PRAXIS 23: Place `try/catch` blocks outside of loops

Exceptions can have a negative impact on the performance of your code. Whether exceptions negatively affect performance has to do with how you structure your code and whether your JVM uses a JIT compiler that optimizes your code as it runs. Here are two areas of exceptions and performance to consider:

- The effect of throwing an exception
- The effects of `try/catch` blocks in your code

The act of throwing an exception is not free. After all, what are Java exceptions? They are objects and objects need to be created. Creating objects is costly (see PRAXIS 32), so, throwing an exception has costs. To throw an exception, you write something similar to this:

```
throw new MyException();
```

This code creates a new object and then transfers control to a `catch` or `finally` block or to the calling method. Because throwing exceptions entails some cost, use exceptions only for error conditions. When exceptions are used for control flow, the code is not as efficient or clear as it will be if typical flow constructs are used (see PRAXIS 24). Limit using exceptions for error and failure conditions. You want code to run fast when it works and typically do not care how long it takes to fail.

When considering the performance aspects of Java, you must take into account the JVM and operating system being utilized. Differences in execution times are observed between JVM's that are running identical code. Therefore, you must perform some level of profiling on your systems to determine the performance differences. The performance data in this PRAXIS was generated with the hardware and software setup detailed on page 98.

Placing `try/catch` blocks inside of loops can slow down the execution of code as the following code illustrates:

```
class ExcPerf
{
  public void method1(int size)
  {
    int[] ia = new int[size];
    try {
      for (int i=0; i<size; i++)
        ia[i] = i;
```