3) Что выполнит java при перехвате exception-а:

try{}

catch {return 1;}

finally {return 2;}

Ответ: http://stackoverflow.com/questions/5701793/behaviour-of-return-statement-in-catch-and-finally

finally is always executed (the only exception is System.exit()). You can think of this behavior this way:

1. An exception is thrown
2. Exception is caught and return value is set to 5
3. Finally block gets executed and return value is set to 10
4. The function returns

Или же This is an easy question if you remember the low level layout of the VM.

1. The return value is put up the stack by the catch code.
2. Afterwards, the finally code is executed and overwrites the value on the stack.
3. Then, the method returns with the most up to date value (10) to be used by the caller.

If unsure about things like this, fall back to your understanding of the underlying system (ultimately going to assembler level).

Или же The problem is that the finally block would remove any exceptions being thrown since it would issue a "normal" return.

From the [JLS spec](http://docs.oracle.com/javase/specs/jls/se5.0/html/statements.html):

Abrupt completion of a finally clause can disrupt the transfer of control initiated by a return statement.

and (more relevant in your case):

Note that abrupt completion of a finally clause can disrupt the transfer of control initiated by a throw statement.

Или же http://stackoverflow.com/questions/22781437/return-statement-finally-block-does-not-complete-normally

The finally block is always run, regardless whether an exception was thrown (and maybe caught) or not. If a finally block terminates in an *unnormal* way (such as itself throwing an excpetion or returning a value) this will always override what was done in the try block or a catch block. That also means that these are got lost.

The conclusion: Never throw an exception ot return a value from the finally block. Only use it for cleaning up processes.

Или же

|  |  |
| --- | --- |
|  | Returning from inside a finally block will cause exceptions to be lost.  **A return statement inside a finally block will cause any exception that might be thrown in the try or catch block to be discarded.**  According to the [**Java Language Specification:**](http://docs.oracle.com/javase/specs/jls/se7/html/jls-14.html#jls-14.20.2)  If execution of the try block completes abruptly for any other reason R, then the finally block is executed, and then there is a choice:  If the finally block completes normally, then the try statement  completes abruptly for reason R.  If the finally block completes abruptly for reason S, then the try  statement completes abruptly for reason S (and reason R is  discarded). |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://www.programmerinterview.com/index.php/java-questions/will-finally-run-after-return/>

**In Java, will the code in the finally block be called and run after a return statement is executed?**

The answer to this question is a simple yes – the code in a finally block will take precedence over the return statement. Take a look at the code below to confirm this fact:

**Code that shows finally runs after return**

class SomeClass

{

public static void main(String args[])

{

// call the proveIt method and print the return value

System.out.println(SomeClass.proveIt());

}

public static int proveIt()

{

try {

return 1;

}

finally {

System.out.println("finally block is run

before method returns.");

}

}

}

Running the code above gives us this output:

finally block is run before method returns.

1

From the output above, you can see that the finally block is executed ***before*** control is returned to the “System.out.println(SomeClass.proveIt());” statement – which is why the “1” is output after the “finally block is run before method returns.” text.

**Very unique situations when finally will not run after return**

The finally block will not be called after return in a couple of unique scenarios: if System.exit() is called first, or if the JVM crashes.

**What if there is a return statement in the finally block as well?**

If you have a return statement in both the finally block and the try block, then you could be in for a surprise. Anything that is returned in the finally block will actually ***override*** any exception or returned value that is inside the try/catch block. Here is an example that will help clarify what we are talking about:

public static int getANumber(){

try{

return 7;

} finally {

return 43;

}

}

The code above will actually return the “43” instead of the “7”, because the return value in the finally block (“43″) will override the return value in the try block (“7″).

Also, if the finally block returns a value, it will override any exception thrown in the try/catch block. Here is an example:

public static int getANumber(){

try{

throw new NoSuchFieldException();

} finally {

return 43;

}

}

**A return statement in the finally block is a bad idea**

Running the method above will return a “43” and the exception in the try block will not be thrown. This is why it is considered to be a very bad idea to have a return statement inside the finally block.