FIRST PRINTING UPDATE

to

java 6TH EDITION SOFTWARE SOLUTIONS

foundations of program design

LEWIS & LOFTUS

Please note that several key Java reserve words were accidentally omitted from some of the programming code listings in this book. You may refer to the corrected pages 34, 66, 67, 68, 90, 96, and 102 in this supplemental booklet for the complete and correct Java code found in Listings 1.2, 2.5, 2.6, 2.9, 2.10, and 2.11, disregarding these same numbered pages as they appear in the bound text. These errors of omission were introduced as part of the composition process during production of this textbook. The authors were in no way responsible for these errors, and the responsibility lies solely with the publisher. We at Addison-Wesley apologize for the inconvenience this has caused you, the reader.

PEARSON Addison Wesley

ISBN-13: 978-0-13-609490-6
ISBN-10: 0-13-609490-2

EAN 9 780136 094906 90000

KEY CONCEPT
Identifier names should be descriptive and readable.

`curStVal` is a good name to represent the current stock value, but another person trying to understand the code may have trouble figuring out what you meant. It might not even be clear to you two months after writing it.

A *name* in Java is a series of identifiers separated by the dot (period) character. The name `System.out` is the way we designate the object through which we invoked the `println` method. Names appear quite regularly in Java programs.

White Space

All Java programs use *white space* to separate the words and symbols used in a program. White space consists of blanks, tabs, and newline characters. The phrase white space refers to the fact that, on a white sheet of paper with black printing, the space between the words and symbols is white. The way a programmer uses white space is important because it can be used to emphasize parts of the code and can make a program easier to read.

KEY CONCEPT
Appropriate use of white space makes a program easier to read and understand.

Except when it's used to separate words, the computer ignores white space. It does not affect the execution of a program. This fact gives programmers a great deal of flexibility in how they format a program. The lines of a program should be divided in logical places and certain lines should be indented and aligned so that the program's underlying structure is clear.

Because white space is ignored, we can write a program in many different ways. For example, taking white space to one extreme, we could put as many words as possible on each line. The code in Listing 1.2, the `Lincoln2` program, is formatted quite differently from `Lincoln` but prints the same message.

LISTING 1.2

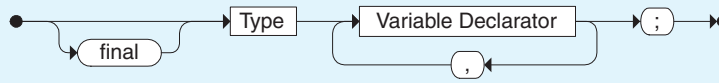
```
//********************************************************************
//  Lincoln2.java       Author: Lewis/Loftus
//
//  Demonstrates a poorly formatted, though valid, program.
//********************************************************************

public class Lincoln2{public static void main(String[]args){
System.out.println("A quote by Abraham Lincoln:");
System.out.println("Whatever you are, be a good one.");}}
```

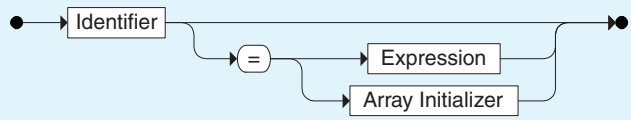
OUTPUT

```
A quote by Abraham Lincoln:
Whatever you are, be a good one.
```

Local Variable Declaration



Variable Declarator



A variable declaration consists of a Type followed by a list of variables. Each variable can be initialized in the declaration to the value of the specified Expression. If the final modifier precedes the declaration, the identifiers are declared as named constants whose values cannot be changed once set.

Examples:

```
int total;
double num1, num2 = 4.356, num3;
char letter = 'A', digit = '7';
final int MAX = 45;
```

LISTING 2.5

```
//********************************************************************
//  PianoKeys.java       Author: Lewis/Loftus
//
//  Demonstrates the declaration, initialization, and use of an
//  integer variable.
//********************************************************************

public class PianoKeys
{
   //-----------------------------------------------------------------
   //  Prints the number of keys on a piano.
   //-----------------------------------------------------------------
   public static void main (String[] args)
```

LISTING 2.5 *continued*

```
   {
      int keys = 88;

      System.out.println ("A piano has " + keys + " keys.");
   }
}
```

OUTPUT

```
A piano has 88 keys.
```

In the `PianoKeys` program, two pieces of information are used in the call to the `println` method. The first is a string, and the second is the variable `keys`. When a variable is referenced, the value currently stored in it is used. Therefore, when the call to `println` is executed, the value of `keys`, which is 88, is obtained. Because that value is an integer, it is automatically converted to a string and concatenated with the initial string. The concatenated string is passed to `println` and printed.

A variable declaration can have multiple variables of the same type declared on one line. Each variable on the line can be declared with or without an initializing value. For example:

```
int count, minimum = 0, result;
```

The Assignment Statement

Let's examine a program that changes the value of a variable. Listing 2.6 shows a program called `Geometry`. This program first declares an integer variable called `sides` and initializes it to `7`. It then prints out the current value of `sides`.

The next line in `main` changes the value stored in the variable `sides`:

```
sides = 10;
```

This is called an *assignment statement* because it assigns a value to a variable. When executed, the expression on the right-hand side of the assignment operator (=) is evaluated, and the result is stored in the memory location indicated by the variable on the left-hand side. In this example, the expression is simply a number, `10`. We discuss expressions that are more involved than this in the next section.

LISTING 2.6

```
//********************************************************************
//  Geometry.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an assignment statement to change the
//  value stored in a variable.
//********************************************************************

public class Geometry
{
   //-----------------------------------------------------------------
   //  Prints the number of sides of several geometric shapes.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int sides = 7;  // declaration with initialization
      System.out.println ("A heptagon has " + sides + " sides.");

      sides = 10;  // assignment statement
      System.out.println ("A decagon has " + sides + " sides.");

      sides = 12;
      System.out.println ("A dodecagon has " + sides + " sides.");
   }
}
```

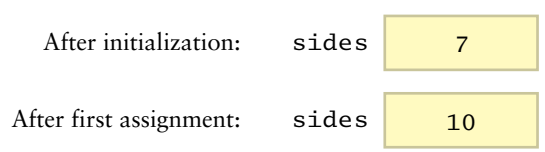
OUTPUT

```
A heptagon has 7 sides.
A decagon has 10 sides.
A dodecagon has 12 sides.
```

KEY CONCEPT

Accessing data leaves it intact in memory, but an assignment statement overwrites the old data.

A variable can store only one value of its declared type. A new value overwrites the old one. In this case, when the value `10` is assigned to `sides`, the original value `7` is overwritten and lost forever, as follows:



of the `java.util` class library. The use of the `import` statement is discussed further in Chapter 3.

Various `Scanner` methods such as `nextInt` and `nextDouble` are provided to read data of particular types. The `GasMileage` program, shown in Listing 2.9,

LISTING 2.9

```
//********************************************************************
//  GasMileage.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the Scanner class to read numeric data.
//********************************************************************

import java.util.Scanner;

public class GasMileage
{
   //-----------------------------------------------------------------
   //  Calculates fuel efficiency based on values entered by the
   //  user.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int miles;
      double gallons, mpg;

      Scanner scan =  Scanner (System.in);

      System.out.print ("Enter the number of miles: ");
      miles = scan.nextInt();

      System.out.print ("Enter the gallons of fuel used: ");
      gallons = scan.nextDouble();

      mpg = miles / gallons;

      System.out.println ("Miles Per Gallon: " + mpg);
   }
}
```

OUTPUT

```
Enter the number of miles: 328
Enter the gallons of fuel used: 11.2
Miles Per Gallon: 29.28571428571429
```

LISTING 2.10

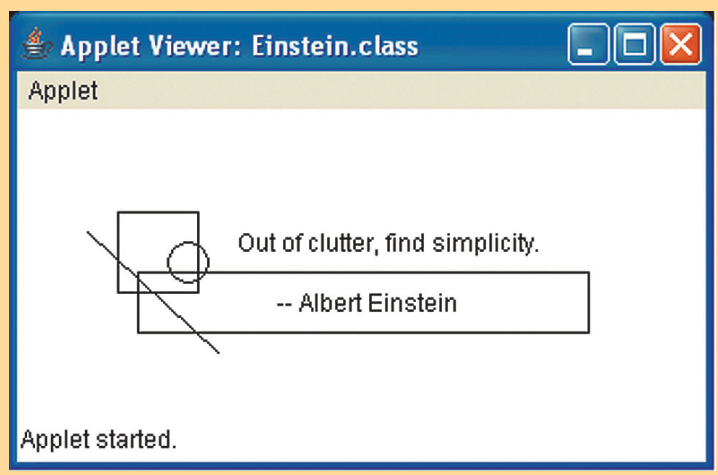
```
//********************************************************************
//  Einstein.java       Author: Lewis/Loftus
//
//  Demonstrates a basic applet.
//********************************************************************

import javax.swing.JApplet;
import java.awt.*;

public class Einstein extends JApplet
{
   //-----------------------------------------------------------------
   //  Draws a quotation by Albert Einstein among some shapes.
   //-----------------------------------------------------------------
   public void paint (Graphics page)
   {
      page.drawRect (50, 50, 40, 40);    // square
      page.drawRect (60, 80, 225, 30);   // rectangle
      page.drawOval (75, 65, 20, 20);    // circle
      page.drawLine (35, 60, 100, 120);  // line

      page.drawString ("Out of clutter, find simplicity.", 110, 70);
      page.drawString ("-- Albert Einstein", 130, 100);
   }
}
```

DISPLAY



LISTING 2.11

```
//********************************************************************
//  Snowman.java       Author: Lewis/Loftus
//
//  Demonstrates basic drawing methods and the use of color.
//********************************************************************

import javax.swing.JApplet;
import java.awt.*;

public class Snowman extends JApplet
{
   //-----------------------------------------------------------------
   //  Draws a snowman.
   //-----------------------------------------------------------------
   public void paint (Graphics page)
   {
      final int MID = 150;
      final int TOP = 50;

      setBackground (Color.cyan);

      page.setColor (Color.blue);
      page.fillRect (0, 175, 300, 50);  // ground

      page.setColor (Color.yellow);
      page.fillOval (-40, -40, 80, 80);  // sun

      page.setColor (Color.white);
      page.fillOval (MID-20, TOP, 40, 40);      // head
      page.fillOval (MID-35, TOP+35, 70, 50);   // upper torso
      page.fillOval (MID-50, TOP+80, 100, 60);  // lower torso

      page.setColor (Color.black);
      page.fillOval (MID-10, TOP+10, 5, 5);   // left eye
      page.fillOval (MID+5, TOP+10, 5, 5);    // right eye

      page.drawArc (MID-10, TOP+20, 20, 10, 190, 160);   // smile

      page.drawLine (MID-25, TOP+60, MID-50, TOP+40);  // left arm
      page.drawLine (MID+25, TOP+60, MID+55, TOP+60);  // right arm

      page.drawLine (MID-20, TOP+5, MID+20, TOP+5);  // brim of hat
      page.fillRect (MID-15, TOP-20, 30, 25);        // top of hat
   }
}
```