Walkthrough of the Learning Aids

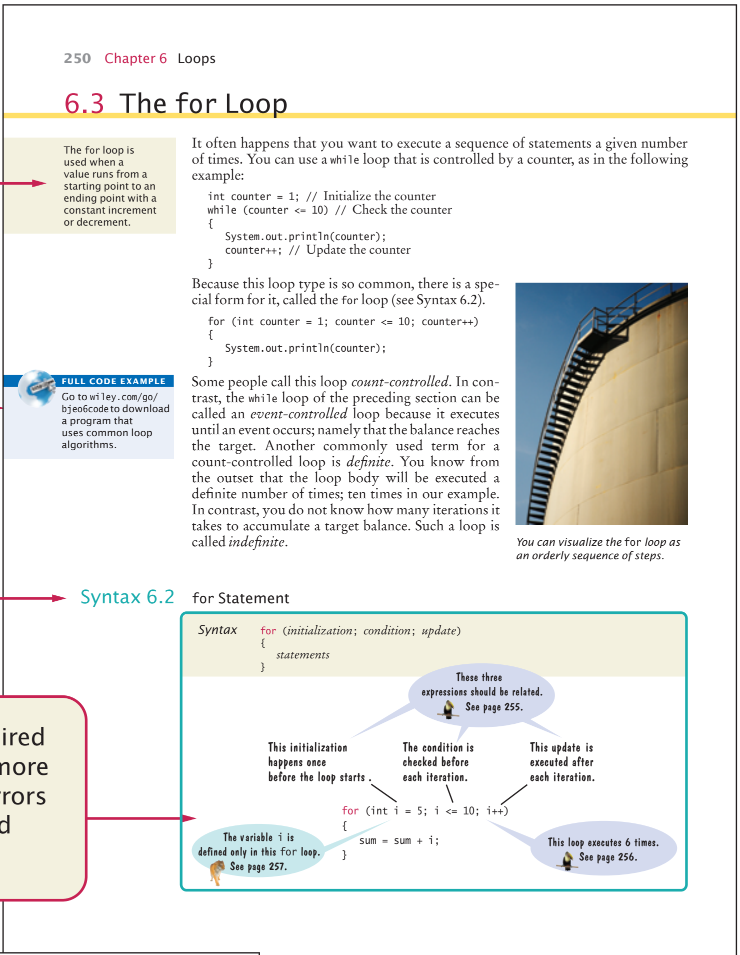
The pedagogical elements in this book work together to focus on and reinforce key concepts and fundamental principles of programming, with additional tips and detail organized to support and deepen these fundamentals. In addition to traditional features, such as chapter objectives and a wealth of exercises, each chapter contains elements geared to today's visual learner.

Throughout each chapter, **margin notes** show where new concepts are introduced and provide an outline of key ideas.

Additional **full code examples** provides complete programs for students to run and modify.

Annotated **syntax boxes** provide a quick, visual overview of new language constructs.

Annotations explain required components and point to more information on common errors or best practices associated with the syntax.



250 Chapter 6 Loops

6.3 The for Loop

The for loop is used when a value runs from a starting point to an ending point with a constant increment or decrement.

It often happens that you want to execute a sequence of statements a given number of times. You can use a while loop that is controlled by a counter, as in the following example:

```
int counter = 1; // Initialize the counter
while (counter <= 10) // Check the counter
{
   System.out.println(counter);
   counter++; // Update the counter
}
```

Because this loop type is so common, there is a special form for it, called the for loop (see Syntax 6.2).

```
for (int counter = 1; counter <= 10; counter++)
{
   System.out.println(counter);
}
```

FULL CODE EXAMPLE
Go to wiley.com/go/bjeo6code to download a program that uses common loop algorithms.

Some people call this loop *count-controlled*. In contrast, the while loop of the preceding section can be called an *event-controlled* loop because it executes until an event occurs; namely that the balance reaches the target. Another commonly used term for a count-controlled loop is *definite*. You know from the outset that the loop body will be executed a definite number of times; ten times in our example. In contrast, you do not know how many iterations it takes to accumulate a target balance. Such a loop is called *indefinite*.

You can visualize the for *loop as an orderly sequence of steps.*

Syntax 6.2 for Statement

```
Syntax    for (initialization; condition; update)
          {
             statements
          }
```

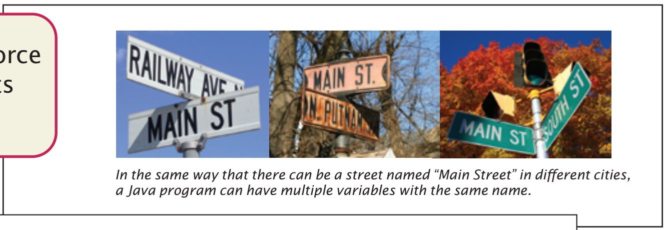
```
for (int i = 5; i <= 10; i++)
{
   sum = sum + i;
}
```

These three expressions should be related. See page 255.
This initialization happens once before the loop starts .
The condition is checked before each iteration.
This update is executed after each iteration.
The variable i is defined only in this for loop. See page 257.
This loop executes 6 times. See page 256.

Like a variable in a computer program, a parking space has an identifier and a contents.

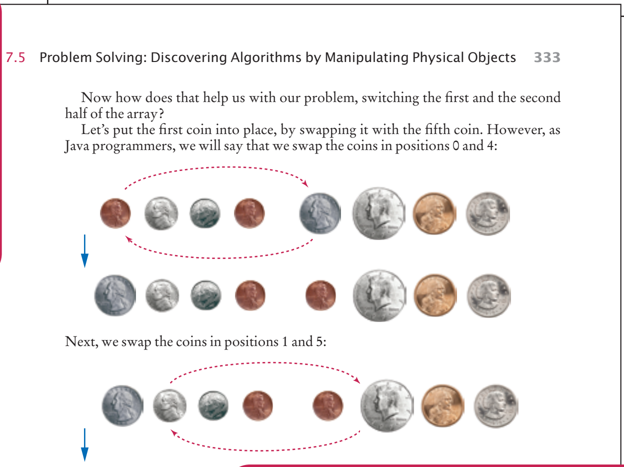
Analogies to everyday objects are used to explain the nature and behavior of concepts such as variables, data types, loops, and more.

Memorable photos reinforce analogies and help students remember the concepts.



In the same way that there can be a street named "Main Street" in different cities, a Java program can have multiple variables with the same name.

Problem Solving sections teach techniques for generating ideas and evaluating proposed solutions, often using pencil and paper or other artifacts. These sections emphasize that most of the planning and problem solving that makes students successful happens away from the computer.

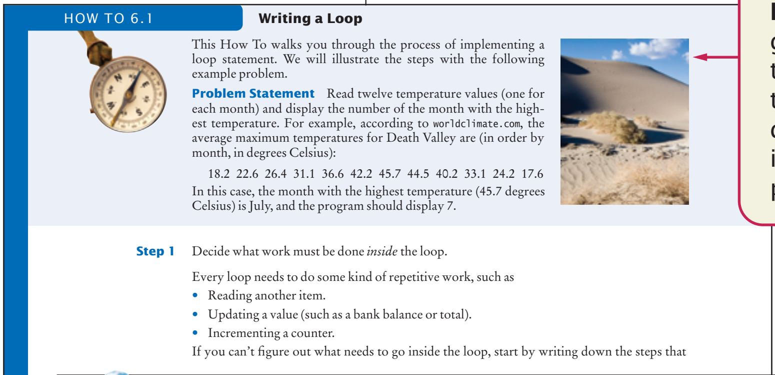


7.5 Problem Solving: Discovering Algorithms by Manipulating Physical Objects 333

Now how does that help us with our problem, switching the first and the second half of the array?

Let's put the first coin into place, by swapping it with the fifth coin. However, as Java programmers, we will say that we swap the coins in positions 0 and 4:

Next, we swap the coins in positions 1 and 5:



HOW TO 6.1 **Writing a Loop**

This How To walks you through the process of implementing a loop statement. We will illustrate the steps with the following example problem.

Problem Statement Read twelve temperature values (one for each month) and display the number of the month with the highest temperature. For example, according to `worldclimate.com`, the average maximum temperatures for Death Valley are (in order by month, in degrees Celsius):

18.2 22.6 26.4 31.1 36.6 42.2 45.7 44.5 40.2 33.1 24.2 17.6

In this case, the month with the highest temperature (45.7 degrees Celsius) is July, and the program should display 7.

Step 1 Decide what work must be done *inside* the loop.

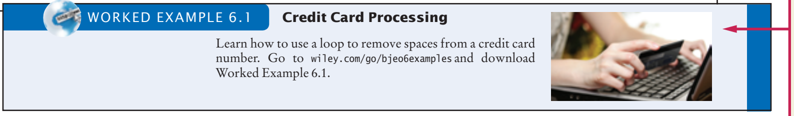
Every loop needs to do some kind of repetitive work, such as

- Reading another item.
- Updating a value (such as a bank balance or total).
- Incrementing a counter.

If you can't figure out what needs to go inside the loop, start by writing down the steps that

How To guides give step-by-step guidance for common programming tasks, emphasizing planning and testing. They answer the beginner's question, "Now what do I do?" and integrate key concepts into a problem-solving sequence.

Worked Examples apply the steps in the How To to a different example, showing how they can be used to plan, implement, and test a solution to another programming problem.



WORKED EXAMPLE 6.1 **Credit Card Processing**

Learn how to use a loop to remove spaces from a credit card number. Go to `wiley.com/go/bjeo6examples` and download Worked Example 6.1.

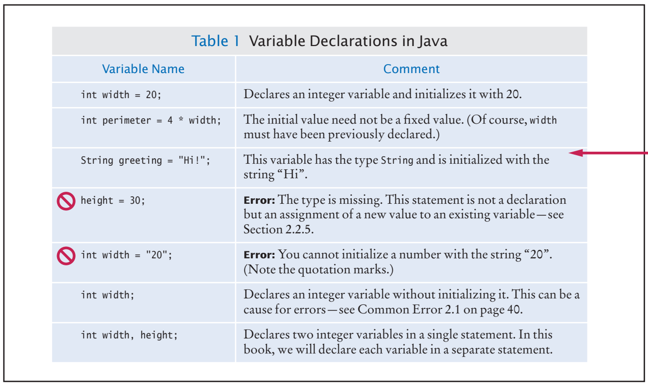
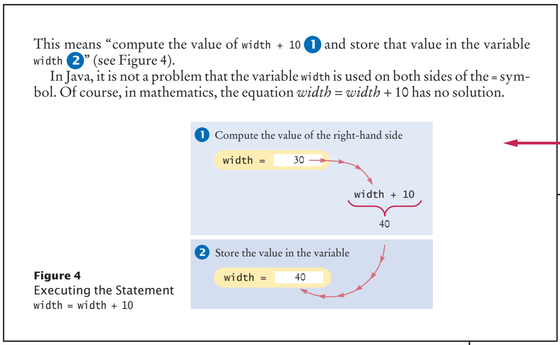


Table 1 Variable Declarations in Java

	Variable Name	Comment
	`int width = 20;`	Declares an integer variable and initializes it with 20.
	`int perimeter = 4 * width;`	The initial value need not be a fixed value. (Of course, `width` must have been previously declared.)
	`String greeting = "Hi!";`	This variable has the type `String` and is initialized with the string "Hi".
⊘	`height = 30;`	**Error:** The type is missing. This statement is not a declaration but an assignment of a new value to an existing variable—see Section 2.2.5.
⊘	`int width = "20";`	**Error:** You cannot initialize a number with the string "20". (Note the quotation marks.)
	`int width;`	Declares an integer variable without initializing it. This can be a cause for errors—see Common Error 2.1 on page 40.
	`int width, height;`	Declares two integer variables in a single statement. In this book, we will declare each variable in a separate statement.

Example tables support beginners with multiple, concrete examples. These tables point out common errors and present another quick reference to the section's topic.



This means “compute the value of width + 10 ❶ and store that value in the variable width ❷” (see Figure 4).

In Java, it is not a problem that the variable width is used on both sides of the = symbol. Of course, in mathematics, the equation *width* = *width* + 10 has no solution.

❶ Compute the value of the right-hand side
width = 30
width + 10
40
❷ Store the value in the variable
width = 40

Figure 4 Executing the Statement width = width + 10

Progressive figures trace code segments to help students visualize the program flow. Color is used consistently to make variables and other elements easily recognizable.

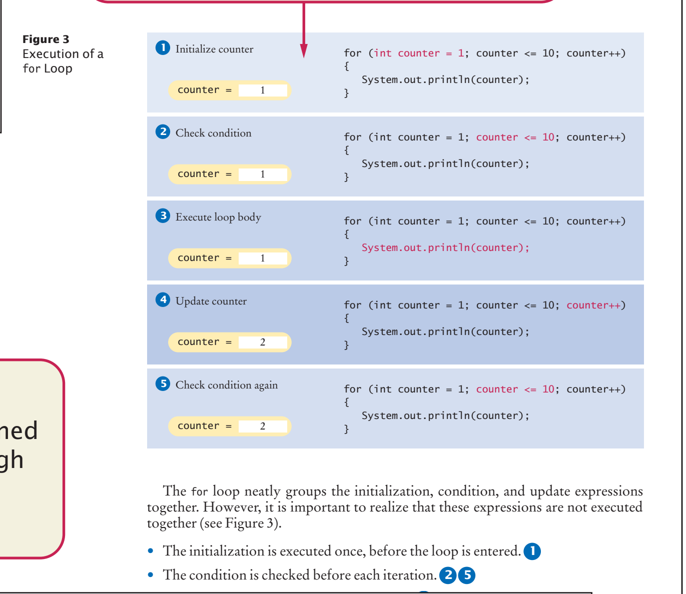


Figure 3 Execution of a for Loop

❶ Initialize counter

counter = 1

```
for (int counter = 1; counter <= 10; counter++)
{
   System.out.println(counter);
}
```

❷ Check condition

counter = 1

```
for (int counter = 1; counter <= 10; counter++)
{
   System.out.println(counter);
}
```

❸ Execute loop body

counter = 1

```
for (int counter = 1; counter <= 10; counter++)
{
   System.out.println(counter);
}
```

❹ Update counter

counter = 2

```
for (int counter = 1; counter <= 10; counter++)
{
   System.out.println(counter);
}
```

❺ Check condition again

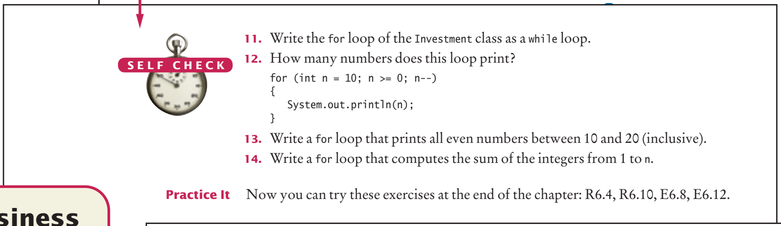
counter = 2

```
for (int counter = 1; counter <= 10; counter++)
{
   System.out.println(counter);
}
```

The for loop neatly groups the initialization, condition, and update expressions together. However, it is important to realize that these expressions are not executed together (see Figure 3).

- The initialization is executed once, before the loop is entered. ❶
- The condition is checked before each iteration. ❷❺

Self-check exercises at the end of each section are designed to make students think through the new material—and can spark discussion in lecture.



SELF CHECK

11. Write the for loop of the Investment class as a while loop.

12. How many numbers does this loop print?

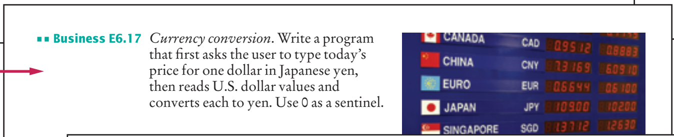
```
for (int n = 10; n >= 0; n--)
{
   System.out.println(n);
}
```

13. Write a for loop that prints all even numbers between 10 and 20 (inclusive).

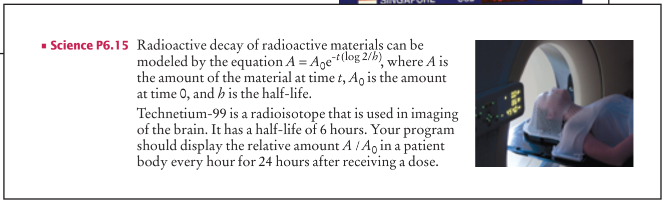
14. Write a for loop that computes the sum of the integers from 1 to n.

Practice It Now you can try these exercises at the end of the chapter: R6.4, R6.10, E6.8, E6.12.

Optional **science and business exercises** engage students with realistic applications of Java.

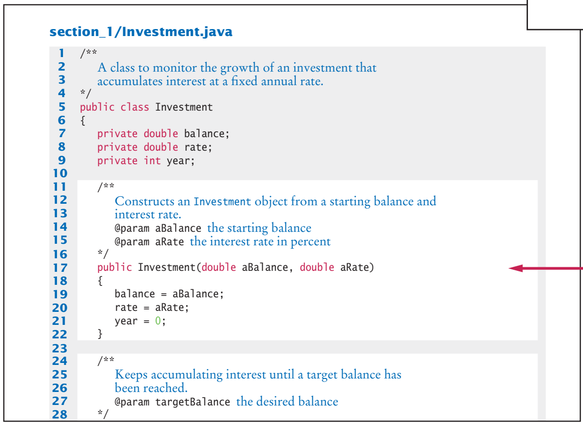


▪▪ **Business E6.17** *Currency conversion.* Write a program that first asks the user to type today’s price for one dollar in Japanese yen, then reads U.S. dollar values and converts each to yen. Use 0 as a sentinel.



▪ **Science P6.15** Radioactive decay of radioactive materials can be modeled by the equation $A = A_0 e^{-t(\log 2/h)}$, where A is the amount of the material at time t, A_0 is the amount at time 0, and h is the half-life.

Technetium-99 is a radioisotope that is used in imaging of the brain. It has a half-life of 6 hours. Your program should display the relative amount A / A_0 in a patient body every hour for 24 hours after receiving a dose.



section_1/Investment.java

```
/**
   A class to monitor the growth of an investment that
   accumulates interest at a fixed annual rate.
*/
public class Investment
{
   private double balance;
   private double rate;
   private int year;

   /**
      Constructs an Investment object from a starting balance and
      interest rate.
      @param aBalance the starting balance
      @param aRate the interest rate in percent
   */
   public Investment(double aBalance, double aRate)
   {
      balance = aBalance;
      rate = aRate;
      year = 0;
   }

   /**
      Keeps accumulating interest until a target balance has
      been reached.
      @param targetBalance the desired balance
   */
```

Program listings are carefully designed for easy reading, going well beyond simple color coding. Methods are set off by a subtle outline.

Common Errors describe the kinds of errors that students often make, with an explanation of why the errors occur, and what to do about them.

Common Error 7.4

Length and Size

Unfortunately, the Java syntax for determining the number of elements in an array, an array list, and a string is not at all consistent. It is a common error to confuse these. You just have to remember the correct syntax for every data type.

Data Type	Number of Elements
Array	`a.length`
Array list	`a.size()`
String	`a.length()`

Programming Tips explain good programming practices, and encourage students to be more productive with tips and techniques such as hand-tracing.

Programming Tip 5.5

Hand-Tracing

A very useful technique for understanding whether a program works correctly is called *hand-tracing*. You simulate the program's activity on a sheet of paper. You can use this method with pseudocode or Java code.

Get an index card, a cocktail napkin, or whatever sheet of paper is within reach. Make a column for each variable. Have the program code ready. Use a marker, such as a paper clip, to mark the current statement. In your mind, execute statements one at a time. Every time the value of a variable changes, cross out the old value and write the new value below the old one.

Hand-tracing helps you understand whether a program works correctly.

For example, let's trace the `getTax` method with the data from the program run above.

When the `TaxReturn` object is constructed, the `income` instance variable is set to 80,000 and `status` is set to `MARRIED`. Then the `getTax` method is called. In lines 31 and 32 of `TaxReturn.java`, `tax1` and `tax2` are initialized to `0`.

```
29 public double getTax()
30 {
31    double tax1 = 0;
32    double tax2 = 0;
33
```

income	status	tax1	tax2
80000	MARRIED	0	0

Because `status` is not `SINGLE`, we move to the `else` branch of the outer if statement (line 46).

```
34    if (status == SINGLE)
35    {
36       if (income <= RATE1_SINGLE_LIMIT)
37       {
38          tax1 = RATE1 * income;
39       }
40       else
41       {
42          tax1 = RATE1 * RATE1_SINGLE_LIMIT;
43          tax2 = RATE2 * (income - RATE1_SINGLE_LIMIT);
```

Special Topics present optional topics and provide additional explanation of others.

Special Topic 11.2

File Dialog Boxes

In a program with a graphical user interface, you will want to use a file dialog box (such as the one shown in the figure below) whenever the users of your program need to pick a file. The `JFileChooser` class implements a file dialog box for the Swing user-interface toolkit.

The `JFileChooser` class has many options to fine-tune the display of the dialog box, but in its most basic form it is quite simple: Construct a file chooser object; then call the `showOpenDialog` or `showSaveDialog` method. Both methods show the same dialog box, but the button for selecting a file is labeled "Open" or "Save", depending on which method you call.

For better placement of the dialog box on the screen, you can specify the user-interface component over which to pop up the dialog box. If you don't care where the dialog box pops up, you can simply pass `null`. The `showOpenDialog` and `showSaveDialog` methods return either `JFileChooser.APPROVE_OPTION`, if the user has chosen a file, or `JFileChooser.CANCEL_OPTION`, if the user canceled the selection. If a file was chosen, then you call the `getSelectedFile` method to obtain a `File` object that describes the file. Here is a complete example:

```
JFileChooser chooser = new JFileChooser();
```

Java 8 Notes provide detail about new features in Java 8.

Java 8 Note 10.4

Lambda Expressions

In the preceding section, you saw how to use interfaces for specifying variations in behavior. The `average` method needs to measure each object, and it does so by calling the `measure` method of the supplied `Measurer` object.

Unfortunately, the caller of the `average` method has to do a fair amount of work; namely, to define a class that implements the `Measurer` interface and to construct an object of that class. Java 8 has a convenient shortcut for these steps, provided that the interface has a *single abstract method*. Such an interface is called a *functional interface* because its purpose is to define a single function. The `Measurer` interface is an example of a functional interface.

To specify that single function, you can use a *lambda expression*, an expression that defines the parameters and return value of a method in a compact notation. Here is an example:

```
(Object obj) -> ((BankAccount) obj).getBalance()
```

This expression defines a function that, given an object, casts it to a `BankAccount` and returns the balance.

Computing & Society presents social and historical topics on computing—for interest and to fulfill the "historical and social context" requirements of the ACM/IEEE curriculum guidelines.

Computing & Society 1.1 Computers Are Everywhere

When computers were first invented in the 1940s, a computer filled an entire room. The photo below shows the ENIAC (*e*lectronic *n*umerical *i*ntegrator *a*nd *c*omputer), completed in 1946 at the University of Pennsylvania. The ENIAC was used by the military to compute the trajectories of projectiles. Nowadays, computing facilities of search engines, Internet shops, and social networks fill huge buildings called data centers. At the other end of the spectrum, computers are all around us. Your cell phone has a computer inside, as do many credit cards and fare cards for public transit. A modern car has several computers—to control the engine, brakes, lights, and the radio.

The advent of ubiquitous computing changed many aspects of our lives. Factories used to employ people to do repetitive assembly tasks that are today carried out by computer-controlled robots, operated by a few people who know how to work with those computers. Books, music, and movies are nowadays often consumed on computers, and computers are almost always involved in their production. The book that you are reading right now could not have been written without computers.

This transit card contains a computer.