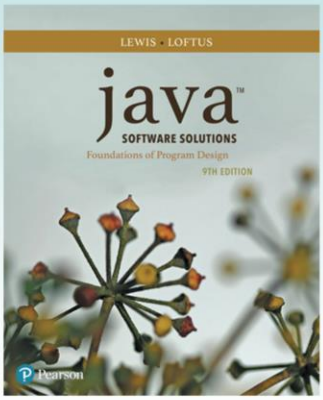
Chapter 6
More Conditionals and Loops



Java Software Solutions

Foundations of Program Design

9th Edition

John Lewis
William Loftus

More Conditionals and Loops

- Now we can fill in some additional details regarding Java conditional and repetition statements
- Chapter 6 focuses on:
  - the `switch` statement
  - the conditional operator
  - the `do` loop
  - the `for` loop

Outline

- **The `switch` Statement**
- **The Conditional Operator**
- **The `do` Statement**
- **The `for` Statement**

The switch Statement

- The *switch statement* provides another way to decide which statement to execute next
- The `switch` statement evaluates an expression, then attempts to match the result to one of several possible *cases*
- Each case contains a value and a list of statements
- The flow of control transfers to statement associated with the first case value that matches

-Note that the conditional used in switch statements can result in one of many values

-Compare this with conditionals used in if statements where the result is binary (true or false)

The switch Statement

- The general syntax of a `switch` statement is:

```
switch ( expression )
{
   case value1 :
      statement-list1
   case value2 :
      statement-list2
   case value3 :
      statement-list3
   case  ...

}
```

`switch` and `case` are reserved words

If *`expression`* matches *`value2`*, control jumps to here

The switch Statement

- Often a *break statement* is used as the last statement in each case's statement list
- A `break` statement causes control to transfer to the end of the `switch` statement
- If a `break` statement is not used, the flow of control will continue into the next case
- Sometimes this may be appropriate, but often we want to execute only the statements associated with one case

-Note that if a break statement is **not** included, flow of control goes to the next case!

-If you do not want multiple cases executed, make sure to include a break statement

The switch Statement

- An example of a switch statement:

```
switch (option)
{
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
}
```

The switch Statement

- A `switch` statement can have an optional *default case*
- The default case has no associated value and simply uses the reserved word `default`
- If the default case is present, control will transfer to it if no other case value matches
- If there is no default case, and no other value matches, control falls through to the statement after the switch

The switch Statement

- The type of a switch expression must be integers, characters, or enumerated types
- As of Java 7, a switch can also be used with strings
- You cannot use a switch with floating point values
- The implicit boolean condition in a `switch` statement is equality
- You cannot perform relational checks with a `switch` statement
- See `GradeReport.java`

-The “type of a switch expression” refers to the result of the conditional expression

-It must result in a char, byte, short, int, or enum, but NOT floating or another expression

-In other words, the condition result is compared (equality) to each case value

-It enters the first case that equals the condition result

```java
//*******************************************************************
//  GradeReport.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a switch statement.
//*******************************************************************

import java.util.Scanner;

public class GradeReport
{
   //-----------------------------------------------------------------
   //  Reads a grade from the user and prints comments accordingly.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int grade, category;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter a numeric grade (0 to 100): ");
      grade = scan.nextInt();

      category = grade / 10;

      System.out.print ("That grade is ");
```

continue

continue

```
      switch (category)
      {
         case 10:
            System.out.println ("a perfect score. Well done.");
            break;
         case 9:
            System.out.println ("well above average. Excellent.");
            break;
         case 8:
            System.out.println ("above average. Nice job.");
            break;
         case 7:
            System.out.println ("average.");
            break;
         case 6:
            System.out.println ("below average. You should see the");
            System.out.println ("instructor to clarify the material "
                                + "presented in class.");
            break;
         default:
            System.out.println ("not passing.");
      }
   }
}
```

continue

Sample Run

```
Enter a numeric grade (0 to 100): 91
That grade is well above average. Excellent.
```

```
swi
{
            System.out.println ("a perfect score. Well done.");
            break;
         case 9:
            System.out.println ("well above average. Excellent.");
            break;
         case 8:
            System.out.println ("above average. Nice job.");
            break;
         case 7:
            System.out.println ("average.");
            break;
         case 6:
            System.out.println ("below average. You should see the");
            System.out.println ("instructor to clarify the material "
                                + "presented in class.");
            break;
         default:
            System.out.println ("not passing.");
      }
   }
}
```

Outline

- The `switch` Statement
- The Conditional Operator
- The `do` Statement
- The `for` Statement

The Conditional Operator

- The *conditional operator* evaluates to one of two expressions based on a boolean condition
- Its syntax is:

```
condition ? expression1 : expression2
```

- If the `condition` is true, `expression1` is evaluated; if it is false, `expression2` is evaluated
- The value of the entire conditional operator is the value of the selected expression

-The conditional operator can be used as a substitute for some if-else statements

-Particularly when if-else statements are simple expressions that return values

-The readability of such an operator, however, can sometimes cause confusion

The Conditional Operator

- The conditional operator is similar to an `if-else` statement, except that it is an expression that returns a value
- For example:

```
larger = ((num1 > num2) ? num1 : num2);
```

- If `num1` is greater than `num2`, then `num1` is assigned to `larger`; otherwise, `num2` is assigned to `larger`
- The conditional operator is *ternary* because it requires three operands

The Conditional Operator

- Another example:

```
System.out.println ("Your change is " + count +
   ((count == 1) ? "Dime" : "Dimes"));
```

- If `count` equals 1, the "Dime" is printed
- If `count` is anything other than 1, then "Dimes" is printed

Quick Check

Express the following logic in a succinct manner using the conditional operator.

```
if (val <= 10)
   System.out.println("It is not greater than 10.");
else
   System.out.println("It is greater than 10.");
```

Quick Check

Express the following logic in a succinct manner using the conditional operator.

```
if (val <= 10)
   System.out.println("It is not greater than 10.");
else
   System.out.println("It is greater than 10.");
```

```
System.out.println("It is" +
   ((val <= 10) ? " not" : "") +
   " greater than 10.");
```

Outline

- The `switch` Statement
- The Conditional Operator
- ➡ The `do` Statement
- The `for` Statement

The do Statement

- A *do statement* has the following syntax:

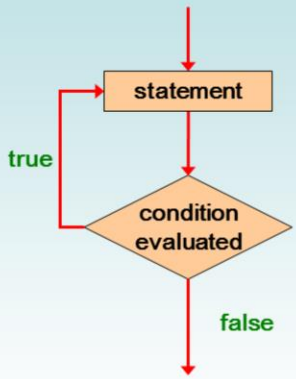
```
do
{
   statement-list;
}
while (condition);
```

- The **statement-list** is executed once initially, and then the **condition** is evaluated
- The statement is executed repeatedly until the condition becomes false

-Note the statements are above the while testing condition

-Notice also the ending semi-colon on the while statement

Logic of a do Loop



The do Statement

- An example of a `do` loop:

```
int count = 0;
do
{
   count++;
   System.out.println (count);
} while (count < 5);
```

- The body of a `do` loop executes at least once
- See `ReverseNumber.java`

```java
//*******************************************************************
//  ReverseNumber.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a do loop.
//*******************************************************************

import java.util.Scanner;

public class ReverseNumber
{
   //-----------------------------------------------------------------
   //  Reverses the digits of an integer mathematically.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int number, lastDigit, reverse = 0;

      Scanner scan = new Scanner (System.in);
```

continue

continue

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

      do
      {
         lastDigit = number % 10;
         reverse = (reverse * 10) + lastDigit;
         number = number / 10;
      }
      while (number > 0);

      System.out.println ("That number reversed is " + reverse);
   }
}
```

continue

```
      System.out.
      number = sc

      do
      {
         lastDigit = number % 10;
         reverse = (reverse * 10) + lastDigit;
         number = number / 10;
      }
      while (number > 0);

      System.out.println ("That number reversed is " + reverse);
   }
}
```

Sample Run

```
Enter a positive integer: 2896
That number reversed is 6982
```

Comparing while and do