```
1  // Fig. 2.1: Welcome1.java
2  // A first program in Java
3
4  public class Welcome1 {
5     public static void main( String args[] )
6     {
7        System.out.println( "Welcome to Java Programming!" );
8     }
9  }
```

```
Welcome to Java Programming!
```

Fig. 2.1 A first program in Java.

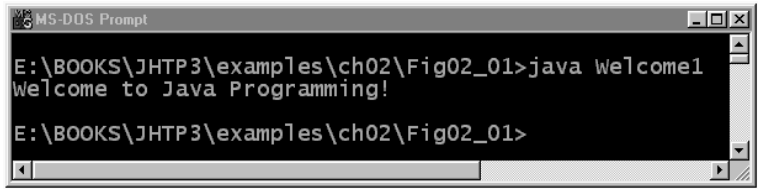

Fig. 2.2 Executing the **`Welcome1`** application in a Microsoft Windows **MS-DOS Prompt**.

```
1  // Fig. 2.3: Welcome2.java
2  // Printing a line with multiple statements
3
4  public class Welcome2 {
5     public static void main( String args[] )
6     {
7        System.out.print( "Welcome to " );
8        System.out.println( "Java Programming!" );
9     }
10 }
```

```
Welcome to Java Programming!
```

Fig. 2.3 Printing on one line with separate statements.

```
1  // Fig. 2.4: Welcome3.java
2  // Printing multiple lines with a single statement
3
4  public class Welcome3 {
5     public static void main( String args[] )
6     {
7        System.out.println( "Welcome\nto\nJava\nProgramming!" );
8     }
9  }
```

```
Welcome
to
Java
Programming!
```

Fig. 2.4 Printing on multiple lines with a single statement.

Escape sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the previous characters output on that line.
`\\`	Backslash. Used to print a backslash character.

`\"`	Double quote. Used to print a double quote character. For example,
	`System.out.println( "\"in quotes\"" );`
	displays
	`"in quotes"`

Fig. 2.5 Some common escape sequences.

```
1  // Fig. 2.6: Welcome4.java
2  // Printing multiple lines in a dialog box
3  import javax.swing.JOptionPane;  // import class JOptionPane
4
5  public class Welcome4 {
6     public static void main( String args[] )
7     {
8        JOptionPane.showMessageDialog(
9           null, "Welcome\nto\nJava\nProgramming!" );
10
11       System.exit( 0 );  // terminate the program
12    }
13 }
```

Fig. 2.6 Displaying multiple lines in a dialog box.

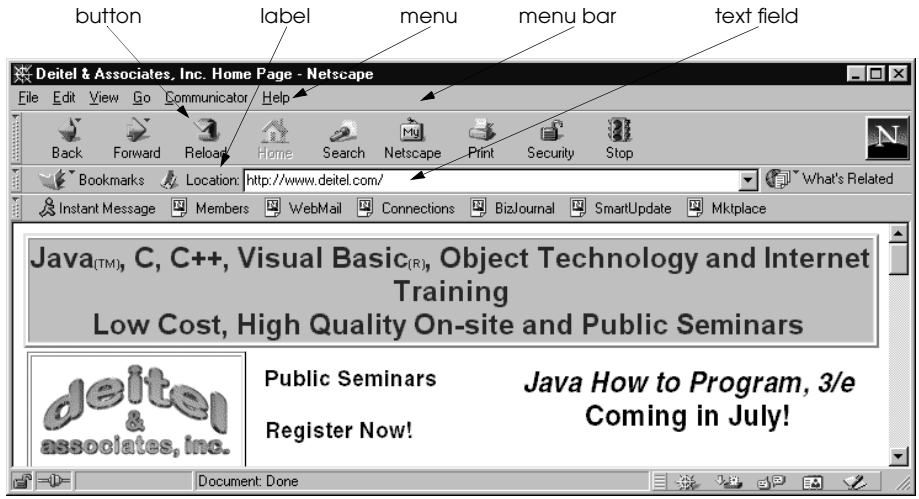

Fig. 2.7 A sample Netscape Navigator window with GUI components.

```
1  // Fig. 2.8: Addition.java
2  // An addition program
3
4  import javax.swing.JOptionPane;  // import class JOptionPane
5
6  public class Addition {
7     public static void main( String args[] )
8     {
9        String firstNumber,   // first string entered by user
10              secondNumber;  // second string entered by user
11       int number1,          // first number to add
12           number2,          // second number to add
13           sum;              // sum of number1 and number2
14
15       // read in first number from user as a string
16       firstNumber =
17          JOptionPane.showInputDialog( "Enter first integer" );
18
19       // read in second number from user as a string
20       secondNumber =
21          JOptionPane.showInputDialog( "Enter second integer" );
22
23       // convert numbers from type String to type int
24       number1 = Integer.parseInt( firstNumber );
25       number2 = Integer.parseInt( secondNumber );
26
27       // add the numbers
28       sum = number1 + number2;
29
30       // display the results
31       JOptionPane.showMessageDialog(
32          null, "The sum is " + sum, "Results",
33          JOptionPane.PLAIN_MESSAGE );
34
35       System.exit( 0 );   // terminate the program
36    }
37 }
```

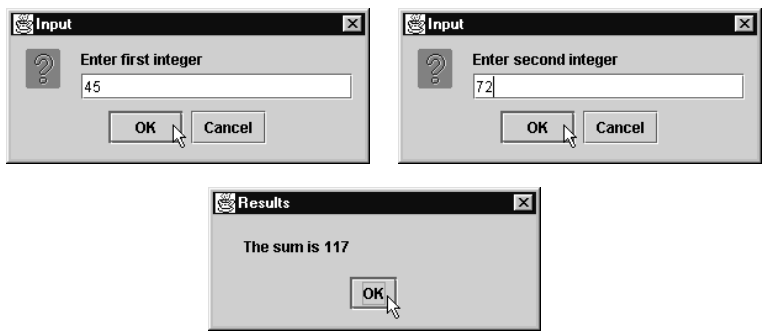

Fig. 2.8 An addition program "in action."

Message dialog type	Icon	Description
`JOptionPane.ERROR_MESSAGE`		Displays a dialog that indicates an error to the application user.
`JOptionPane.INFORMATION_MESSAGE`		Displays a dialog with an informational message to the application user—the user can simply dismiss the dialog.
`JOptionPane.WARNING_MESSAGE`		Displays a dialog that warns the application user of a potential problem.
`JOptionPane.QUESTION_MESSAGE`		Displays a dialog that poses a question to the application user. This normally requires a response such as clicking a **Yes** or **No** button.
`JOptionPane.PLAIN_MESSAGE`	no icon	Displays a dialog that simply contains a message with no icon.

Fig. 2.9 **`JOptionPane`** constants for message dialogs .

Fig. 2.10 Memory location showing the name and value of variable **number1**.

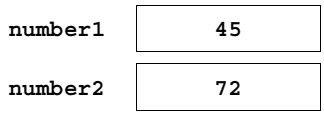

Fig. 2.11 Memory locations after values for variables **number1** and **number2** have been input.

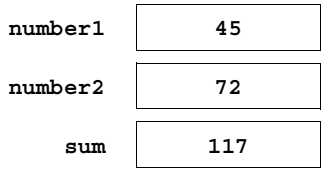

Fig. 2.12 Memory locations after a calculation.

Java operation	Arithmetic operator	Algebraic expression	Java expression
Addition	`+`	$f + 7$	`f + 7`
Subtraction	`-`	$p - c$	`p - c`
Multiplication	`*`	bm	`b * m`
Division	`/`	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Modulus	`%`	$r \bmod s$	`r % s`

Fig. 2.13 Arithmetic operators.

Operator(s)	Operation(s)	Order of evaluation (precedence)
`( )`	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/` or `%`	Multiplication Division Modulus	Evaluated second. If there are several, they are evaluated left to right.
`+` or `-`	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Fig. 2.14 Precedence of arithmetic operators.

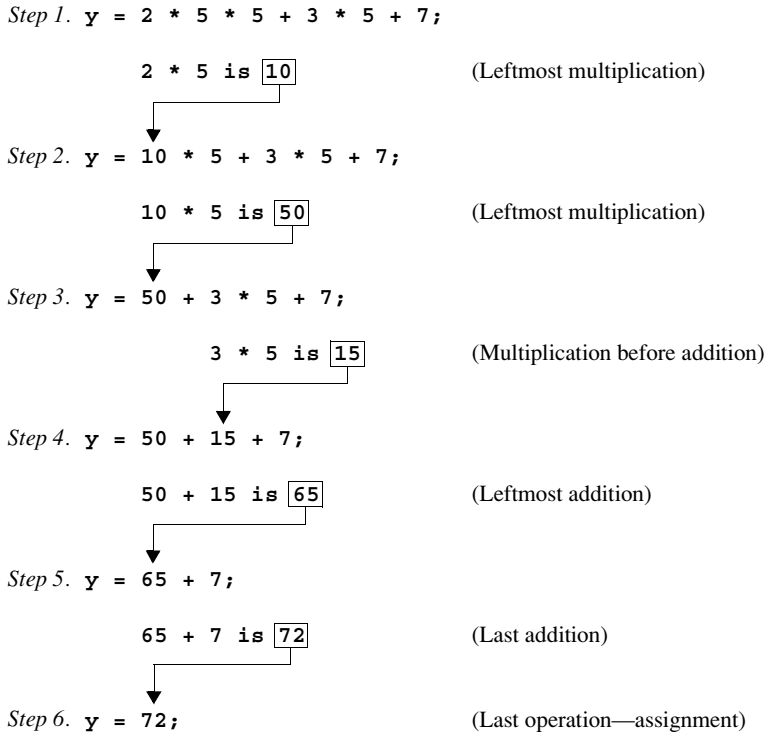

Fig. 2.15 Order in which a second-degree polynomial is evaluated.

Standard algebraic equality operator or relational operator	Java equality or relational operator	Example of Java condition	Meaning of Java condition
Equality operators			
=	`==`	`x == y`	`x` is equal to `y`
≠	`!=`	`x != y`	`x` is not equal to `y`
Relational operators			
>	`>`	`x > y`	`x` is greater than `y`
<	`<`	`x < y`	`x` is less than `y`
≥	`>=`	`x >= y`	`x` is greater than or equal to `y`
≤	`<=`	`x <= y`	`x` is less than or equal to `y`

Fig. 2.16 Equality and relational operators.

```
1  // Fig. 2.17: Comparison.java
2  // Using if statements, relational operators
3  // and equality operators
4
5  import javax.swing.JOptionPane;
6
7  public class Comparison {
8     public static void main( String args[] )
9     {
10       String firstNumber,   // first string entered by user
11              secondNumber,  // second string entered by user
12              result;        // a string containing the output
13       int number1,          // first number to compare
14           number2;          // second number to compare
15
16       // read first number from user as a string
17       firstNumber =
18          JOptionPane.showInputDialog( "Enter first integer:" );
19
20       // read second number from user as a string
21       secondNumber =
22         JOptionPane.showInputDialog( "Enter second integer:" );
23
24       // convert numbers from type String to type int
25       number1 = Integer.parseInt( firstNumber );
26       number2 = Integer.parseInt( secondNumber );
27
28       // initialize result to the empty string
29       result = "";
30
31       if ( number1 == number2 )
32          result = result + number1 + " == " + number2;
33
34       if ( number1 != number2 )
35          result = result + number1 + " != " + number2;
36
37       if ( number1 < number2 )
38          result = result + "\n" + number1 + " < " + number2;
39
40       if ( number1 > number2 )
41          result = result + "\n" + number1 + " > " + number2;
42
43       if ( number1 <= number2 )
44          result = result + "\n" + number1 + " <= " + number2;
45
46       if ( number1 >= number2 )
47          result = result + "\n" + number1 + " >= " + number2;
48
```

Fig. 2.17 Using equality and relational operators (part 1 of 2).

```
49          // Display results
50          JOptionPane.showMessageDialog(
51             null, result, "Comparison Results",
52             JOptionPane.INFORMATION_MESSAGE );
53
54          System.exit( 0 );
55       }
56    }
```

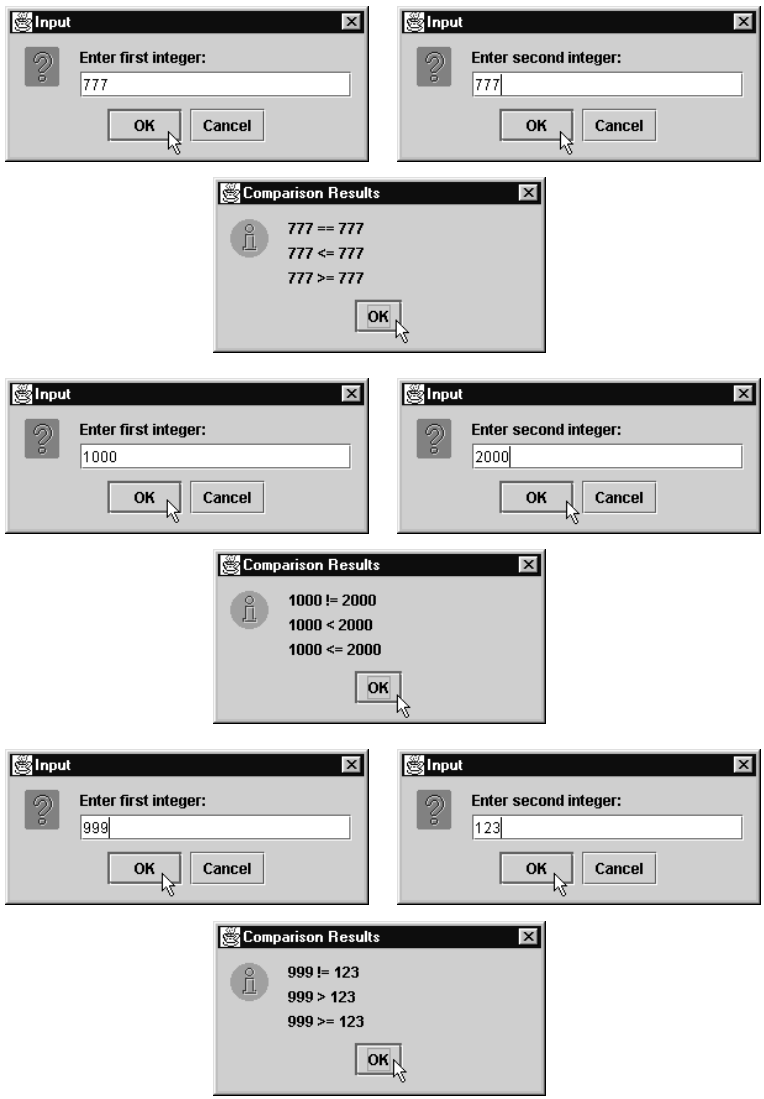

Fig. 2.17 Using equality and relational operators (part 2 of 2).

Operators	Associativity	Type
`()`	left to right	parentheses
`*  /  %`	left to right	multiplicative
`+  -`	left to right	additive
`<  <=  >  >=`	left to right	relational
`==  !=`	left to right	equality
`=`	right to left	assignment

Fig. 2.18 Precedence and associativity of the operators discussed so far.