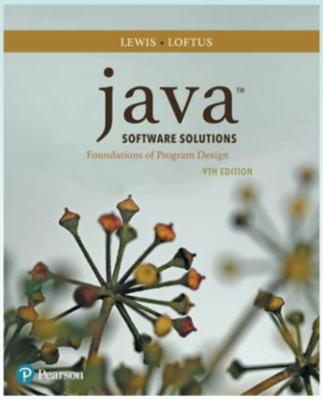
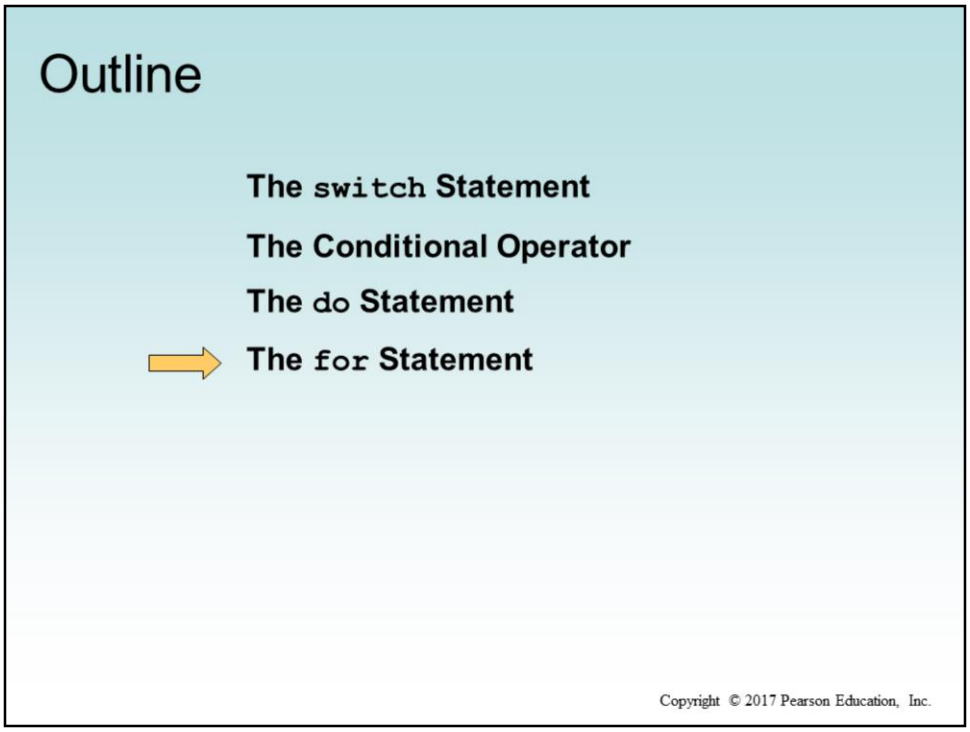
Chapter 6
More Conditionals and Loops



Java Software Solutions
Foundations of Program Design
9th Edition

John Lewis
William Loftus



-while/do loops are typically used when you **don't** initially know how many times to execute the loop

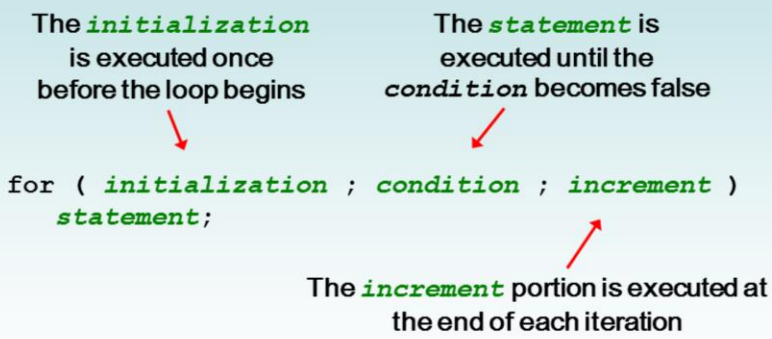
-Think about our labs where we use a while/do loop that take user input until they are done

-We don't know when we write the while loop how many times the user will enter input

-for loops are to be used when we **do** know (when we write the code) how many times to execute it

The for Statement

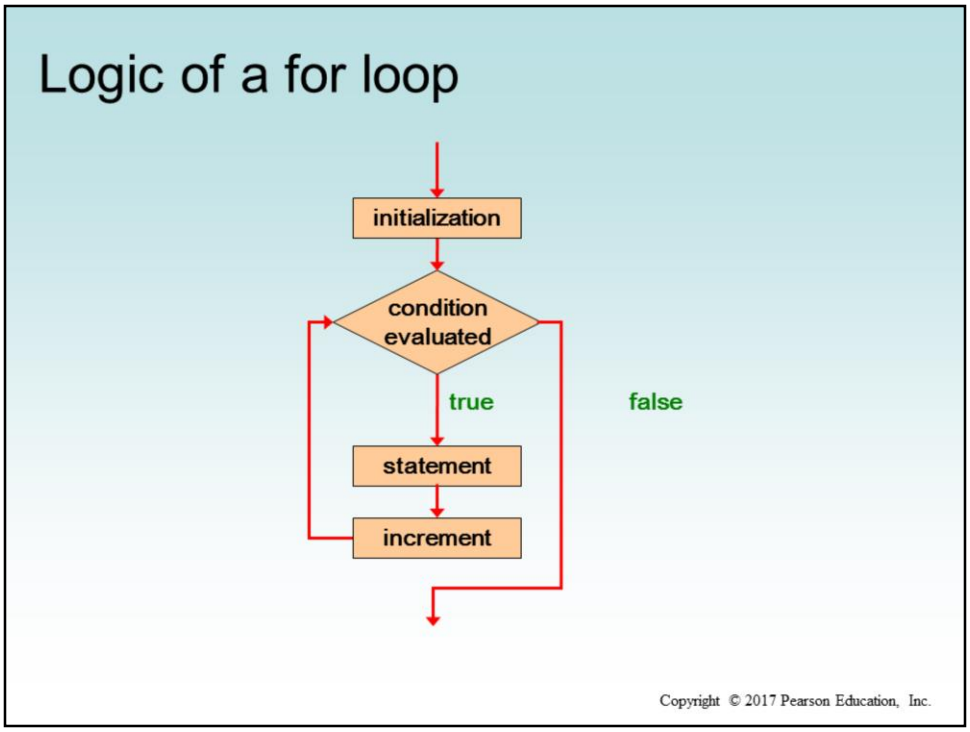
- A *for statement* has the following syntax:



-Note the three parts of the for loop (initialization, condition, increment)

-Note the condition result is a binary result (true or false), similar to if and while/do loops

-Note also that although named the “increment” portion, you can also decrement



-The order of a for loop is important to understand:

1) the initialization statement is executed

2) the condition is tested

3) if true, statements are executed, if false, loop exits

4) the increment statement is executed

continue steps 2,3,4 until the condition is false and thus the loop exits

The for Statement

- A `for` loop is functionally equivalent to the following `while` loop structure:

```
initialization;
while ( condition )
{
   statement;
   increment;
}
```

The for Statement

- An example of a `for` loop:

```
for (int count=1; count <= 5; count++)
   System.out.println (count);
```

- The initialization section can be used to declare a variable
- Like a `while` loop, the condition of a `for` loop is tested prior to executing the loop body
- Therefore, the body of a `for` loop will execute zero or more times

-Note how the variable **count** is used as the counter that determines how many times we execute the loop

-We often refer to this variable as the **loop counter** since it counts how many times we execute the loop

-Note how since we are using this variable **for** the for loop, we can declare it **in** the for loop

-Since this is declared in the loop, it's scope is only in the loop

-It cannot be used outside the loop

The for Statement

- The increment section can perform any calculation:

```
for (int num=100; num > 0; num -= 5)
   System.out.println (num);
```

- A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance
- See `Multiples.java`
- See `Stars.java`

```
//********************************************************************
//  Multiples.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a for loop.
//********************************************************************

import java.util.Scanner;

public class Multiples
{
   //-----------------------------------------------------------------
   //  Prints multiples of a user-specified number up to a user-
   //  specified limit.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      final int PER_LINE = 5;
      int value, limit, mult, count = 0;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter a positive value: ");
      value = scan.nextInt();
```

continue

continue

```
      System.out.print ("Enter an upper limit: ");
      limit = scan.nextInt();

      System.out.println ();
      System.out.println ("The multiples of " + value + " between " +
                          value + " and " + limit + " (inclusive) are:");

      for (mult = value; mult <= limit; mult += value)
      {
         System.out.print (mult + "\t");

         // Print a specific number of values per line of output
         count++;
         if (count % PER_LINE == 0)
            System.out.println();
      }
   }
}
```

Sample Run

```
Enter a positive value: 7
Enter an upper limit: 400

The multiples of 7 between 7 and 400 (inclusive) are:
7       14      21      28      35
42      49      56      63      70
77      84      91      98      105
112     119     126     133     140
147     154     161     168     175
182     189     196     203     210
217     224     231     238     245
252     259     266     273     280
287     294     301     308     315
322     329     336     343     350
357     364     371     378     385
392     399
```

```
//********************************************************************
//  Stars.java       Author: Lewis/Loftus
//
//  Demonstrates the use of nested for loops.
//********************************************************************

public class Stars
{
   //-----------------------------------------------------------------
   //  Prints a triangle shape using asterisk (star) characters.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      final int MAX_ROWS = 10;

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print ("*");

         System.out.println();
      }
   }
}
```


-Note how as with if and while statements, we can nest for loops

-One for loop can contain another for loop

```
//*********************
//  Stars.java       Auth
//
//  Demonstrates the use
//*********************

public class Stars
{
   //--------------------
   //  Prints a triangle
   //--------------------
   public static void mai
   {
      final int MAX_ROWS

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print ("*");

         System.out.println();
      }
   }
}
```

Output

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Quick Check

Write a code fragment that rolls a die 100 times and counts the number of times a 3 comes up.

Quick Check

Write a code fragment that rolls a die 100 times and counts the number of times a 3 comes up.

```
Die die = new Die();
int count = 0;
for (int num=1; num <= 100; num++)
   if (die.roll() == 3)
      count++;
Sytem.out.println (count);
```

The for Statement

- Each expression in the header of a `for` loop is optional
- If the initialization is left out, no initialization is performed
- If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
- If the increment is left out, no increment operation is performed

-Below is an example that leaves out the **initialization** portion of the for loop

```
int count = 0;
for(; count < 5; ++count)
{
   System.out.println("count: " + count);
}
```

-Below is an example that leaves out both the **initialization** and **increment** portions

```
int count = 0;
for(; count < 5;)
{
   System.out.println("count: " + count);
   ++count;
}
```

For-each Loops

- A variant of the `for` loop simplifies the repetitive processing of items in an iterator
- For example, suppose `bookList` is an `ArrayList<Book>` object
- The following loop will print each book:

```
for (Book myBook : bookList)
   System.out.println (myBook);
```

- This version of a for loop is often called a *for-each loop*

For-each Loops

- A for-each loop can be used on any object that implements the `Iterable` interface
- It eliminates the need to retrieve an iterator and call the `hasNext` and `next` methods explicitly
- It also will be helpful when processing arrays, which are discussed in Chapter 8

Quick Check

Write a for-each loop that prints all of the `Student` objects in an `ArrayList<Student>` object called `roster`.

Quick Check

Write a for-each loop that prints all of the `Student` objects in an `ArrayList<Student>` object called `roster`.

```
for (Student student : roster)
   System.out.println (student);
```