11.12 `try`-with-Resources: Automatic Resource Deallocation

Typically *resource-release code* should be placed in a `finally` block to ensure that a resource is released, regardless of whether there were exceptions when the resource was used in the corresponding `try` block. An alternative notation—the **`try`-with-resources** statement—simplifies writing code in which you obtain one or more resources, use them in a `try` block and release them in a corresponding `finally` block. For example, a file-processing application could process a file with a `try`-with-resources statement to ensure that the file is closed properly when it's no longer needed—we demonstrate this in Chapter 15. Each resource must be an object of a class that implements the `AutoCloseable` interface and thus provides a `close` method.

The general form of a `try`-with-resources statement is

```
try (ClassName theObject = new ClassName()) {
    // use theObject here, then release its resources
    // the end of the try block
}
catch (Exception e) {
   // catch exceptions that occur while using the res
   }
```

where *ClassName* is a class that implements `AutoCloseable`. This code creates a *ClassName* object, uses it in the `try` block, then calls its `close` method *at the end of the* `try` *block—or, if an exception occurs, at the end of a* `catch` *block*—to release the object's resources. You can create multiple `AutoCloseable` objects in the parentheses following `try` by separating them with a semicolon (`;`). You'll see examples of the `try`-with-resources statement in Chapters 15 and 24.

Java SE 9: `try`-with-Resources Can Use Effectively `final` Variables

8

Java SE 8 introduced **effectively** `final` local variables. If the compiler can *infer* that the variable could have been declared `final`, because its enclosing method never modifies the variable after it's declared and initialized, then the variable is effectively `final`. Such variables frequently are used with lambdas (Chapter 17, Lambdas and Streams).

9

As of Java SE 9, you can create an `AutoCloseable` object and assign it to a local variable that's explicitly declared `final` or that's effectively `final`. Then, you can use it in a `try`-with-resources statement that releases the object's resources at the end of the `try` block.

```
ClassName theObject = new ClassName();

try (theObject) {
   // use theObject here, then release its resources
   // the end of the try block
}
catch (Exception e) {
   // catch exceptions that occur while using the res
}
```

As before, you can separate with a semicolon (`;`) multiple `AutoCloseable` objects in the parentheses following `try`. This simplifies the `try`-with-resources statement's code, especially for cases in which the statement uses and releases multiple `AutoCloseable` objects.