Chapter 1: Introduction

"The primary duty of an exception handler is to get the error out of the lap of the programmer and into the surprised face of the user. Provided you keep this cardinal rule in mind, you can't go far wrong."

—Verity Stob

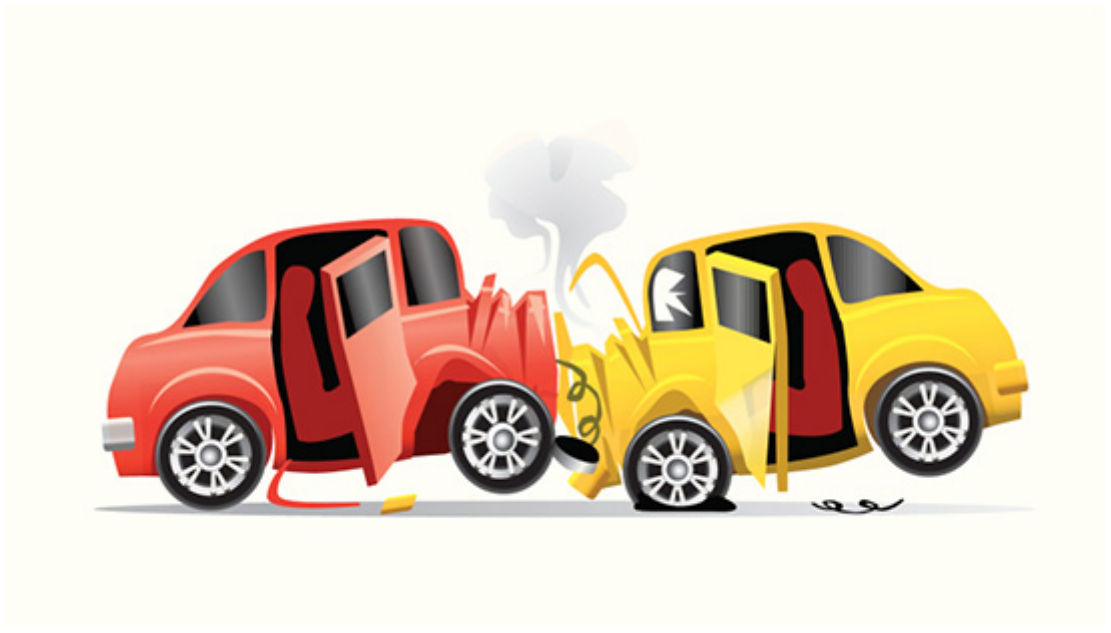
Introduction

Our temperature conversion project meets all the requirements of our original design, but some aspects of it aren't ideal. The client class makes some of the decisions and does some of the processing regarding temperatures, and that should really be the job of the Temperature class.

For example, our client class (TemperatureDriver) has to check each temperature's type before accepting it as valid and creating a Temperature object. What if we decided we needed to incorporate a fourth temperature scale into our class? Then we'd have to change the code of every program using our class—which could potentially be a large number—to check and process that fourth temperature type.

Wouldn't it be better if our Temperature class could tell users whether a temperature type is valid before creating a Temperature object? We can do that, and in the next chapter, I'll show you how.

We've also looked (briefly) at what happens when our program runs into an exception. Programmers say it *crashes* or *blows up*. Not very convenient, wouldn't you say? So in this lesson we'll look at how Java generates exceptions. You'll find out how we can intercept those exceptions and use them to make our programs more *robust* (less likely to "crash and burn").



A crash happens when program and user don't get along!

Let's get started!