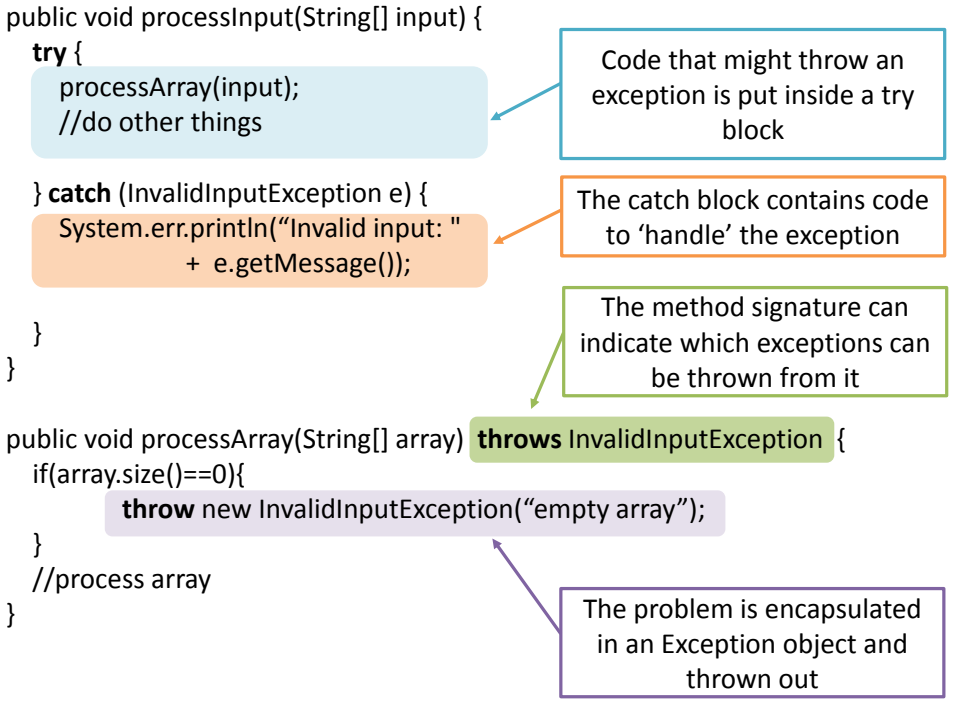
[L3P2]

Exceptions: Stuff happens. Deal with it.

Exceptions are used to deal with 'unusual' but not entirely unexpected situations that the program might encounter at run time.

> An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions. –Java Tutorial (Oracle Inc.)

For example, a network connection might timeout due to a slow server. That is not a program bug but rather an unusual situation that needs to be recovered from, if possible. However, the code segement that encountered the unusual situation might not know how to recover from it. That is why most languages allow a method to encapsulate the unusual situation in an Exception object and 'throw' that object so that another piece of code can 'catch' it and deal with it. Usually, an exception thrown by a method is caught by the caller method. If the called method does not know how to deal with the exception it caught, it will throw the Exception object to its own caller. If none of the callers is prepared to deal with the exception, the exceptions can propagate through the method call stack until it is received by the main method and thrown to the runtime, thus halting the system. In the Java code given below, processArray can potentially throw an InvalidInputException. Because of that, processInput method invokes processArray method inside a try{ } block and has a catch{ } block to specify what to do if the exception is actually thrown.



The ability to propagate error information through the call stack is one advantage of using Exceptions. Another advantage is the separation of code that deals with 'unusual' situations from the code that does the 'usual' work.