, which have less scope than the exception(s) declared in the overridden method.

Let’s understand the above explanation with the help of few examples:

**Example 1: If base class doesn’t throw any exception but child class throws an unchecked exception.**  
In this example class Room is overriding the method color(). The overridden method is not throwing any exception however the overriding method is throwing an unchecked exception (NullPointerException). Upon compilation code ran successfully.

class Building {

void color()

{

System.out.println("Blue");

}

}

class Room extends Building{

//It throws an unchecked exception

void color() throws NullPointerException

{

System.out.println("White");

}

public static void main(String args[]){

Building obj = new Room();

obj.color();

}

}

Output:

White

**Example 2: If base class doesn’t throw any exception but child class throws an checked exception**

import java.io.\*;

class Building {

void color()

{

System.out.println("Blue");

}

}

class Room extends Building{

void color() throws IOException

{

System.out.println("White");

}

public static void main(String args[]){

Building obj = new Room();

try{

obj.color();

}catch(Exception e){

System.out.println(e);

}

}

}

Output:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

Exception IOException is not compatible with throws clause in Building.color()

The above code is having a compilation error: Because the overriding method (child class method) cannot throw a checked exception if the overridden method(method of base class) is not throwing an exception.

**Example 3: When base class and child class both throws a checked exception**

import java.io.\*;

class Building {

void color() throws IOException

{

System.out.println("Blue");

}

}

class Room extends Building{

void color() throws IOException

{

System.out.println("White");

}

public static void main(String args[]){

Building obj = new Room();

try{

obj.color();

}catch(Exception e){

System.out.println(e);

}

}

}

Output:

White

The code ran fine because color() method of child class is **NOT** throwing a checked exception with scope broader than the exception declared by color() method of base class.

**Example 4: When child class method is throwing border checked exception compared to the same method of base class**

package beginnersbook.com;

import java.io.\*;

class Building {

void color() throws IOException

{

System.out.println("Blue");

}

}

class Room extends Building{

void color() throws Exception

{

System.out.println("White");

}

public static void main(String args[]){

Building obj = new Room();

try{

obj.color();

}catch(Exception e){

System.out.println(e);

}

}

}