3.19 (GUI) Confirmation Dialogs

You have used `showMessageDialog` to display a message dialog box and `showInputDialog` to display an input dialog box. Occasionally it is useful to answer a question with a confirmation dialog box. A confirmation dialog can be created using the following statement:

```
int option =
  JOptionPane.showConfirmDialog
    (null, "Continue");
```

When a button is clicked, the method returns an option value. The value is `JOptionPane.YES_OPTION` (0) for the *Yes* button, `JOptionPane.NO_OPTION` (1) for the *No* button, and `JOptionPane.CANCEL_OPTION` (2) for the *Cancel* button.

You may rewrite the guess-birthday program in Listing 3.3 using confirmation dialog boxes, as shown in Listing 3.10. Figure 3.6 shows a sample run of the program for the day `19`.

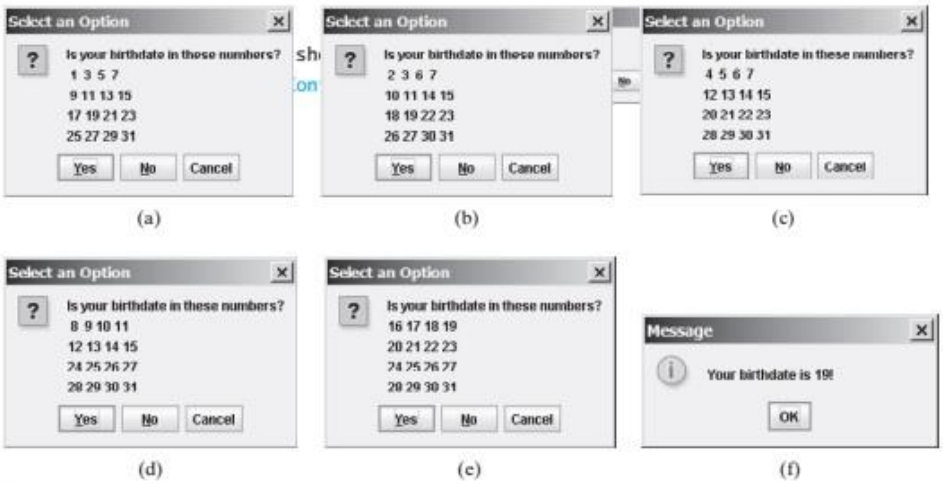

FIGURE 3.6 Click Yes in (a), Yes in (b), No in (c), No in (d), and Yes in (e).

LISTING 3.10 GuessBirthdayUsingConfirmationDialog.java

```
1  import javax.swing.JOptionPane;                                    import class
2
3  public class GuessBirthdayUsingConfirmationDialog {
4    public static void main(String[] args) {
5      String set1 =                                                  set1
6        " 1  3  5  7\n" +
7        " 9 11 13 15\n" +
8        "17 19 21 23\n" +
9        "25 27 29 31";
10
11     String set2 =                                                  set2
12       " 2  3  6  7\n" +
13       "10 11 14 15\n" +
14       "18 19 22 23\n" +
15       "26 27 30 31";
16
17     String set3 =                                                  set3
18       " 4  5  6  7\n" +
19       "12 13 14 15\n" +
20       "20 21 22 23\n" +
21       "28 29 30 31";
22
23     String set4 =                                                  set4
24       " 8  9 10 11\n" +
25       "12 13 14 15\n" +
26       "24 25 26 27\n" +
27       "28 29 30 31";
28
29     String set5 =                                                  set5
30       "16 17 18 19\n" +
31       "20 21 22 23\n" +
32       "24 25 26 27\n" +
33       "28 29 30 31";
34
```

confirmation dialog

in set1?

in set2?

in set3?

in set4?

in set5?

```
35      int day = 0;
36
37      // Prompt the user to answer questions
38      int answer = JOptionPane.showConfirmDialog(null,
39        "Is your birthday in these numbers?\n" + set1);
40
41      if (answer == JOptionPane.YES_OPTION)
42        day += 1;
43
44      answer = JOptionPane.showConfirmDialog(null,
45        "Is your birthday in these numbers?\n" + set2);
46
47      if (answer == JOptionPane.YES_OPTION)
48        day += 2;
49
50      answer = JOptionPane.showConfirmDialog(null,
51        "Is your birthday in these numbers?\n" + set3);
52
53      if (answer == JOptionPane.YES_OPTION)
54        day += 4;
55
56      answer = JOptionPane.showConfirmDialog(null,
57        "Is your birthday in these numbers?\n" + set4);
58
59      if (answer == JOptionPane.YES_OPTION)
60        day += 8;
61
62      answer = JOptionPane.showConfirmDialog(null,
63        "Is your birthday in these numbers?\n" + set5);
64
65      if (answer == JOptionPane.YES_OPTION)
66        day += 16;
67
68      JOptionPane.showMessageDialog(null, "Your birthday is " +
69        day + "!");
70    }
71  }
```

The program displays confirmation dialog boxes to prompt the user to answer whether a number is in Set1 (line 38), Set2 (line 44), Set3 (line 50), Set4 (line 56), and Set5 (line 62). If the answer is Yes, the first number in the set is added to `day` (lines 42, 48, 54, 60, and 66).

KEY TERMS

Boolean expression 72
Boolean value 72
`boolean` type 72
`break` statement 94
conditional operator 90
dangling-`else` ambiguity 82
fall-through behavior 94
operator associativity 97
operator precedence 97
selection statement 74
short-circuit evaluation 90

4.1 Introduction

Suppose that you need to print a string (e.g., `"Welcome to Java!"`) a hundred times. It would be tedious to have to write the following statement a hundred times:

100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
...
System.out.println("Welcome to Java!");
```

problem

So, how do you solve this problem?

why loop?

Java provides a powerful construct called a *loop* that controls how many times an operation or a sequence of operations is performed in succession. Using a loop statement, you simply tell the computer to print a string a hundred times without having to code the print statement a hundred times, as follows:

```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java!");
  count++;
}
```

The variable `count` is initially `0`. The loop checks whether (`count < 100`) is `true`. If so, it executes the loop body to print the message `"Welcome to Java!"` and increments `count` by `1`. It repeatedly executes the loop body until (`count < 100`) becomes `false`. When (`count < 100`) is `false` (i.e., when `count` reaches `100`), the loop terminates and the next statement after the loop statement is executed.

Loops are constructs that control repeated executions of a block of statements. The concept of looping is fundamental to programming. Java provides three types of loop statements: `while` loops, `do-while` loops, and `for` loops.

4.2 The `while` Loop

The syntax for the `while` loop is as follows:

while loop

```
while (loop-continuation-condition) {
  // Loop body
  Statement(s);
}
```

loop body

iteration

Figure 4.1(a) shows the `while`-loop flow chart. The part of the loop that contains the statements to be repeated is called the *loop body*. A one-time execution of a loop body is referred to as an *iteration of the loop*. Each loop contains a loop-continuation-condition, a Boolean expression that controls the execution of the body. It is evaluated each time to determine if the loop body is executed. If its evaluation is `true`, the loop body is executed; if its evaluation is `false`, the entire loop terminates and the program control turns to the statement that follows the `while` loop.

The loop for printing `Welcome to Java!` a hundred times introduced in the preceding section is an example of a `while` loop. Its flow chart is shown in Figure 4.1(b). The `loop-continuation-condition` is `(count < 100)` and loop body contains two statements as shown below:

loop-continuation-condition

loop body

```
int count = 0;
while (count < 100) {
  System.out.println("Welcome to Java!");
  count++;
}
```

ple run is shown in Figure 4.4.

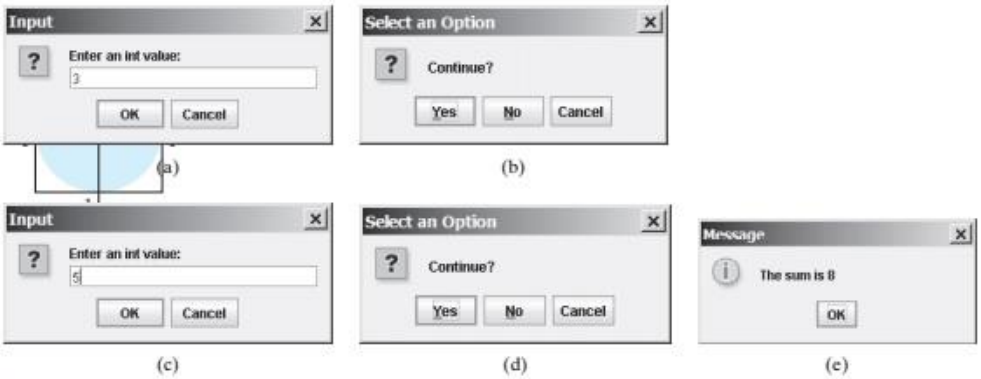

FIGURE 4.4 The user enters 3 in (a), clicks Yes in (b), enters 5 in (c), clicks No in (d), and the result is shown in (e).

LISTING 4.15 SentinelValueUsingConfirmationDialog.java

confirmation option
check option
input dialog
confirmation dialog
message dialog

```
1  import javax.swing.JOptionPane;
2
3  public class SentinelValueUsingConfirmationDialog {
4    public static void main(String[] args) {
5      int sum = 0;
6
7      // Keep reading data until the user answers No
8      int option = JOptionPane.YES_OPTION;
9      while (option == JOptionPane.YES_OPTION) {
10       // Read the next data
11       String dataString = JOptionPane.showInputDialog(
12         "Enter an int value: ");
13       int data = Integer.parseInt(dataString);
14
15       sum += data;
16
17       option = JOptionPane.showConfirmDialog(null, "Continue?");
18     }
19
20     JOptionPane.showMessageDialog(null, "The sum is " + sum);
21   }
22 }
```

A program displays an input dialog to prompt the user to enter an integer (line 11) and adds it to `sum` (line 15). Line 17 displays a confirmation dialog to let the user decide whether to continue the input. If the user clicks *Yes*, the loop continues; otherwise the loop exits. Finally the program displays the result in a message dialog box (line 20).

KEY TERMS

`break` statement 136
`continue` statement 136
`do-while` loop 124
`for` loop 126
loop control structure 127
infinite loop 117