11.1 Introduction

As you know from Chapters 7–8, an exception indicates a problem during a program's execution. Exception handling enables applications to resolve (or handle) exceptions. In some cases, a program can continue executing as if no problem had been encountered. The features presented in this chapter help you write *robust* and *fault-tolerant* programs that can deal with problems and continue executing or *terminate gracefully*.[1]

[1] Java exception handling is based in part on the work of Andrew Koenig and Bjarne Stroustrup—A. Koenig and B. Stroustrup, "Exception Handling for C++ (revised)," *Proceedings of the Usenix C++ Conference*, pp. 149–176, San Francisco, April 1990.

First, we handle an exception that occurs when a method attempts to divide an integer by zero. We discuss when to use exception handling and show a portion of the exception-handling class hierarchy. As you'll see, only subclasses of `Throwable` can be used with exception handling. We introduce the `try` statement's `finally` block, which executes whether or not an exception occurs. We then show how to use *chained exceptions* to add application-specific information to an exception and how to create your own exception types. Next, we introduce *preconditions* and *postconditions*, which must be true when your methods are called and when they return, respectively. We then present *assertions*, which you can use at development time to help debug your code. We also discuss catching multiple exceptions with one `catch` handler, and the `try`-with-resources statement that automatically releases a resource after it’s used in the `try` block.

This chapter focuses on the exception-handling concepts and presents several mechanical examples that demonstrate various features. Many Java API methods throw exceptions that we handle in our code. Figure 11.1 shows some of the exception types you’ve already seen and others you’ll learn about.

Software Engineering Observation 11.1

In industry, you’re likely to find that companies have strict design, coding, testing, debugging and maintenance standards. These often vary among companies. Companies often have their own exception-handling standards that are sensitive to the type of application, such as real-time systems, high-performance mathematical calculations, big data, network-based distributed systems, etc. This chapter’s tips provide observations consistent with these types of industry policies.

Chapter	Sample of exceptions used
Chapter 7	`ArrayIndexOutOfBoundsException`
Chapters 8–10	`IllegalArgumentException`

Chapter 11	`ArithmeticException`, `InputMismatchException`
Chapter 15	`SecurityException`, `FileNotFoundException`, `IOException`, `ClassNotFoundException`, `IllegalStateException`, `FormatterClosedException`, `NoSuchElementException`
Chapter 16	`ClassCastException`, `UnsupportedOperationException`, `NullPointerException`, custom exception types
Chapter 20	`ClassCastException`, custom exception types
Chapter 21	`IllegalArgumentException`, custom exception types
Chapter 23	`InterruptedException`, `IllegalMonitorStateException`, `ExecutionException`, `CancellationException`
Chapter 24	`SQLException`, `IllegalStateException`, `PatternSyntaxException`
Chapter 28	`MalformedURLException`, `EOFException`, `SocketException`, `InterruptedException`, `UnknownHostException`
Chapter 31	`SQLException`

Fig. 11.1

Various exception types that you’ll see throughout this book