**Java Finally block – Exception handling**

**What is Finally Block**

1. A [**finally statement**](http://beginnersbook.com/2013/05/java-finally-return/) must be associated with a [**try statement**](http://beginnersbook.com/2013/04/try-catch-in-java/). It identifies a block of statements that needs to be executed regardless of whether or not an [**exception occurs**](http://beginnersbook.com/2013/04/java-exception-handling/) within the try block.

2. After all other try-catch processing is complete, the [**code inside the finally block executes**](http://beginnersbook.com/2013/05/flow-in-try-catch-finally/). It is not mandatory to include a finally block at all, but if you do, it will run regardless of whether an exception was thrown and handled by the try and catch parts of the block.

3. In normal execution the finally block is executed after try block. When any exception occurs first the catch block is executed and then finally block is executed.

4. An exception in the finally block, exactly [**behaves like any other exception**](http://beginnersbook.com/2013/04/user-defined-exception-in-java/).

5. The code present in the **finally block** executes even if the try or catch block contains control transfer statements like [**return**](http://beginnersbook.com/2013/05/java-finally-return/), break or continue.

To understand above concepts better refer the below [**examples**](http://beginnersbook.com/2013/04/exception-handling-examples/).

**Syntax of Finally block**

try

{

//statements that may cause an exception

}

finally

{

//statements to be executed

}

**Cases when the finally block doesn’t execute**

The circumstances that prevent execution of the code in a finally block are:  
– The death of a Thread  
– Using of the System. exit() method.  
– Due to an exception arising in the finally block.

**Finally block and Return statement**

Finally block executes even if there is a return statement in try-catch block. PFB the example –

class JavaFinally

{

public static void main(String args[])

{

System.out.println(JavaFinally.myMethod());

}

public static int myMethod()

{

try {

return 112;

}

finally {

System.out.println("This is Finally block");

System.out.println("Finally block ran even after return statement");

}

}

}

**Output of above program:**

This is Finally block

Finally block ran even after return statement

112

**Finally and Close()**

**Close()** is generally used to close all the open streams in one go. Its a good practice to use close() inside finally block. Since finally block executes even if exception occurs so you can be sure that all input and output streams are closed properly regardless of whether the exception occurs or not.

E.g.

....

try{

OutputStream osf = new FileOutputStream( "filename" );

OutputStream osb = new BufferedOutputStream(opf);

ObjectOutput op = new ObjectOutputStream(osb);

try{

output.writeObject(writableObject);

}

finally{

op.close();

}

}

catch(IOException e1){

System.out.println(e1);

}

...

**Finally block without catch**

A try-finally block is possible without catch block. Which means a try block can be used with finally without having a catch block.

...

InputStream input = null;

try {

input = new FileInputStream("inputfile.txt");

}

finally {

if (input != null) {

try {

in.close();

}catch (IOException exp) {

System.out.println(exp);

}

}

}

...

**Finally block and System.exit()**

**System.exit()** statement behaves differently than**return statement**. Unlike return statement whenever System.exit() gets called in try block then **Finally block** doesn’t get executed. Refer the below example to understand it better –

....

try {

//try block

System.out.println("Inside try block");

System.exit(0)

}

catch (Exception exp) {

System.out.println(exp);

}

finally {

System.out.println("Java finally block");

}

....

In the above example if the**System.exit(0)** gets called without any exception then finally won’t execute. However if any exception occurs while calling **System.exit(0)** then finally block will be executed.

**Handling try-catch-finally block**

* Either a try statement should be associated with a catch block or with finally.
* Since catch performs exception handling and finally performs the cleanup, the best approach is to merge both of them.

Syntax:

try

{

//statements that may cause an exception

}

catch (…)‏

{

//error handling code

}

finally

{

//statements to be executed

}

**Examples of Try catch finally blocks**

Example 1: Below example illustrates finally block when no exception occurs in try block

class Example1{

public static void main(String args[]){

try{

System.out.println("First statement of try block");

int num=45/3;

System.out.println(num);

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("ArrayIndexOutOfBoundsException");

}

finally{

System.out.println("finally block");

}

System.out.println("Out of try-catch-finally block");

}

}

**Output:**

finally block

Out of try-catch-finally block

Example 2:Below example illustrates finally block execution when exception occurs in try block but doesn’t get handled in catch block.

class Example2{

public static void main(String args[]){

try{

System.out.println("First statement of try block");

int num=45/0;

System.out.println(num);

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println("ArrayIndexOutOfBoundsException");

}

finally{

System.out.println("finally block");

}

System.out.println("Out of try-catch-finally block");

}

}

**Output:**

First statement of try block

finally block

Exception in thread "main" java.lang.ArithmeticException: / by zero

at beginnersbook.com.Example2.main(Details.java:6)

Example 3:Below example illustrates execution of finally, when exception occurs in try block and handled in catch block.

class Example3{

public static void main(String args[]){

try{

System.out.println("First statement of try block");

int num=45/0;

System.out.println(num);

}

catch(ArithmeticException e){

System.out.println("ArithmeticException");

}

finally{

System.out.println("finally block");

}

System.out.println("Out of try-catch-finally block");

}

}

**Output:**

First statement of try block

ArithmeticException

finally block

Out of try-catch-finally block