25.16 Wrap-Up

9

In this chapter, you used JShell—Java 9's new interactive REPL for exploration, discovery and experimentation. We showed how to start a JShell session and work with various types of code snippets, including statements, variables, expressions, methods and classes—all without having to declare a class containing a `main` method to execute the code.

You saw that you can list the valid snippets in the current session, and recall and execute prior snippets and commands using the up and down arrow keys. You also saw that you can list the current session's variables, methods, types and `import`s. We showed how to clear the current JShell session to remove all existing snippets and how to save snippets to a file then reload them.

We demonstrated JShell's auto-completion capabilities for code and commands, and showed how you can explore a class's members and view documentation directly in JShell. We explored class `Math`, demonstrating how to list its `static` members, how to view a method's parameters and overloads, view a method's documentation and view a `public` field's documentation. We also explored the methods of a `String` object.

You declared methods and forward referenced an undeclared method that you declared later in the session, then saw that you could go back and execute the first method. We also showed that you can replace a method declaration with a new method—in fact, you can replace any declaration of a variable, method or type.

We showed that JShell catches all exceptions and simply displays a stack trace followed by the next `jshell>` prompt, so you can continue the session. You imported an existing compiled class from a package, then used that class in a JShell session.

Next, we summarized and demonstrated various other JShell commands. We showed how to configure a custom snippet editor, view JShell's help documentation, reload a session, drop snippets from a session, configure feedback modes and more. We listed some additional keyboard shortcuts for editing the current snippet at the `jshell>` prompt. Finally, we discussed how JShell reinterprets Java for interactive use and IDE support for JShell.