25.12 Summary of JShell Commands

Figure 25.6 shows the basic JShell commands. Many of these commands have been presented throughout this chapter. Others are discussed in this section.

Command	Description
`/help` or `/?`	Displays JShell's list of commands.
`/help intro`	Displays a brief introduction to JShell.
`/help shortcuts`	Displays a description of several JShell shortcut keys.
`/list`	By default, lists the valid snippets you've entered in the current session. To list all snippets, use `/list -all`.
`/!`	Recalls and re-evaluates the last snippet.
`/`*id*	Recalls and re-evaluates the snippet with the specified *id*.
`/-`*n*	Recalls and re-evaluates a prior snippet—for *n*, `1` is the last snippet, `2` is the second-to-last, etc.
`/edit`	By default, opens a **JShell Edit Pad** window containing the valid snippets you've entered in the current session. See Section 25.11 to learn how to configure an external editor.
`/save`	Saves the current session's valid snippets to a specified file.
`/open`	Opens a specified file of code snippets, loads the snippets into the current session and evaluates the loaded snippets.

`/vars`	Displays the current session’s variables and their corresponding values.
`/methods`	Displays the signatures of the current session’s declared methods.
`/types`	Displays types declared in the current session.
`/imports`	Displays the current session’s `import` declarations.
`/exit`	Terminates the current JShell session.
`/reset`	Resets the current JShell session, deleting all code snippets.
`/reload`	Reloads a JShell session and executes the valid snippets (Section 25.12.3).
`/drop`	Deletes a specified snippet from the current session (Section 25.12.4).
`/env`	Makes changes to the JShell environment, such as adding packages or modules so you can use their types in JShell.
`/history`	Lists everything you’ve typed in the current JShell session, including all snippets (valid, invalid or overwritten) and JShell commands—the `/list` command shows only snippets, not JShell commands.
`/set`	Sets various JShell configuration options, such as the editor used in response to the `/edit` command, the text used for the JShell prompts, the imports to specify when a session starts, etc. (Sections 25.12.5–25.12.6).

Fig. 25.6

Jshell commands.

25.12.1 Getting Help in

JShell

JShell’s help documentation is incorporated directly via the **`/help`** or **`/?`** commands—`/?` is simply a shorthand for `/help`. For a quick introduction to JShell, type:

```
/help intro
```

To display JShell’s list of commands, type

```
/help
```

For more information on a given command’s options, type

```
/help command
```

For example

```
/help /list
```

displays the `/list` command’s more detailed help documentation. Similarly

```
/help /set start
```

displays more detailed help documentation for the `/set` command's `start` option. For a list of the shortcut key combinations in JShell, type

```
/help shortcuts
```

25.12.2 `/edit` Command: Additional Features

We've discussed using `/edit` to load all valid snippets, a snippet with a specified ID or a method with a specified name into **JShell Edit Pad**. You can specify the identifier for any variable, method or type declaration that you'd like to edit. For example, if the current JShell session contains the declaration of a class named `Account`, the following loads that class into **JShell Edit Pad**:

```
/edit Account
```

25.12.3 `/reload` Command

At the time of this writing, you cannot use the `/`*id* command to execute a range of previous snippets. However, JShell's

`/reload` command can re-execute all valid snippets in the current session. Consider the session from Sections 25.3.9–25.3.10:

```
jshell> /list

   1 : 45
   2 : 72
   3 : if ($1 < $2) {
          System.out.printf("%d < %d%n", $1, $2);
       }
   4 : if ($1 > $2) {
          System.out.printf("%d > %d%n", $1, $2);
       }
   5 : $1 = 100;
   6 : if ($1 > $2) {
          System.out.printf("%d > %d%n", $1, $2);
       }

jshell>
```

The following reloads that session one snippet at a time:

```
jshell> /reload
|  Restarting and restoring state.
-: 45
-: 72
-: if ($1 < $2) {
      System.out.printf("%d < %d%n", $1, $2);
   }
45 < 72
-: if ($1 > $2) {
      System.out.printf("%d > %d%n", $1, $2);
   }
-: $1 = 100
-: if ($1 > $2) {
        System.out.printf("%d > %d%n", $1, $2);
     }
100 > 72

jshell>
```

Each reloaded snippet is preceded by `-:` and in the case of the `if` statements, the output (if any) is shown immediately following each `if` statement. If you prefer not to see the snippets as they reload, you can use the `/reload` command's `-quiet` option:

```
jshell> /reload -quiet
| Restarting and restoring state.
45 < 72
100 > 72

jshell>
```

In this case, only the results of output statements are displayed. Then, you can view the snippets that were reloaded with the `/list` command.

25.12.4 `/drop` Command

You can eliminate a snippet from the current session with JShell's **`/drop`** command followed by a snippet ID or an identifier. The following new JShell session declares a variable `x` and a method `cube`, then drops `x` via its snippet ID and drops `cube` via its identifier:

```
jshell> int x = 10
x ==> 10

jshell> int cube(int y) {return y * y * y;}
|  created method cube(int)

jshell> /list

   1 : int x = 10;
   2 : int cube(int y) {return y * y * y;}

jshell> /drop 1
|  dropped variable x

jshell> /drop cube
|  dropped method cube(int)

jshell> /list

jshell>
```

25.12.5 Feedback Modes

JShell has several feedback modes that determine what gets displayed after each interaction. To change the feedback mode, use JShell's **`/set feedback`** command:

```
/set feedback mode
```

where *mode* is `concise`, `normal` (the default), `silent` or `verbose`. All of the prior JShell interactions in this chapter used the `normal` mode.

Feedback Mode `verbose`

Below is a new JShell session in which we used `verbose` mode, which beginning programmers might prefer:

```
jshell> /set feedback verbose
|  Feedback mode: verbose

jshell> int x = 10
x ==> 10
|  created variable x : int

jshell> int cube(int y) {return y * y * y;}
|  created method cube(int)

jshell> cube(x)
$3 ==> 1000
|  created scratch variable $3 : int

jshell> x = 5
x ==> 5
|  assigned to x : int

jshell> cube(x)
$5 ==> 125
|  created scratch variable $5 : int

jshell>
```

Notice the additional feedback indicating that

- variable `x` was created,
- variable `$3` was created on the first call to `cube`—JShell refers to the implicit variable as a *scratch variable*,
- an `int` was assigned to the variable `x`, and
- scratch variable `$5` was created on the second call to `cube`.

Feedback Mode `concise`

Next, we `/reset` the session then set the feedback mode to `concise` and repeat the preceding session:

```
jshell> /set feedback concise
jshell> int x = 10
jshell> int cube(int y) {return y * y * y;}
jshell> cube(x)
$3 ==> 1000
jshell> x = 5
jshell> cube(x)
$5 ==> 125
jshell>
```

As you can see, the only feedback displayed is the result of each call to `cube`. If an error occurs, its feedback also will be displayed.

Feedback Mode `silent`

Next, we `/reset` the session then set the feedback mode to `silent` and repeat the preceding session:

```
jshell> /set feedback silent
-> int x = 10
-> int cube(int y) {return y * y * y;}
-> cube(x)
-> x = 5
-> cube(x)
-> /set feedback normal
|  Feedback mode: normal

jshell>
```

In this case, the `jshell>` prompt becomes `->` and only error feedback will be displayed. You might use this mode if you've copied code from a Java source file and want to paste it into JShell, but do not want to see the feedback for each line.

25.12.6 Other JShell Features Configurable with `/set`

So far, we've demonstrated the `/set` command's capabilities for setting an external snippet editor and setting feedback modes. The `/set` command provides extensive capabilities for creating custom feedback modes via the commands:

- `/set mode`

- `/set prompt`
- `/set truncation`
- `/set format`

The `/set mode` command creates a user-defined custom feedback mode. Then you can use the other three commands to customize all aspects of JShell’s feedback. The details of these commands are beyond the scope of this chapter. For more information, see JShell’s help documentation for each of the preceding commands.

Customizing JShell Startup

Section 25.10 showed the set of common packages JShell `import`s at the start of each session. Using JShell’s **`/set start`** command

```
/set start filename
```

you can provide a file of Java snippets and JShell commands that will be used in the current session when it restarts due to a `/reset` or `/reload` command. You can also remove all startup snippets with

```
/set start -none
```

or return to the default startup snippets with

```
/set start -default
```

In all three cases, the setting applies only to the current session unless you also include the `-retain` option. For example, the following command indicates that all subsequent JShell sessions should load the specified file of startup snippets and commands:

```
/set start -retain filename
```

You can restore the defaults for future sessions with

```
/set start -retain -default
```