



Exceptions

Self-review Questions

8.1 *Is there anything wrong with simply stopping a program when an error occurs?*

Yes, for a number of reasons:

- It leads to a bad user experience when, from the users point of view, the program appears to have crashed.
- Any data held in memory, along with the general state of the program, is lost and cannot be recovered.
- Useful or detailed information about why the error occurred is lost, especially if no stack trace is generated. This makes it impossible to accurately report the cause of an error and very hard for the developer to reproduce and fix the problem.

8.2 *What kinds of error should the exception handling mechanism be used to catch?*

8.3 *Why should methods that throw exceptions not also try to catch them?*

8.4 *What happens to an uncaught exception?*

If a method does not catch an exception in a try-catch block the exception will be *propagated*, or passed back, to the calling method. If the calling method does not catch the exception it will in turn propagate the exception back to its calling method. If the exception is propagated all the way up the method call chain to the `main` method where it is still not caught the program will terminate displaying a stack trace.

If an exception is thrown and not caught in a thread started by a program, the exception is propagated back to the `run` method and the thread terminates displaying a stack trace. The program will continue running if there are other threads but may or may not be able to continue depending on how critical the loss of the thread that threw the exception is.

8.5 *What is the purpose of a throws declaration?*

The throws declaration states that the named exception will not be handled in the method but, if it is raised, it will be propagated.

8.6 *How does a catch block work?*

A catch block is a place where a thrown exception can be handled. A catch block specifies which exception it will handle, it will also handle any subclasses of the named exception. If an exception is thrown in the try block then each catch block associated with the try block is tried in turn, in the order they appear in the source code. If the thrown exception is compatible with the exception handled by the catch block then that catch block will handle the exception. If it does not then the next is tried, and so on.

8.7 *What does finally do?*

A finally block, if present, will always be executed. If a try block throws an exception, then the catch blocks are searched. If one of them handles the exception then the finally block will be executed immediately afterwards but before any return. If no catch block handles the exception then the finally block is executed prior to exception propagation. If there is no exception raised then the finally block is executed prior to execution continuing with the statements immediately following the try-catch-finally block. If the try block or a catch block executes a return the finally block is executed prior to the return.

Programming Exercises

- 8.1 *Create an implementation of the interface `Queue` that throws exceptions appropriately. Declare any exception classes needed and write a test class.*
- 8.2 *Create an implementation of interface `Deque` that throws exceptions appropriately. Declare any exception classes needed and write a test class.*
- 8.3 *Update the `Matrix` interface and related classes shown in Section 7.10, page 247, so that they throw exceptions to deal with error conditions. Write a test class to check that the exception handling works correctly.*
- 8.4 *Create a program to find out how much memory Java can use on the machines you have access to. Find out how to change the amount of memory available using command line options and check if your program has all this memory available to it.*