```
try {                                                          //1
  while(true)
  {
    data = in.getData();
    //Do something with data
  }
}
catch (NoMoreDataException e1) {}
```

Notice at //1 that a `try/catch` block is placed around the loop containing the `getData` method. The `getData` method no longer returns zero when the stream is empty, but rather throws a `NoMoreDataException`. Although this code is a bit uglier than the original code, some programmers believe that code should be written in this manner. They argue that code that can utilize exceptions should no longer use traditional error handling techniques.

The code without the exceptions is much more intuitive, although it relies on the older methods to deal with errors or unexpected results. Exceptions can be overused, as the latter code example illustrates.

Use exceptions for conditions outside of the expected behavior of the code. In the previous examples, you expect to get to the end of the stream, so a simple zero return from the `getData` method is appropriate and natural. Throwing an exception in this case is unwise, since this is not an exceptional condition but rather an expected one. You do not, however, expect the stream to be corrupted. That type of condition calls for an exception to be generated. The point is not to use exceptions for all conditions, but to use them where it makes sense, that is, where exceptional conditions exist.

The exclusive use of exceptions, as in the second example, might make your program "pure" as far as exceptions are concerned. However, you have used exceptions for every error condition. Simply returning zero is much faster and more intuitive than creating an exception object and requiring the caller to implement a `catch` block to handle it.

## PRAXIS 26: **Throw exceptions from constructors**

Traditional error reporting from constructors can be problematic because a constructor has no return value. Thus, you cannot simply return an error code when a constructor fails. True, constructors are not methods, but you are not precluded from throwing an exception from a constructor.