Object Oriented Programming

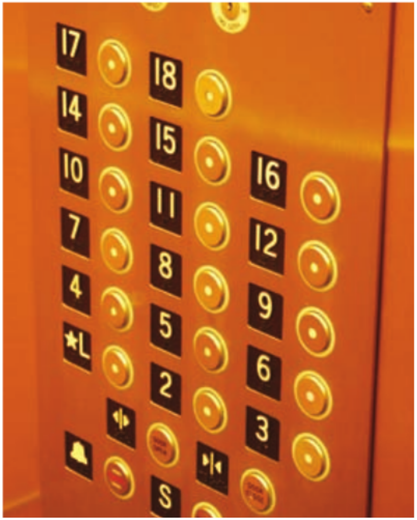
#3 Decision and Loop

Hilmy A. Tawakal & Irfan A.

September 30, 2016

Table of contents

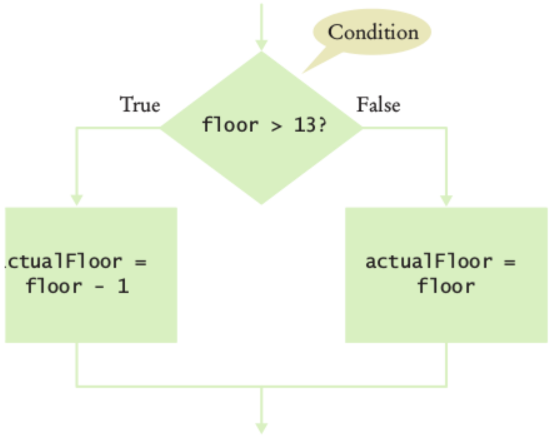
Spooky Number



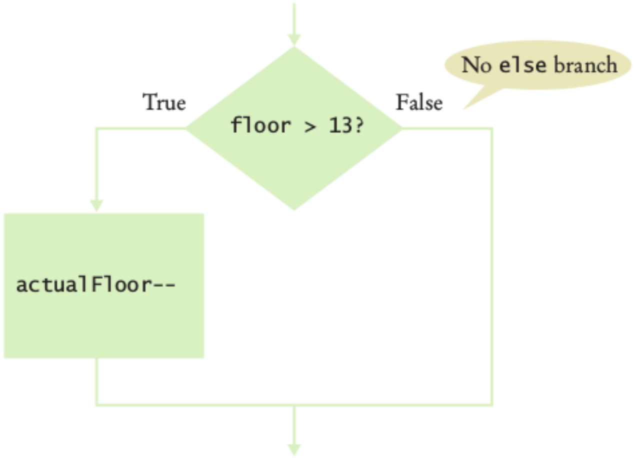
```
int actualFloor;

if (floor > 13)
{
   actualFloor = floor - 1;
}
else
{
   actualFloor = floor;
}
```

Flow Chart


Condition
True
False
floor > 13?
ctualFloor =
floor - 1
actualFloor =
floor

Flow Chart



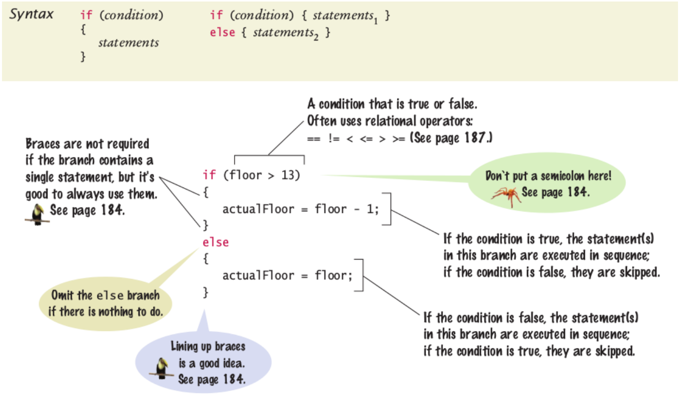
Without Else

```
int actualFloor = floor;

if (floor > 13)
{
   actualFloor--;
} // No else needed
```

Illustration

If Structure



Syntax

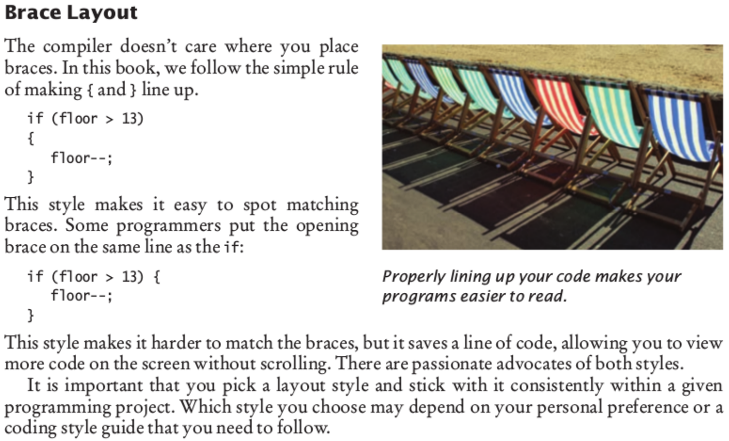
```
if (condition)
{
   statements
}
```

```
if (condition) { statements1 }
else { statements2 }
```

```
if (floor > 13)
{
   actualFloor = floor - 1;
}
else
{
   actualFloor = floor;
}
```

A condition that is true or false. Often uses relational operators: == != < <= > >= (See page 187.)
Braces are not required if the branch contains a single statement, but it's good to always use them. See page 184.
Don't put a semicolon here! See page 184.
If the condition is true, the statement(s) in this branch are executed in sequence; if the condition is false, they are skipped.
Omit the else branch if there is nothing to do.
If the condition is false, the statement(s) in this branch are executed in sequence; if the condition is true, they are skipped.
Lining up braces is a good idea. See page 184.

Tips 1: Using Braces



Brace Layout

The compiler doesn't care where you place braces. In this book, we follow the simple rule of making { and } line up.

```
if (floor > 13)
{
   floor--;
}
```

This style makes it easy to spot matching braces. Some programmers put the opening brace on the same line as the if:

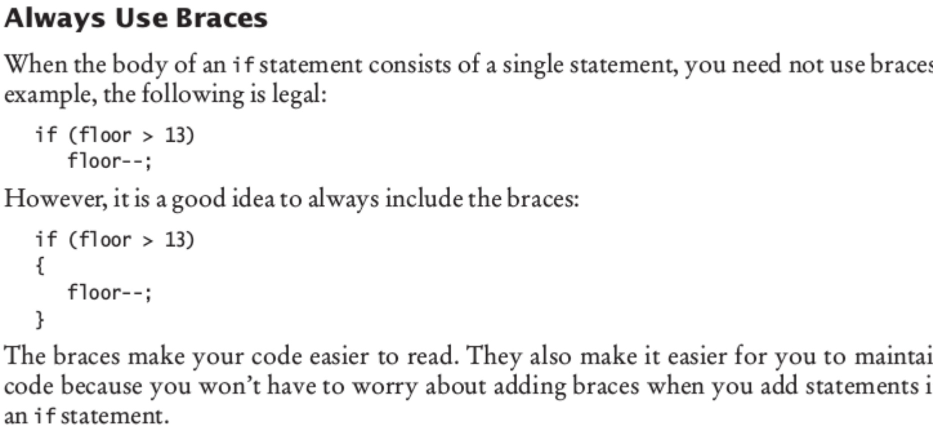
```
if (floor > 13) {
   floor--;
}
```

Properly lining up your code makes your programs easier to read.

This style makes it harder to match the braces, but it saves a line of code, allowing you to view more code on the screen without scrolling. There are passionate advocates of both styles.

It is important that you pick a layout style and stick with it consistently within a given programming project. Which style you choose may depend on your personal preference or a coding style guide that you need to follow.

Tips 1: Using Braces



Always Use Braces

When the body of an `if` statement consists of a single statement, you need not use braces
example, the following is legal:

```
if (floor > 13)
   floor--;
```

However, it is a good idea to always include the braces:

```
if (floor > 13)
{
   floor--;
}
```

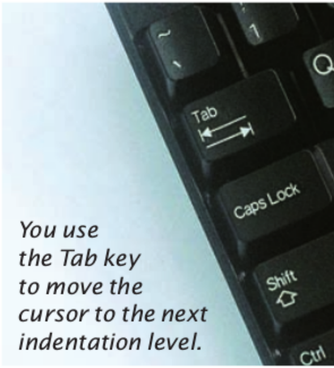
The braces make your code easier to read. They also make it easier for you to maintai
code because you won't have to worry about adding braces when you add statements i
an `if` statement.

Tips 2: Semicolon error

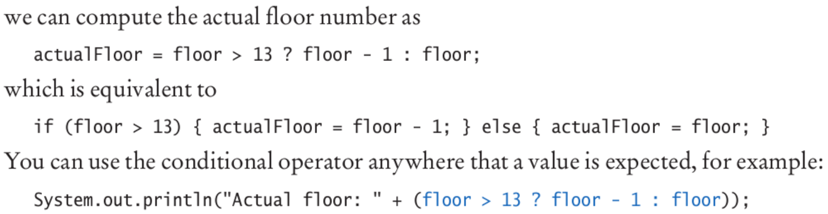
```
if (floor > 13) ; // ERROR
{
   floor--;
}
```

Tips 3: Indentation

```
public class ElevatorSimulation
{
|   public static void main(String[] args)
|   {
|   |   int floor;
|   |   . . .
|   |   if (floor > 13)
|   |   {
|   |   |   floor--;
|   |   }
|   |   . . .
|   }
|   |   |   |
0   1   2   3    Indentation level
```


You use the Tab key to move the cursor to the next indentation level.

Tips 4: Ternary Operator



we can compute the actual floor number as

```
actualFloor = floor > 13 ? floor - 1 : floor;
```

which is equivalent to

```
if (floor > 13) { actualFloor = floor - 1; } else { actualFloor = floor; }
```

You can use the conditional operator anywhere that a value is expected, for example:

```
System.out.println("Actual floor: " + (floor > 13 ? floor - 1 : floor));
```

Tips 5: Avoid repetition

```
if (floor > 13)
{
   actualFloor = floor - 1;
   System.out.println("Actual floor: " + actualFloor);
}
else
{
   actualFloor = floor;
   System.out.println("Actual floor: " + actualFloor);
}
```

Tips 5: Avoid repetition

```
if (floor > 13)
{
   actualFloor = floor - 1;
}
else
{
   actualFloor = floor;
}
System.out.println("Actual floor: " + actualFloor);
```

Relational Operator

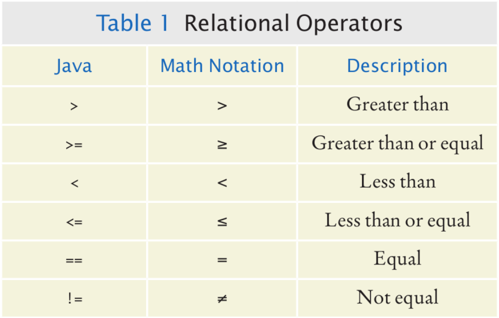
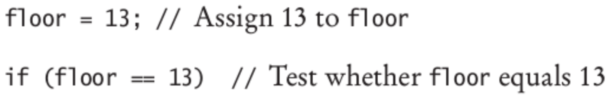


Table 1 Relational Operators

Java	Math Notation	Description
>	>	Greater than
>=	≥	Greater than or equal
<	<	Less than
<=	≤	Less than or equal
==	=	Equal
!=	≠	Not equal

Relational Operator



```
floor = 13; // Assign 13 to floor
if (floor == 13)   // Test whether floor equals 13
```

Operator Order

```
floor - 1 < 13
```

Short-Circuit

```
quantity > 0 && price / quantity < 10
```

Comparing String

```
if (string1.equals(string2)) . . .
```

Comparing String

```
if (string1 == string2) // Not useful
```

Comparing String

```
string1.compareTo(string2) < 0
string1.compareTo(string2) > 0
string1.compareTo(string2) == 0
```

- All uppercase letters come before the lowercase letters. For example, "Z" comes before "a".
- The space character comes before all printable characters.
- Numbers come before letters.

Denver Airport

IF implementation

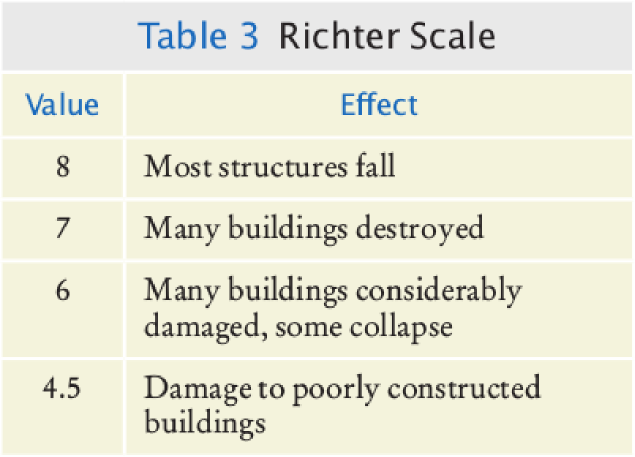
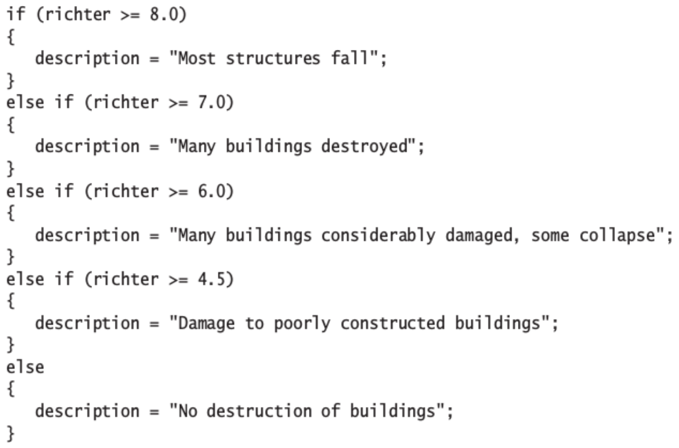


Table 3 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings

Sollution

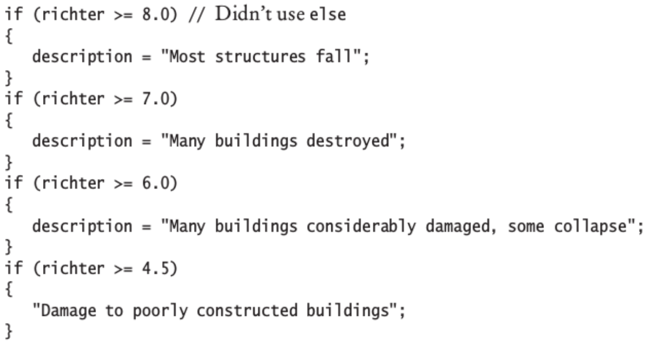


```
if (richter >= 8.0)
{
   description = "Most structures fall";
}
else if (richter >= 7.0)
{
   description = "Many buildings destroyed";
}
else if (richter >= 6.0)
{
   description = "Many buildings considerably damaged, some collapse";
}
else if (richter >= 4.5)
{
   description = "Damage to poorly constructed buildings";
}
else
{
   description = "No destruction of buildings";
}
```

Reverse Order Problem?

```
if (richter >= 4.5) // Tests in wrong order
{
   description = "Damage to poorly constructed buildings";
}
else if (richter >= 6.0)
{
   description = "Many buildings considerably damaged, some collapse";
}
else if (richter >= 7.0)
{
   description = "Many buildings destroyed";
```

Multiple IF Problem?



```
if (richter >= 8.0) // Didn't use else
{
   description = "Most structures fall";
}
if (richter >= 7.0)
{
   description = "Many buildings destroyed";
}
if (richter >= 6.0)
{
   description = "Many buildings considerably damaged, some collapse";
}
if (richter >= 4.5)
{
   "Damage to poorly constructed buildings";
}
```

Switch Case

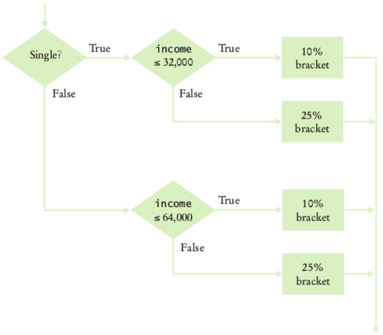
```
int digit = . . .;
switch (digit)
{
   case 1: digitName = "one"; break;
   case 2: digitName = "two"; break;
   case 3: digitName = "three"; break;
   case 4: digitName = "four"; break;
   case 5: digitName = "five"; break;
   case 6: digitName = "six"; break;
   case 7: digitName = "seven"; break;
   case 8: digitName = "eight"; break;
   case 9: digitName = "nine"; break;
   default: digitName = ""; break;
}
```

```
int digit = . . .;
if (digit == 1) { digitName = "one"; }
else if (digit == 2) { digitName = "two"; }
else if (digit == 3) { digitName = "three"; }
else if (digit == 4) { digitName = "four"; }
else if (digit == 5) { digitName = "five"; }
else if (digit == 6) { digitName = "six"; }
else if (digit == 7) { digitName = "seven"; }
else if (digit == 8) { digitName = "eight"; }
else if (digit == 9) { digitName = "nine"; }
else { digitName = ""; }
```

Switch Case

The switch *statement lets you choose from a fixed set of alternatives.*

Nested If


Single?
True
False
income ≤ 32,000
True
False
10% bracket
25% bracket
income ≤ 64,000
True
False
10% bracket
25% bracket

The Dangling else Problem

```
double shippingCharge = 5.00; // $5 inside continental U.S.
if (country.equals("USA"))
   if (state.equals("HI"))
      shippingCharge = 10.00; // Hawaii is more expensive
else // Pitfall!
   shippingCharge = 20.00; // As are foreign shipments
```

```
double shippingCharge = 5.00; // $5 inside continental U.S.
if (country.equals("USA"))
   if (state.equals("HI"))
      shippingCharge = 10.00; // Hawaii is more expensive
   else // Pitfall!
      shippingCharge = 20.00; // As are foreign shipments
```

```
double shippingCharge = 5.00; // $5 inside continental U.S.
if (country.equals("USA"))
{
   if (state.equals("HI"))
   {
      shippingCharge = 10.00; // Hawaii is more expensive
   }
}
else
{
   shippingCharge = 20.00; // As are foreign shipments
}
```

Variable Scope

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.

```
if (status == TAXABLE)
{
    double tax = price * TAX_RATE;
    price = price + tax;
}
```

Variable Scope

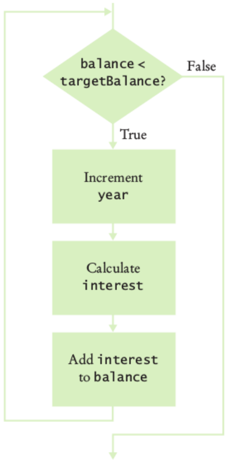
```
double tax = 0;
if (status == TAXABLE)
{
   tax = price * TAX_RATE;
}
price = price + tax;
```

```
double tax = 0;
if (status == TAXABLE)
{
   double tax = price * TAX_RATE;
   // Error: Cannot declare another variable with the same name
   price = price + tax;
}
```

Variable Scope

```
if (Math.random() > 0.5)
{
   Rectangle r = new Rectangle(5, 10, 20, 30);
   . . .
} // Scope of r ends here
else
{
   int r = 5;
   // OK—it is legal to declare another r here
   . . .
}
```

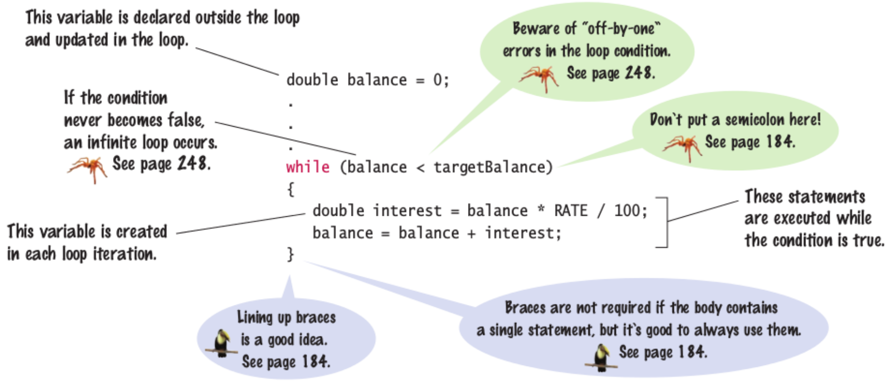
While Loop



```
while (balance < targetBalance)
{
   year++;
   double interest = balance * RATE / 100;
   balance = balance + interest;
}
```

While Loop

Syntax
```
while (condition)
{
   statements
}
```



```
double balance = 0;
.
.
.
while (balance < targetBalance)
{
   double interest = balance * RATE / 100;
   balance = balance + interest;
}
```

This variable is declared outside the loop and updated in the loop.
If the condition never becomes false, an infinite loop occurs. See page 248.
This variable is created in each loop iteration.
Beware of "off-by-one" errors in the loop condition. See page 248.
Don't put a semicolon here! See page 184.
These statements are executed while the condition is true.
Lining up braces is a good idea. See page 184.
Braces are not required if the body contains a single statement, but it's good to always use them. See page 184.

Problem?

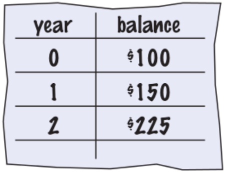
```
int year = 1;
while (year <= 20)
{
   double interest = balance * RATE / 100;
   balance = balance + interest;
}
```

Problem?

```
int year = 20;
while (year > 0)
{
   double interest = balance * RATE / 100;
   balance = balance + interest;
   year++;
}
```

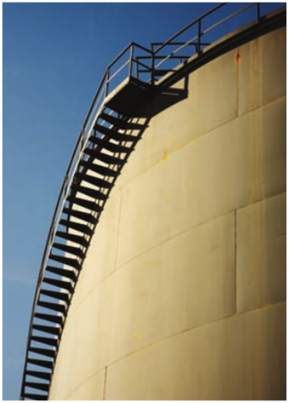
Off by one error

```
int year = 0;
while (balance < targetBalance)
{
   year++;
   balance = balance * (1 + RATE / 100);
}
System.out.println("The investment doubled after "
   + year + " years.");
```



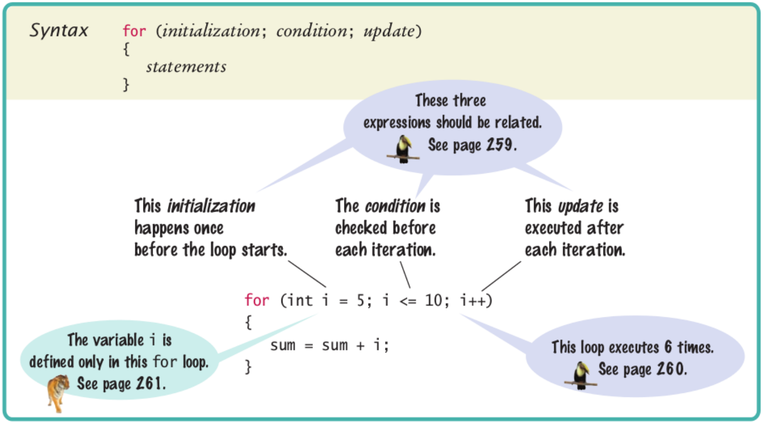
year	balance
0	$100
1	$150
2	$225

For Loop



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

For Loop



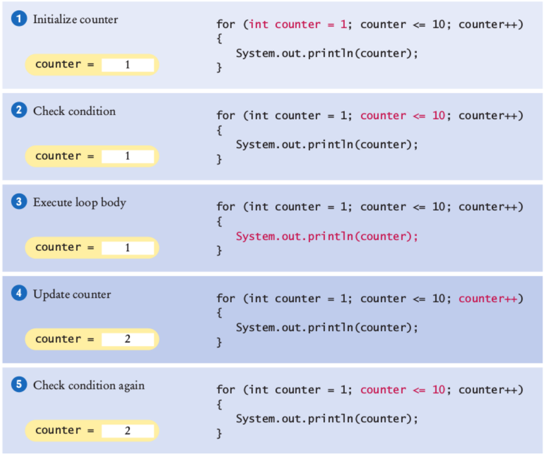
Syntax

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

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

These three expressions should be related. See page 259.
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 261.
This loop executes 6 times. See page 260.

For Illustration



1 Initialize counter

counter = 1

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

2 Check condition

counter = 1

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

3 Execute loop body

counter = 1

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

4 Update counter

counter = 2

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

5 Check condition again

counter = 2

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

Loop Example

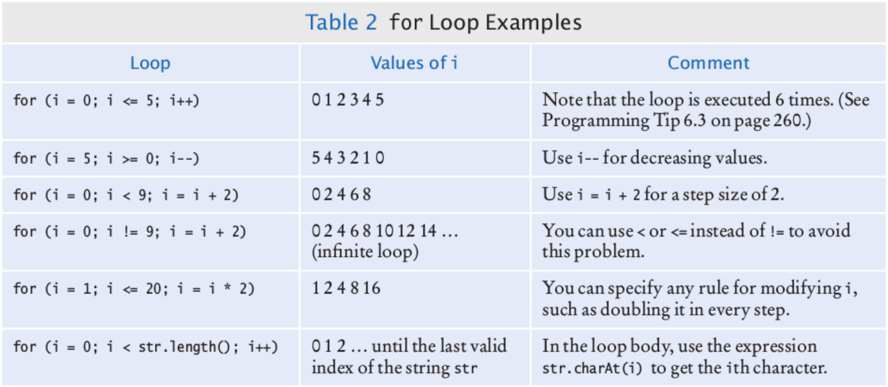
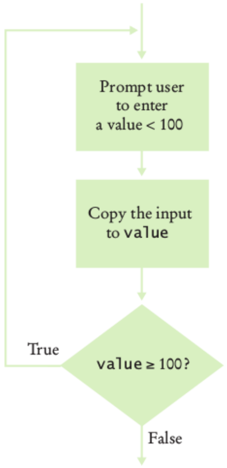


Table 2 for Loop Examples

Loop	Values of i	Comment
`for (i = 0; i <= 5; i++)`	0 1 2 3 4 5	Note that the loop is executed 6 times. (See Programming Tip 6.3 on page 260.)
`for (i = 5; i >= 0; i--)`	5 4 3 2 1 0	Use `i--` for decreasing values.
`for (i = 0; i < 9; i = i + 2)`	0 2 4 6 8	Use `i = i + 2` for a step size of 2.
`for (i = 0; i != 9; i = i + 2)`	0 2 4 6 8 10 12 14 … (infinite loop)	You can use `<` or `<=` instead of `!=` to avoid this problem.
`for (i = 1; i <= 20; i = i * 2)`	1 2 4 8 16	You can specify any rule for modifying `i`, such as doubling it in every step.
`for (i = 0; i < str.length(); i++)`	0 1 2 … until the last valid index of the string `str`	In the loop body, use the expression `str.charAt(i)` to get the ith character.

Do - While


Prompt user to enter a value < 100
Copy the input to value
value ≥ 100?
True
False