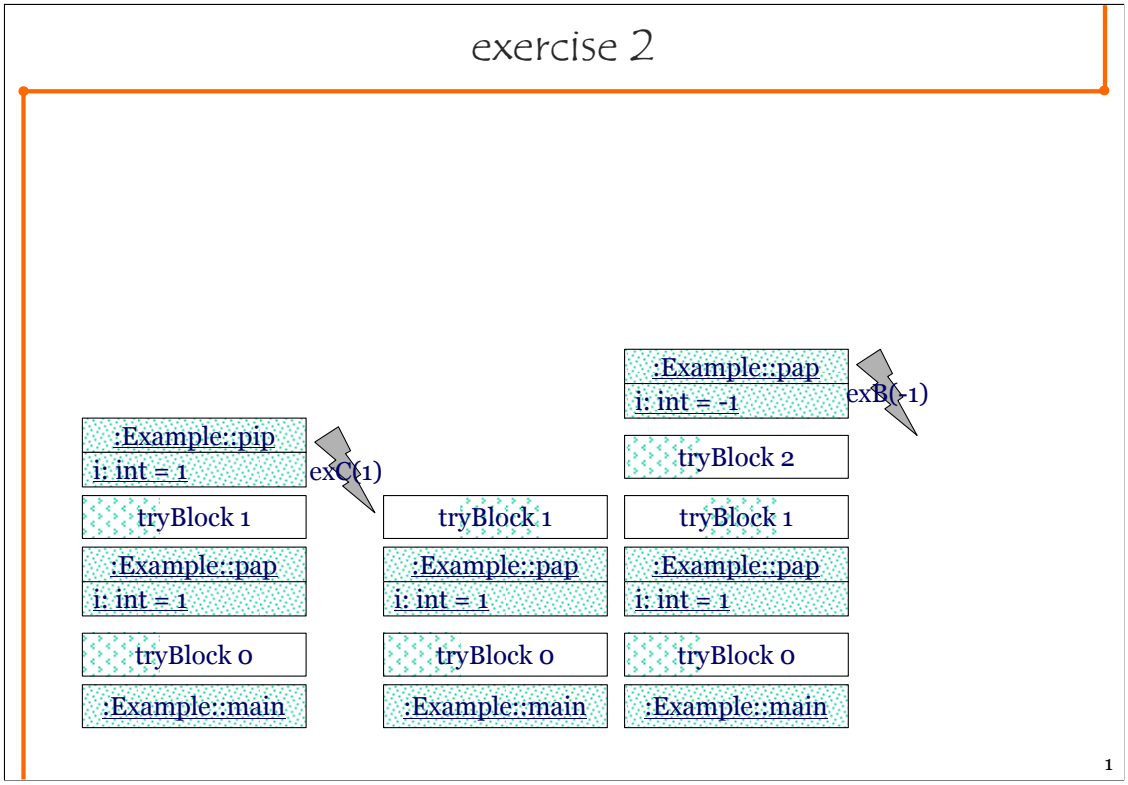


The first column represents the call stack when we start the main method, and until the first exception is thrown.

The dotted boxes represent normal method calls, and the not dotted boxes represent try-blocks.

The try blocks have a pattern which can be to the left, in the middle or the the right.

If the pattern is in the left part, it means that we are in the try part, if the pattern is in the middle we are executing a catch part, and if it is in the right, we are executing something in the finally part of the try – catch – finally statement.

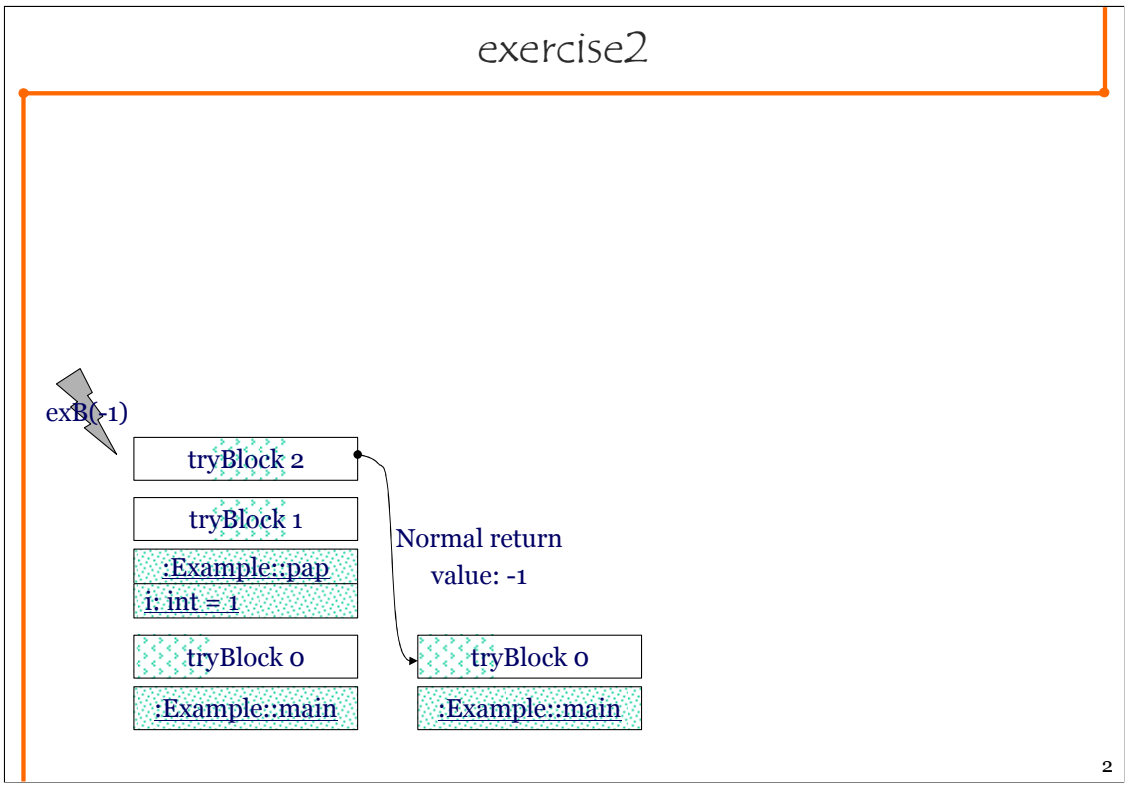
When an exception of type T is thrown, on looks down through the call stack to find a try block which its pattern in the left side, and with a matching catch. The try block which has a matching catch is then marked in the middle, and the execution continues from there.

If the try block has its pattern in the middle or in the right, it will be ignored when we look for a matching try block.

All finally parts between the top of the stack, and the matching try block are executed before the matching catch is executed.

An exceptionC is thrown in pip (first stack picture). The first try block to examine is try block 1, which do have a matching catch. The pattern is shown in the middle to indicate this (middle stack picture). The catch in try block 1 contains the nested try block 2, which calls pap recursively, decrementing the argument by 2 (right stack picture).

Because the argument is now less than 0 (it is –1), an exceptionB is thrown.



This exceptionB is caught in try block2 (left stack picture). The behaviour of the catch in try block to is to return from the surrounding method call with the value found in the exception (right picture). The value stored in the exception was –1.

Notice, we return normally to try block 0. Therefore in the right picture, the try block has its pattern in the left part still. The return value of –1 is printed out in try block 0.

If one runs the program with the initial argument to pap being 5, the execution becomes quite different, in that pip will throw an exceptionA, which is caught in pap; but pap then throws an exceptionB, which is caught in main, and we do not have a return value to print.