25.4 Command-Line Input in JShell

[*Note:* This section may be read after studying Chapter 2, Introduction to Java Applications; Input/Output and Operators and the preceding sections in this chapter.]

In Chapter 2, we showed command-line input using a `Scanner` object:

```
Scanner input = new Scanner(System.in);

System.out.print("Enter first integer: ");
int number1 = input.nextInt();
```

We created a `Scanner`, prompted the user for input, then used `Scanner` method `nextInt` to read a value. Recall that the program then waited for you to type an integer and press *Enter* before proceeding to the next statement. The on-screen interaction appeared as:

```
Enter first integer: 45
```

This section shows what that interaction looks like in JShell.

Creating a `Scanner`

Start a new JShell session or `/reset` the current one, then create a `Scanner` object:

```
jshell> Scanner input = new Scanner(System.in)
input ==> java.util.Scanner[delimiters=\p{javaWhitesp
   \E][infinity string=\Q∞\E]

jshell>
```

You do not need to import `Scanner`. JShell automatically imports the `java.util` package and several others—we show the complete list in Section 25.10. When you create an object, JShell displays its text representation. The notation to the right of `input ==>` is the `Scanner`'s text representation (which you can simply ignore).

Prompting for Input and Reading a Value

Next, prompt the user for input:

```
jshell> System.out.print("Enter first integer: ")
Enter first integer:
jshell>
```

The statement's output is displayed immediately, followed by

the next `jshell>` prompt. Now enter the input statement:

```
jshell> int number1 = input.nextInt()
_
```

At this point, JShell waits for your input. The input cursor is positioned below the `jshell>` prompt and snippet you just entered—indicated by the underscore (_) above—rather than next to the prompt `"Enter first integer:"` as it was in Chapter 2. Now type an integer and press *Enter* to assign it to `number1`—the last snippet's execution is now complete, so the next `jshell>` prompt appears.:

```
jshell> int number1 = input.nextInt()
45
number1 ==> 45

jshell>
```

Though you can use `Scanner` for command-line input in JShell, in most cases it's unnecessary. The goal of the preceding interactions was simply to store an integer value in the variable `number1`. You can accomplish that in JShell with the simple assignment

```
jshell> int number1 = 45
number1 ==> 45

jshell>
```

For this reason, you'll typically use assignments, rather than command-line input in JShell. We introduced Scanner here, because sometimes you'll want to copy code you developed in JShell into a conventional Java program.