

JAVA™

AN INTRODUCTION TO
PROBLEM SOLVING
AND PROGRAMMING

7TH EDITION

WALTER SAVITCH



Basic Computation 2

FIGURE 2.1 Primitive Type

Type Name	Kind of Value	Memory Used	Range of Values
byte	Integer	1 byte	-128 to 127
short	Integer	2 bytes	-32,768 to 32,767
int	Integer	4 bytes	-2,147,483,648 to 2,147,483,647
long	Integer	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
double	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
char	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
boolean		1 bit	True or false

LISTING 2.2 A Program with Keyboard Input

```
import java.util.Scanner;
public class EggBasket2
{
    public static void main(String[] args)
    {
        int numberOfBaskets, eggsPerBasket, totalEggs;
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter the number of eggs in each basket:");
        eggsPerBasket = keyboard.nextInt();
        System.out.println("Enter the number of baskets:");
        numberOfBaskets = keyboard.nextInt();

        totalEggs = numberOfBaskets * eggsPerBasket;

        System.out.println("If you have");
        System.out.println(eggsPerBasket + " eggs per basket and");
        System.out.println(numberOfBaskets + " baskets, then");
        System.out.println("the total number of eggs is " + totalEggs);

        System.out.println("Now we take two eggs out of each basket.");

        eggsPerBasket = eggsPerBasket - 2;
        totalEggs = numberOfBaskets * eggsPerBasket;

        System.out.println("You now have");
        System.out.println(eggsPerBasket + " eggs per basket and");
        System.out.println(numberOfBaskets + " baskets.");
        System.out.println("The new total number of eggs is " + totalEggs);
    }
}
```

Gets the Scanner class from the package (library) java.util

Sets up things so the program can accept keyboard input

Reads one whole number from the keyboard

Sample Screen Output

Enter the number of eggs in each basket:

6

Enter the number of baskets:

10

If you have

6 eggs per basket and

10 baskets, then

the total number of eggs is 60

Now we take two eggs out of each basket.

You now have

4 eggs per basket and

10 baskets.

The new total number of eggs is 40

FIGURE 2.2 Precedence Rules

Highest Precedence

First: the unary operators +, -, !, ++, and --

Second: the binary arithmetic operators *, /, and %

Third: the binary arithmetic operators + and -

Lowest Precedence

FIGURE 2.3 Some Arithmetic Expressions in Java

Ordinary Math	Java (Preferred Form)	Java (Parenthesized)
$rate^2 + delta$	<code>rate * rate + delta</code>	<code>(rate * rate) + delta</code>
$2(salary + bonus)$	<code>2 * (salary + bonus)</code>	<code>2 * (salary + bonus)</code>
$\frac{1}{time + 3mass}$	<code>1 / (time + 3 * mass)</code>	<code>1 / (time + (3 * mass))</code>
$\frac{a - 7}{t + 9v}$	<code>(a - 7) / (t + 9 * v)</code>	<code>(a - 7) / (t + (9 * v))</code>

LISTING 2.3 A Change-Making Program

```
import java.util.Scanner;

public class ChangeMaker
{
    public static void main(String[] args)
    {
        int amount, originalAmount,
            quarters, dimes, nickels, pennies;

        System.out.println("Enter a whole number from 1 to 99.");
        System.out.println("I will find a combination of coins");
        System.out.println("that equals that amount of change.");

        Scanner keyboard = new Scanner(System.in);
        amount = keyboard.nextInt();

        originalAmount = amount;
        quarters = amount / 25;
        amount = amount % 25;
        dimes = amount / 10;
        amount = amount % 10;
        nickels = amount / 5;
        amount = amount % 5;
        pennies = amount;

        System.out.println(originalAmount +
            " cents in coins can be given as:");
        System.out.println(quarters + " quarters");
        System.out.println(dimes + " dimes");
        System.out.println(nickels + " nickels and");
        System.out.println(pennies + " pennies");
    }
}
```

← 25 goes into 87 three times
← with 12 left over.

87 / 25 is 3.
87 % 25 is 12.

87 cents is three quarters
with 12 cents left over.

Sample Screen Output

```
Enter a whole number from 1 to 99.  
I will find a combination of coins  
that equals that amount of change.
```

```
87
```

```
87 cents in coins can be given as:
```

```
3 quarters
```

```
1 dimes
```

```
0 nickels and
```

```
2 pennies
```

FIGURE 2.4 String Indices

<i>Indices</i> —	0	1	2	3	4	5	6	7	8	9	10	11
	J	a	v	a		i	s		f	u	n	.

Note that the blanks and the period count as characters in the string.

FIGURE 2.5 Some Methods in the Class `String`

Method	Return Type	Example for <code>String s = "Java";</code>	Description
<code>charAt</code> (<i>index</i>)	<code>char</code>	<code>c = s.charAt(2);</code> <code>// c='v'</code>	Returns the character at <i>index</i> in the string. Index numbers begin at 0.
<code>compareTo</code> (<i>a_string</i>)	<code>int</code>	<code>i = s.compareTo("C++");</code> <code>// i is positive</code>	Compares this string with <i>a_string</i> to see which comes first in lexicographic (alphabetic, with upper before lower case) ordering. Returns a negative integer if this string is first, zero if the two strings are equal, and a positive integer if <i>a_string</i> is first.
<code>concat</code> (<i>a_string</i>)	<code>String</code>	<code>s2 = s.concat("rocks");</code> <code>// s2 = "Javarocks"</code>	Returns a new string with this string concatenated with <i>a_string</i> . You can use the <code>+</code> operator instead.
<code>equals</code> (<i>a_string</i>)	<code>boolean</code>	<code>b = s.equals("Java");</code> <code>// b = true</code>	Returns true if this string and <i>a_string</i> are equal. Otherwise returns false.
<code>equals</code> <code>IgnoreCase</code> (<i>a_string</i>)	<code>boolean</code>	<code>b = s.equals("Java");</code> <code>// b = true</code>	Returns true if this string and <i>a_string</i> are equal, considering upper and lower case versions of a letter to be the same. Otherwise returns false.
<code>indexOf</code> (<i>a_string</i>)	<code>int</code>	<code>i = s.indexOf("va");</code> <code>// i = 2</code>	Returns the index of the first occurrence of the substring <i>a_string</i> within this string or -1 if <i>a_string</i> is not found. Index numbers begin at 0.

lastIndexOf (<i>a_string</i>)	int	<pre>i = s.lastIndexOf("a"); // i = 3</pre>	Returns the index of the last occurrence of the substring <i>a_string</i> within this string or -1 if <i>a_string</i> is not found. Index numbers begin at 0.
length()	int	<pre>i = s.length(); // i = 4</pre>	Returns the length of this string.
toLowerCase()	String	<pre>s2 = s.toLowerCase(); // s = "java"</pre>	Returns a new string having the same characters as this string, but with any uppercase letters converted to lowercase. This string is unchanged.
toUpperCase()	String	<pre>s2 = s.toUpperCase(); // s2 = "JAVA"</pre>	Returns a new string having the same characters as this string, but with any lowercase letters converted to uppercase. This string is unchanged.
replace (<i>oldchar</i> , <i>newchar</i>)	String	<pre>s2 = s.replace('a','o'); // s2 = "Jovo";</pre>	Returns a new string having the same characters as this string, but with each occurrence of <i>oldchar</i> replaced by <i>newchar</i> .
substring (<i>start</i>)	String	<pre>s2 = s.substring(2); // s2 = "va";</pre>	Returns a new string having the same characters as the substring that begins at index <i>start</i> through to the end of the string. Index numbers begin at 0.
substring (<i>start</i> , <i>end</i>)	String	<pre>s2 = s.substring(1,3); // s2 = "av";</pre>	Returns a new string having the same characters as the substring that begins at index <i>start</i> through to but not including the character at index <i>end</i> . Index numbers begin at 0.
trim()	String	<pre>s = " Java "; s2 = s.trim(); // s2 = "Java"</pre>	Returns a new string having the same characters as this string, but with leading and trailing whitespace removed.

LISTING 2.4 Using the String Class

```
public class StringDemo
{
    public static void main(String[] args)
    {
        String sentence = "Text processing is hard!";
        int position = sentence.indexOf("hard");
        System.out.println(sentence);
        System.out.println("012345678901234567890123");
        System.out.println("The word \"hard\" starts at index "
            + position);
        sentence = sentence.substring(0, position) + "easy!";
        sentence = sentence.toUpperCase();
        System.out.println("The changed string is:");
        System.out.println(sentence);
    }
}
```

The meaning of \" is discussed in the section entitled \"Escape Characters.\"

Screen Output

```
Text processing is hard!
012345678901234567890123
The word "hard" starts at index 19
The changed string is:
TEXT PROCESSING IS EASY!
```

LISTING 2.5 A Demonstration of Keyboard Input (part 1 of 2)

```
import java.util.Scanner; ← Gets the Scanner
                           class from the package
                           (library) java.util

public class ScannerDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in); ← Sets things up
                                                    so the program
                                                    can accept
                                                    keyboard input

        System.out.println("Enter two whole numbers");
        System.out.println("separated by one or more spaces:");

        int n1, n2;
        n1 = keyboard.nextInt(); ← Reads one int value
                                 from the keyboard
        n2 = keyboard.nextInt();
        System.out.println("You entered " + n1 + " and " + n2);

        System.out.println("Next enter two numbers.");
        System.out.println("A decimal point is OK.");

        double d1, d2;
        d1 = keyboard.nextDouble(); ← Reads one double
                                     value from the keyboard
        d2 = keyboard.nextDouble();
        System.out.println("You entered " + d1 + " and " + d2);

        System.out.println("Next enter two words:");

        String s1, s2;
        s1 = keyboard.next(); ← Reads one word from
                              the keyboard
        s2 = keyboard.next();
        System.out.println("You entered \"" +
                           s1 + "\" and \"" + s2 + "\"");

        s1 = keyboard.nextLine(); //To get rid of '\n' ← This line is explained in
                                                                    the next Gotcha section.

        System.out.println("Next enter a line of text:");
        s1 = keyboard.nextLine(); ← Reads an entire line
        System.out.println("You entered: \"" + s1 + "\"");
    }
}
```

(continued)

LISTING 2.5 A Demonstration of Keyboard Input *(part 2 of 2)*

Sample Screen Output

Enter two whole numbers
separated by one or more spaces:

42 43

You entered 42 and 43
Next enter two numbers.
A decimal point is OK.

9.99 21

You entered 9.99 and 21.0
Next enter two words:

plastic spoons

You entered "plastic" and "spoons"
Next enter a line of text:

May the hair on your toes grow long and curly.

You entered "May the hair on your toes grow long and curly."

FIGURE 2.7 Some Methods in the Class `Scanner`

Method for <code>Scanner kbd;</code>	Return Type	Description
<code>next()</code>	<code>String</code>	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.
<code>nextLine()</code>	<code>String</code>	Reads the rest of the current keyboard input line and returns the characters read as a value of type <code>String</code> . Note that the line terminator <code>'\n'</code> is read and discarded; it is not included in the string returned
<code>nextInt()</code>	<code>int</code>	Returns the next keyboard input as a value of type <code>int</code> .
<code>nextDouble()</code>	<code>double</code>	Returns the next keyboard input as a value of type <code>double</code> .
<code>nextFloat()</code>	<code>float</code>	Returns the next keyboard input as a value of type <code>float</code> .
<code>nextLong()</code>	<code>long</code>	Returns the next keyboard input as a value of type <code>long</code> .
<code>nextByte()</code>	<code>byte</code>	Returns the next keyboard input as a value of type <code>byte</code> .
<code>nextShort()</code>	<code>short</code>	Returns the next keyboard input as a value of type <code>short</code> .
<code>nextBoolean()</code>	<code>boolean</code>	Returns the next keyboard input as a value of type <code>boolean</code> . The values of <code>true</code> and <code>false</code> are entered as the words <code>true</code> and <code>false</code> . Any combination of uppercase and lowercase letters is allowed in spelling <code>true</code> and <code>false</code> .
<code>useDelimiter (<i>Delimiter_Word</i>)</code>	<code>Scanner</code>	Makes the string <i>Delimiter_Word</i> 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 <i>Delimiter_Word</i> . This is a simple case of the use of the <code>useDelimiter</code> 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.

LISTING 2.6 Changing Delimiters (Optional)

```
import java.util.Scanner;

public class DelimitersDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard1 = new Scanner(System.in);
        Scanner keyboard2 = new Scanner(System.in);
        keyboard2.useDelimiter("##");
        //The delimiters for keyboard1 are the whitespace
        //characters.
        //The only delimiter for keyboard2 is ##.

        String s1, s2;

        System.out.println("Enter a line of text with two words:");
        s1 = keyboard1.next();
        s2 = keyboard1.next();
        System.out.println("The two words are \"" + s1 +
            "\" and \"" + s2 + "\"");

        System.out.println("Enter a line of text with two words");
        System.out.println("delimited by ##:");
        s1 = keyboard2.next();
        s2 = keyboard2.next();
        System.out.println("The two words are \"" + s1 +
            "\" and \"" + s2 + "\"");
    }
}
```

keyboard1 and keyboard2 have different delimiters.

Sample Screen Output

```
Enter a line of text with two words:
```

```
funny wo##rd##
```

```
The two words are "funny" and "wor##rd##"
```

```
Enter a line of text with two words  
delimited by ##:
```

```
funny wor##rd##
```

```
The two words are "funny wo" and "rd"
```

FIGURE 2.8 Selected Format Specifiers for `System.out.printf`

Format Specifier	Type of Output	Examples
<code>%c</code>	Character	A single character: <code>%c</code>
		A single character in a field of two spaces: <code>%2c</code>
<code>%d</code>	Decimal integer number	An integer: <code>%d</code>
		An integer in a field of 5 spaces: <code>%5d</code>
<code>%f</code>	Floating-point number	A floating-point number: <code>%f</code>
		A floating-point number with 2 digits after the decimal: <code>%1.2f</code>
		A floating-point number with 2 digits after the decimal in a field of 6 spaces: <code>%6.2f</code>
<code>%e</code>	Exponential floating-point number	A floating-point number in exponential format: <code>%e</code>
<code>%s</code>	String	A string formatted to a field of 10 spaces: <code>%10s</code>

LISTING 2.7 Comments and Indentation

```
import java.util.Scanner;
```

This import can go after the big comment if you prefer.

```
/**  
 * Program to compute area of a circle.  
 * Author: Jane Q. Programmer.  
 * E-mail Address: janeq@somemachine.etc.etc.  
 * Programming Assignment 2.  
 * Last Changed: October 7, 2008.  
 */
```

```
public class CircleCalculation
```

```
{
```

The vertical lines indicate the indenting pattern.

```
    public static void main(String[] args)
```

```
    {
```

```
        double radius; //in inches
```

```
        double area; //in square inches
```

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

```
        System.out.println("Enter the radius of a circle in inches:");
```

```
        radius = keyboard.nextDouble();
```

```
        area = 3.14159 * radius * radius;
```

```
        System.out.println("A circle of radius " + radius + " inches");
```

```
        System.out.println("has an area of " + area + " square inches.");
```

```
    }
```

```
}
```

Later in this chapter, we will give an improved version of this program.

Sample Screen Output

```
Enter the radius of a circle in inches:
```

```
2.5
```

```
A circle of radius 2.5 inches  
has an area of 19.6349375 square inches.
```

LISTING 2.8 Naming a Constant

```
import java.util.Scanner;

/**
 * Program to compute area of a circle.
 * Author: Jane Q. Programmer.
 * E-mail Address: janeq@somemachine.etc.etc.
 * Programming Assignment 2.
 * Last Changed: October 7, 2008.
 */

public class CircleCalculation2
{
    public static final double PI = 3.14159;

    public static void main(String[] args)
    {
        double radius; //in inches
        double area; //in square inches
        Scanner keyboard = new Scanner(System.in);

        System.out.println("Enter the radius of a circle in inches:");
        radius = keyboard.nextDouble();
        area = PI * radius * radius;
        System.out.println("A circle of radius " + radius + " inches");
        System.out.println("has an area of " + area + " square inches.");
    }
}
```

Although it would not be as clear, it is legal to place the definition of **PI** here instead.

Sample Screen Output

```
Enter the radius of a circle in inches:
```

```
2.5
```

```
A circle of radius 2.5 inches  
has an area of 19.6349375 square inches.
```

LISTING 2.9 Revision of Listing 1.2 Using Comments and Named Constants

```
import javax.swing.JApplet;  
import java.awt.Graphics;
```

← These can go after the big comment if you prefer

```
/**  
 * Applet that displays a happy face.  
 * Author: Jane Q. Programmer.  
 * Revision of Listing 1.2.  
 */  
public class HappyFace extends JApplet  
{  
    public static final int FACE_DIAMETER = 200;  
    public static final int X_FACE = 100;  
    public static final int Y_FACE = 50;  
  
    public static final int EYE_WIDTH = 10;  
    public static final int EYE_HEIGHT = 20;  
    public static final int X_RIGHT_EYE = 155;  
    public static final int Y_RIGHT_EYE = 100;  
    public static final int X_LEFT_EYE = 230;  
    public static final int Y_LEFT_EYE = Y_RIGHT_EYE;  
  
    public static final int MOUTH_WIDTH = 100;  
    public static final int MOUTH_HEIGHT = 50;  
    public static final int X_MOUTH = 150;  
    public static final int Y_MOUTH = 160;  
    public static final int MOUTH_START_ANGLE = 180;  
    public static final int MOUTH_EXTENT_ANGLE = 180;
```

```
public void paint(Graphics canvas)
{
    super.paint(canvas);
    //Draw face outline:
    canvas.drawOval(X_FACE, Y_FACE, FACE_DIAMETER, FACE_DIAMETER);
    //Draw eyes:
    canvas.fillOval(X_RIGHT_EYE, Y_RIGHT_EYE, EYE_WIDTH, EYE_HEIGHT);
    canvas.fillOval(X_LEFT_EYE, Y_LEFT_EYE, EYE_WIDTH, EYE_HEIGHT);
    //Draw mouth:
    canvas.drawArc(X_MOUTH, Y_MOUTH, MOUTH_WIDTH, MOUTH_HEIGHT,
                  MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE);
}
}
```

*The applet drawing is the same as
the one shown in Listing 1.2.*

LISTING 2.10 A Java GUI Application using the JFrame Class

```
import javax.swing.JFrame;
import java.awt.Graphics;

public class HappyFaceJFrame extends JFrame
{
    public static final int FACE_DIAMETER = 200;
    public static final int X_FACE = 100;
    public static final int Y_FACE = 50;

    public static final int EYE_WIDTH = 10;
    public static final int EYE_HEIGHT = 20;
    public static final int X_RIGHT_EYE = 155;
    public static final int Y_RIGHT_EYE = 100;
    public static final int X_LEFT_EYE = 230;
    public static final int Y_LEFT_EYE = Y_RIGHT_EYE;

    public static final int MOUTH_WIDTH = 100;
    public static final int MOUTH_HEIGHT = 50;
    public static final int X_MOUTH = 150;
    public static final int Y_MOUTH = 160;
    public static final int MOUTH_START_ANGLE = 180;
    public static final int MOUTH_DEGREES_SHOWN = 180;
```

```

public void paint(Graphics canvas)
{
    super.paint(canvas);
    //Draw face outline:
    canvas.drawOval(X_FACE, Y_FACE, FACE_DIAMETER, FACE_DIAMETER);
    //Draw eyes:
    canvas.fillOval(X_RIGHT_EYE, Y_RIGHT_EYE, EYE_WIDTH, EYE_HEIGHT);
    canvas.fillOval(X_LEFT_EYE, Y_LEFT_EYE, EYE_WIDTH, EYE_HEIGHT);
    //Draw mouth:
    canvas.drawArc(X_MOUTH, Y_MOUTH, MOUTH_WIDTH, MOUTH_HEIGHT,
                  MOUTH_START_ANGLE, MOUTH_DEGREES_SHOWN);
}

public HappyFaceJFrame()
{
    setSize(600,400);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
}
public static void main(String[] args)
{
    HappyFaceJFrame guiWindow = new HappyFaceJFrame();
    guiWindow.setVisible(true);
}
}

```

This application draws the same Happy Face image as the applet in Listing 2.9

LISTING 2.11 Program Using JOptionPane for I/O (part 1 of 2)

```
import javax.swing.JOptionPane;

public class JOptionPaneDemo
{
    public static void main(String[] args)
    {
        String appleString =
            JOptionPane.showInputDialog("Enter number of apples:");
        int appleCount = Integer.parseInt(appleString);

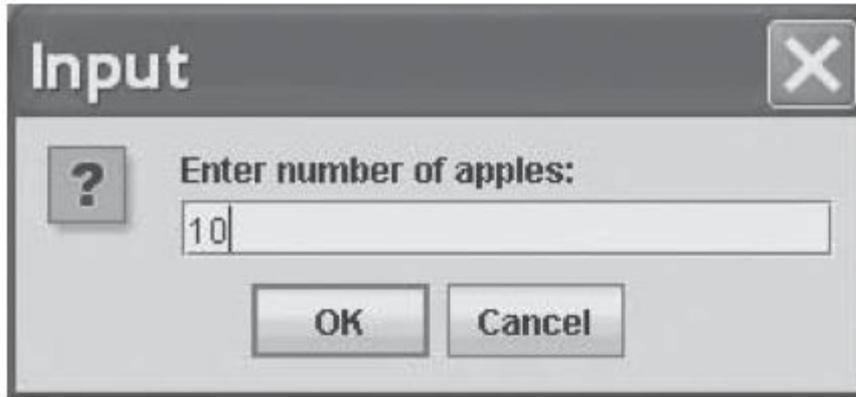
        String orangeString =
            JOptionPane.showInputDialog("Enter number of oranges:");
        int orangeCount = Integer.parseInt(orangeString);

        int totalFruitCount = appleCount + orangeCount;

        JOptionPane.showMessageDialog(null,
            "The total number of fruits = " + totalFruitCount);

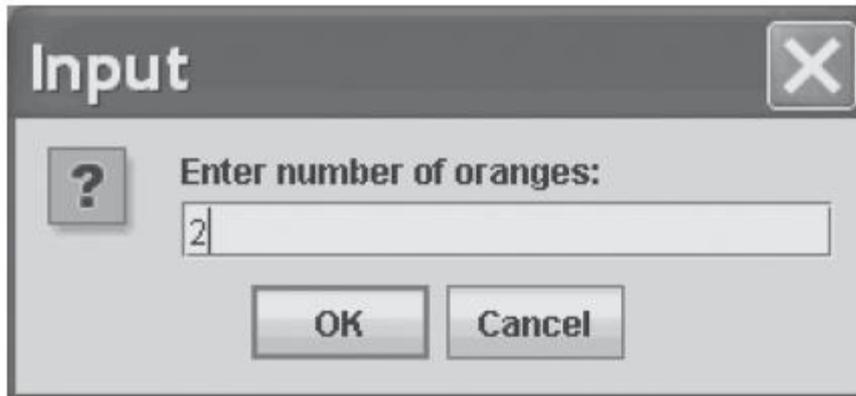
        System.exit(0);
    }
}
```

Dialog 1



When the user clicks OK, the window goes away and the next window (if any) is displayed.

Dialog 2



(continued)

LISTING 2.11 Program Using JOptionPane for I/O (part 2 of 2)

Dialog 3

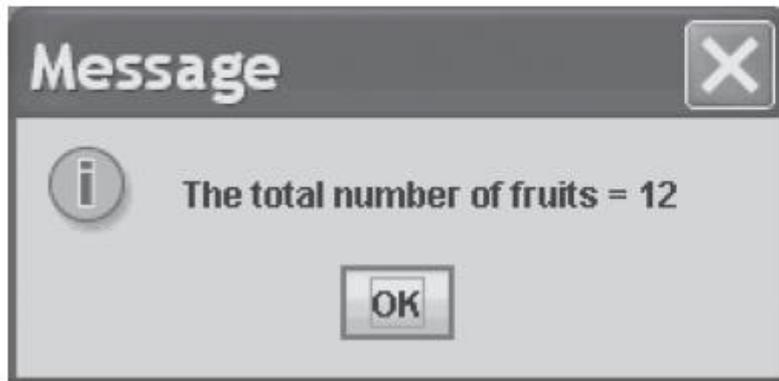


FIGURE 2.9 Methods for Converting Strings to Numbers

Result Type	Method for Converting
byte	Byte.parseInt(<i>String_To_Convert</i>)
short	Short.parseInt(<i>String_To_Convert</i>)
int	Integer.parseInt(<i>String_To_Convert</i>)
long	Long.parseLong(<i>String_To_Convert</i>)
float	Float.parseFloat(<i>String_To_Convert</i>)
double	Double.parseDouble(<i>String_To_Convert</i>)

LISTING 2.12 A Change-Making Program with Windows for I/O (part 1 of 2)

```
import javax.swing.JOptionPane;
public class ChangeMakerWindow
{
    public static void main(String[] args)
    {
        String amountString = JOptionPane.showInputDialog(
            "Enter a whole number from 1 to 99.\n" +
            "I will output a combination of coins\n" +
            "that equals that amount of change.");

        int amount, originalAmount,
            quarters, dimes, nickels, pennies;
        amount = Integer.parseInt(amountString);
        originalAmount = amount;

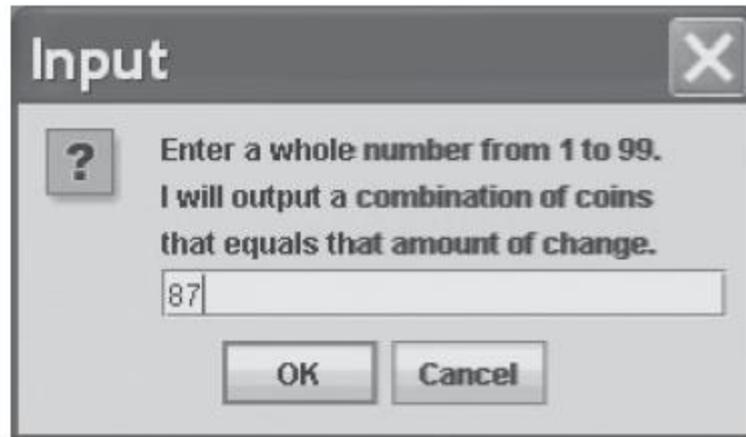
        quarters = amount / 25;
        amount = amount % 25;
        dimes = amount / 10;
        amount = amount % 10;
        nickels = amount / 5;
        amount = amount % 5;
        pennies = amount;

        JOptionPane.showMessageDialog(null, originalAmount +
            " cents in coins can be given as:\n" +
            quarters + " quarters\n" +
            dimes + " dimes\n" +
            nickels + " nickels and\n" +
            pennies + " pennies");

        System.exit(0);
    }
}
```

← Do not forget that you need **System.exit** in a program with input or output windows.

Input Dialog



Output Dialog

