

FIGURE 2.7 Some Methods in the Class `Scanner`

`Scanner_Object_Name.next()`

Returns the `String` value consisting of the next keyboard characters up to, but not including, the first delimiter character. The default delimiters are whitespace characters.

`Scanner_Object_Name.nextLine()`

Reads the rest of the current keyboard input line and returns the characters read as a value of type `String`. Note that the line terminator `'\n'` is read and discarded; it is not included in the string returned.

`Scanner_Object_Name.nextInt()`

Returns the next keyboard input as a value of type `int`.

`Scanner_Object_Name.nextDouble()`

Returns the next keyboard input as a value of type `double`.

`Scanner_Object_Name.nextFloat()`

Returns the next keyboard input as a value of type `float`.

`Scanner_Object_Name.nextLong()`

Returns the next keyboard input as a value of type `long`.

`Scanner_Object_Name.nextByte()`

Returns the next keyboard input as a value of type `byte`.

`Scanner_Object_Name.nextShort()`

Returns the next keyboard input as a value of type `short`.

`Scanner_Object_Name.nextBoolean()`

Returns the next keyboard input as a value of type `boolean`. The values of `true` and `false` are entered as the words *true* and *false*. Any combination of uppercase and lowercase letters is allowed in spelling *true* and *false*.

`Scanner_Object_Name.useDelimiter(Delimiter_Word);`

Makes the string *Delimiter_Word* the only delimiter used to separate input. Only the exact word will be a delimiter. In particular, blanks, line breaks, and other whitespace will no longer be delimiters unless they are a part of *Delimiter_Word*.

This is a simple case of the use of the `useDelimiter` method. There are many ways to set the delimiters to various combinations of characters and words, but we will not go into them in this book.

So far it may not seem as though there is any potential for problems, but suppose the input were instead

```
42  
and don't you  
forget it.
```