```
}
```

22. What is the purpose of the call `super(message)` in the second `InsufficientFundsException` constructor?
23. Suppose you read bank account data from a file. Contrary to your expectation, the next input value is not of type `double`. You decide to implement a `BadDataException`. Which exception class should you extend?

Practice It Now you can try these exercises at the end of the chapter: R11.7, R11.8, R11.9.

Programming Tip 11.1

Throw Early, Catch Late

When a method detects a problem that it cannot solve, it is better to throw an exception rather than try to come up with an imperfect fix. For example, suppose a method expects to read a number from a file, and the file doesn't contain a number. Simply using a zero value would be a poor choice because it hides the actual problem and perhaps causes a different problem elsewhere.

Throw an exception as soon as a problem is detected. Catch it only when the problem can be handled.

Conversely, a method should only catch an exception if it can really remedy the situation. Otherwise, the best remedy is simply to have the exception propagate to its caller, allowing it to be caught by a competent handler.

These principles can be summarized with the slogan "throw early, catch late".

Programming Tip 11.2

Do Not Squelch Exceptions

When you call a method that throws a checked exception and you haven't specified a handler, the compiler complains. In your eagerness to continue your work, it is an understandable impulse to shut the compiler up by squelching the exception:

```
try
{
   Scanner in = new Scanner(new File(filename));
   // Compiler complained about FileNotFoundException
   . . .
}
catch (FileNotFoundException e) {} // So there!
```

The do-nothing exception handler fools the compiler into thinking that the exception has been handled. In the long run, this is clearly a bad idea. Exceptions were designed to transmit problem reports to a competent handler. Installing an incompetent handler simply hides an error condition that could be serious.

Programming Tip 11.3

Do Not Use `catch` and `finally` in the Same `try` Statement

It is possible to have a `finally` clause following one or more `catch` clauses. Then the code in the `finally` clause is executed whenever the `try` block is exited in any of three ways:

1. After completing the last statement of the `try` block
2. After completing the last statement of a `catch` clause, if this `try` block caught an exception
3. When an exception was thrown in the `try` block and not caught