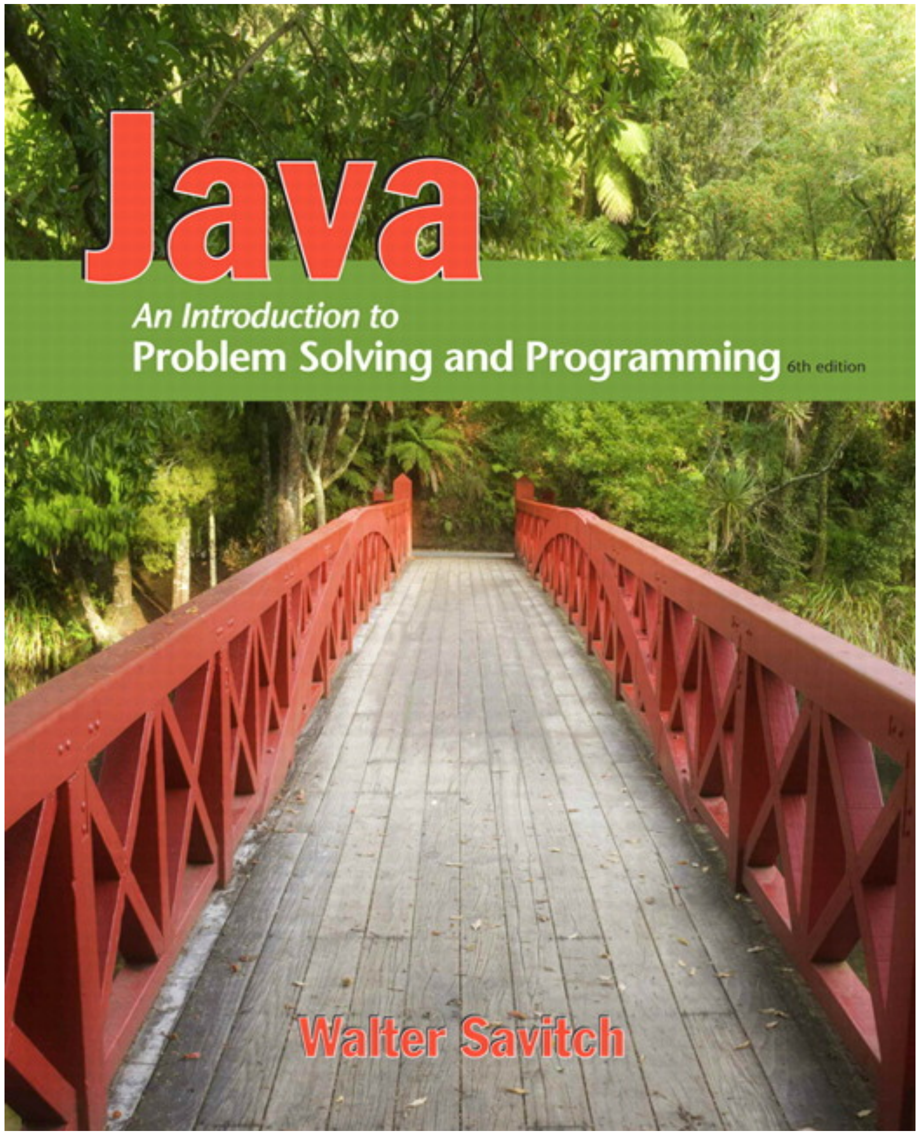


Basic Computation 2

LISTING 2.1 A Simple Java Program

```
public class EggBasket
{
   public static void main(String[] args)
   {
      int numberOfBaskets, eggsPerBasket, totalEggs;

      numberOfBaskets = 10;
      eggsPerBasket = 6;

      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);
   }
}
```

Variable declarations

Assignment statement

Sample Screen Output

```
If you have
6 eggs per basket and
10 baskets, then
the total number of eggs is 60
```

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	–9,223,372,036,8547,75,808 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)
$rate^2 + delta$	`rate * rate + delta`	`(rate * rate) + delta`
$2(salary + bonus)$	`2 * (salary + bonus)`	`2 * (salary + bonus)`
$\frac{1}{time + 3mass}$	`1 / (time + 3 * mass)`	`1 / (time + (3 * mass))`
$\frac{a - 7}{t + 9v}$	`(a - 7) / (t + 9 * v)`	`(a - 7) / (t + (9 * v))`

LISTING 2.3 A Change-Making Program

```java
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.

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

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

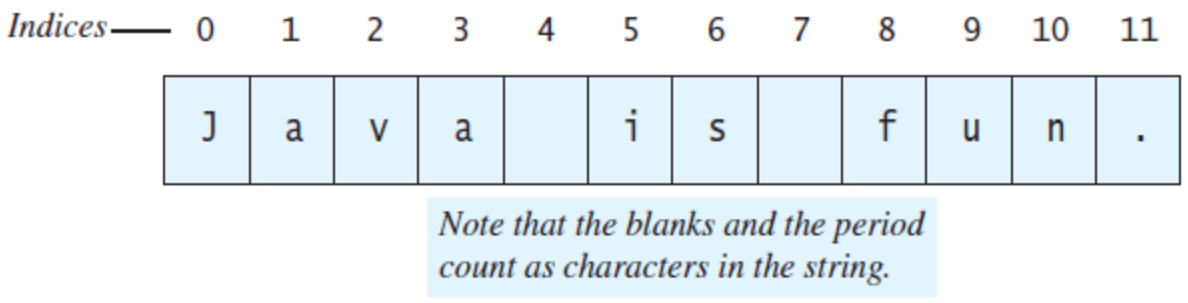

Indices
0
1
2
3
4
5
6
7
8
9
10
11
J
a
v
a
i
s
f
u
n
.
Note that the blanks and the period count as characters in the string.

FIGURE 2.5 Some Methods in the Class `String`

`charAt` (*Index*)	Returns the character at *Index* in this string. Index numbers begin at 0.
`compareTo`(*A_String*)	Compares this string with *A_String* to see which string comes first in the lexicographic ordering. (Lexicographic ordering is the same as alphabetical ordering when both strings are either all uppercase letters or all lowercase letters.) Returns a negative integer if this string is first, returns zero if the two strings are equal, and returns a positive integer if *A_String* is first.
`concat`(*A_String*)	Returns a new string having the same characters as this string concatenated with the characters in *A_String*. You can use the + operator instead of `concat`.
`equals`(*Other_String*)	Returns true if this string and *Other_String* are equal. Otherwise, returns false.
`equalsIgnoreCase`(*Other_String*)	Behaves like the method `equals`, but considers uppercase and lowercase versions of a letter to be the same.
`indexOf`(*A_String*)	Returns the index of the first occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.
`lastIndexOf`(*A_String*)	Returns the index of the last occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.
`length()`	Returns the length of this string.

`toLowerCase()`

Returns a new string having the same characters as this string, but with any uppercase letters converted to lowercase.

`toUpperCase()`

Returns a new string having the same characters as this string, but with any lowercase letters converted to uppercase.

`replace(`*OldChar*`,` *NewChar*`)`

Returns a new string having the same characters as this string, but with each occurrence of *OldChar* replaced by *NewChar*.

`substring(`*Start*`)`

Returns a new string having the same characters as the substring that begins at index *Start* of this string through to the end of the string. Index numbers begin at 0.

`substring(`*Start*`,`*End*`)`

Returns a new string having the same characters as the substring that begins at index *Start* of this string through, but not including, index *End* of the string. Index numbers begin at 0.

`trim()`

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

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

```java
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);
    }
}
```

Screen Output

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

FIGURE 2.6 **Escape Characters**

`\"`	Double quote.
`\'`	Single quote.
`\\`	Backslash.
`\n`	New line. Go to the beginning of the next line.
`\r`	Carriage return. Go to the beginning of the current line.
`\t`	Tab. Add whitespace up to the next tab stop.

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

```
import java.util.Scanner;

public class ScannerDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);

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

        int n1, n2;
        n1 = keyboard.nextInt();
        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();
        d2 = keyboard.nextDouble();
        System.out.println("You entered " + d1 + " and " + d2);

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

        String s1, s2;
        s1 = keyboard.next();
        s2 = keyboard.next();
        System.out.println("You entered \"" +
                           s1 + "\" and \"" + s2 + "\"");

        s1 = keyboard.nextLine(); //To get rid of '\n'

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

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

Sets things up so the program can accept keyboard input

Reads one `int` *value from the keyboard*

Reads one `double` *value from the keyboard*

Reads one word from the keyboard

This line is explained in the next Gotcha section.

Reads an entire line

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`

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.

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

LISTING 2.7 Comments and Indentation

```
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 CircleCalculation
{
    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.");
    }
}
```

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

The vertical lines indicate the indenting pattern.

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)
    {
        //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

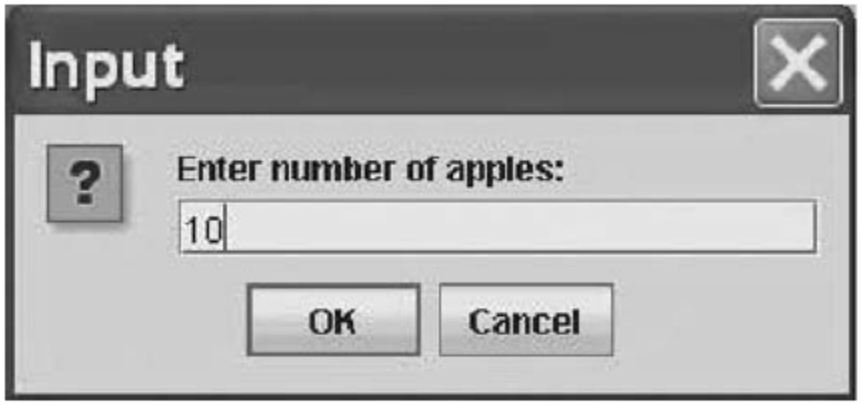
```java
import javax.swing.JFrame;
import java.awt.Graphics;

public class HappyFaceJFrame extends JFrame
{
    public static finalint 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)
    {
        //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

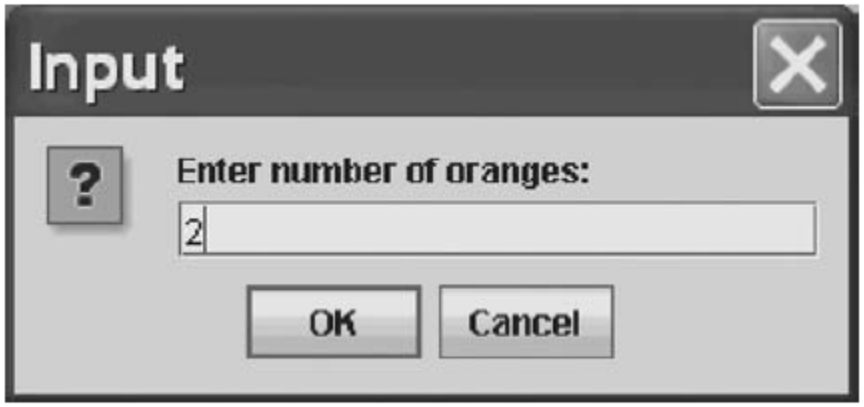


Enter number of apples:

10

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

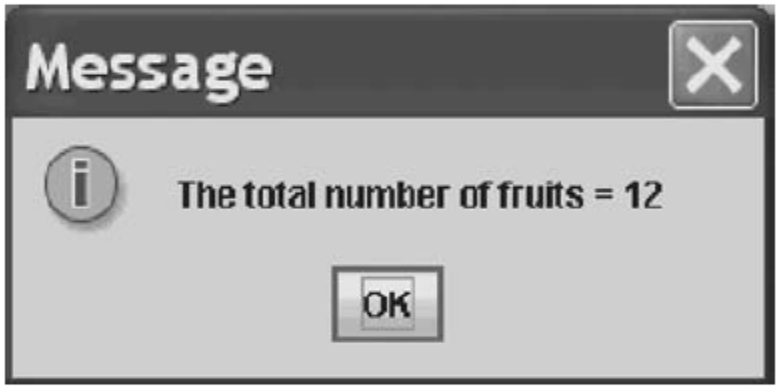
Dialog 2



Enter number of oranges:

2

Dialog 3

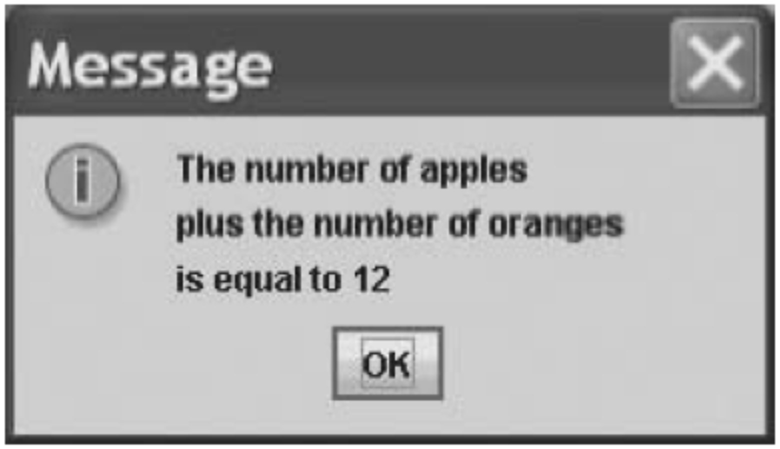


```
The total number of fruits = 12
```

FIGURE 2.9 Methods for Converting Strings to Numbers

Result Type	Method for Converting
`byte`	`Byte.parseByte(`*String_To_Convert*`)`
`short`	`Short.parseShort(`*String_To_Convert*`)`
`int`	`Integer.parseInt(`*String_To_Convert*`)`
`long`	`Long.parseLong(`*String_To_Convert*`)`
`float`	`Float.parseFloat(`*String_To_Convert*`)`
`double`	`Double.parseDouble(`*String_To_Convert*`)`

FIGURE 2.10 **A Dialog Window Containing Multiline Output**



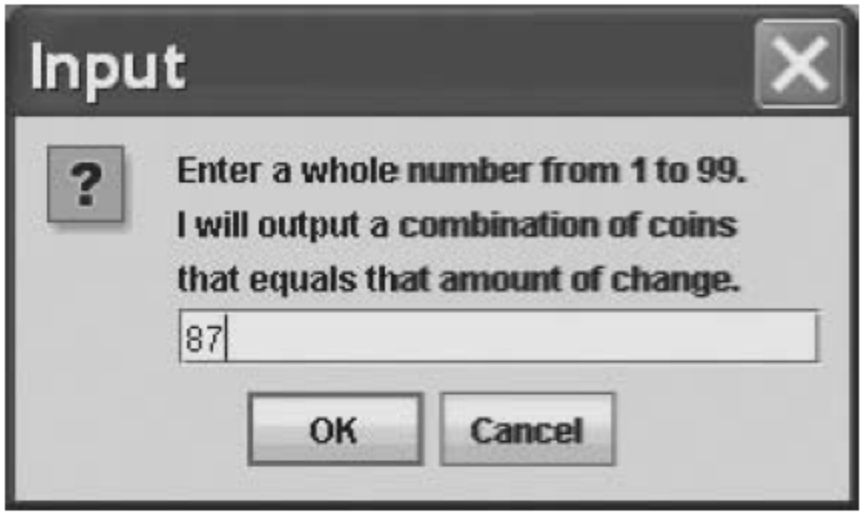
```
The number of apples
plus the number of oranges
is equal to 12
```

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.*

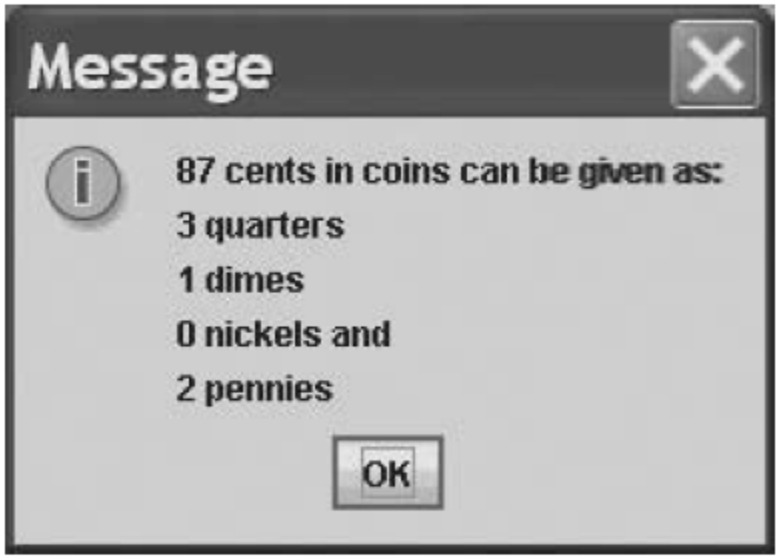
Input Dialog



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

87

Output Dialog



```
87 cents in coins can be given as:
3 quarters
1 dimes
0 nickels and
2 pennies
```