Chapter 6
More Conditionals and Loops

Java Software Solutions
Foundations of Program Design
Seventh Edition

John Lewis
William Loftus

Copyright © 2012 Pearson Education, Inc.
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
  – drawing with the aid of conditionals and loops
  – dialog boxes
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
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**
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

• An example of a switch statement:

```java
switch (option)
{
    case 'A':
        aCount++;
        break;
    case 'B':
        bCount++;
        break;
    case 'C':
        cCount++;
        break;
}
```
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 (more) often we want to execute only the statements associated with one case
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**
import java.util.Scanner;

public class GradeReport {
    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 ");
    }
}
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.");
}
}
Sample Run

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

```
    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.");
    }
}
Quiz

You might choose to use a switch statement instead of nested if-else statements if

A) the variable being tested might equal one of several hundred int values
B) the variable being tested might equal one of only a few int values
C) there are two or more int variables being tested, each of which could be one of several hundred values
D) there are two or more int variables being tested, each of which could be one of only a few values
E) none of the above, you would never choose to use a switch statement in place of nested if-else statements under any circumstance
Quiz

You might choose to use a switch statement instead of nested if-else statements if

A) the variable being tested might equal one of several hundred int values

B) the variable being tested might equal one of only a few int values

C) there are two or more int variables being tested, each of which could be one of several hundred values

D) there are two or more int variables being tested, each of which could be one of only a few values

E) none of the above, you would never choose to use a switch statement in place of nested if-else statements under any circumstance

The switch statement can only be used if there is a single variable being tested and it is an integral type (an int or a char in Java). Further, because you have to enumerate each possible value being tested, the switch statement only makes sense if the number of values being tested is a small number.
If \( x \) is currently equal to 5, what will the value of \( x \) be after the switch statement executes?

```java
switch (x) {
    case 3 : x += 1;
    case 4 : x += 2;
    case 5 : x += 3;
    case 6 : x++;
    case 7 : x += 2;
    case 8 : x--;
    case 9 : x++;
}
```

A) 8   B) 6   C) 11   D) 5   E) 10
Quiz

If x is currently equal to 5, what will the value of x be after the switch statement executes?

```
switch (x) {
    case 3 : x += 1;
    case 4 : x += 2;
    case 5 : x += 3;
    case 6 : x++;
    case 7 : x += 2;
    case 8 : x--;    
    case 9 : x++;
}
```

A) 8    B) 6    C) 11    D) 5    E) 10
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
The Conditional Operator

- The *conditional operator* **evaluates** to one of two expressions based on a boolean condition.

- Its syntax is:

  \[
  \text{condition} \ ? \ \text{expression1} \ : \ \text{expression2}
  \]

- 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.

- **It is an operator not a statement.**
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, then "Dime" is printed
• If `count` is anything other than 1, then "Dimes" is printed
Quiz

• The statement \( if \ (x < 0) \ y = x; \ else \ y = 0; \) can be rewritten using a conditional operator as:

A) \( y = (x < 0) \ ? \ x : 0; \)
B) \( x = (x < 0) \ ? \ y : 0; \)
C) \( (x < 0) \ ? \ y = x : y = 0; \)
D) \( y = (x < 0); \)
E) \( y = if \ (x < 0) \ x : 0; \)
Quiz

• The statement \( if \ (x < 0) \ y = x; \ else \ y = 0; \) can be rewritten using a conditional operator as:

A) \( y = (x < 0) \ ? \ x : 0; \)
B) \( x = (x < 0) \ ? \ y : 0; \)
C) \( (x < 0) \ ? \ y = x : y = 0; \) \( \leftarrow \) Remember the conditional operator is not a \textbf{statement}
D) \( y = (x < 0); \)
E) \( y = if \ (x < 0) \ x : 0; \)
Express the following logic in a succinct manner using the conditional operator.

```java
if (val <= 10)
    System.out.println("It is not greater than 10.");
else
    System.out.println("It is greater than 10.");
```
Express the following logic in a succinct manner using the conditional operator.

```java
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.");
```
Quiz

• What is the difference between a conditional operator and a conditional statement?
Quiz

• What is the difference between a conditional operator and a conditional statement?

• A conditional operator is a ternary operator that evaluates a condition and produces one of two possible values.

• A conditional statement (e.g., if) is a category of statements that allow conditions to be evaluated and the appropriate statements executed as a result.
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
The do Statement

• A *do statement* has the following syntax:

```java
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
Logic of a do Loop

- **Condition Evaluated**: The condition is evaluated first.
- **Statement**: If the condition is true, the statement is executed. If the condition is false, the loop exits.

- Flow Diagram:
  - **True Path**: From condition evaluated to statement.
  - **False Path**: From condition evaluated to false.
The do Statement

• An example of a do loop:

```java
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
import java.util.Scanner;

public class ReverseNumber
{
    public static void main (String[] args)
    {
        int number, lastDigit, reverse = 0;

        Scanner scan = new Scanner (System.in);

        continue
```java
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);
```
```java
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);
```

Sample Run

Enter a positive integer: 2896
That number reversed is 6982
Comparing while and do

The **while** Loop

- condition evaluated
- true
- statement
- false

The **do** Loop

- statement
- true
- condition evaluated
- false
Quiz

How many times will the following loop iterate?

```java
int x = 10;
do {
    System.out.println(x);
    x--;
} while (x > 0);
```
Quiz

How many times will the following loop iterate?

```java
int x = 10;
do {
    System.out.println(x);
    x--;
} while (x > 0);
```

10 times
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
The for Statement

• A *for statement* has the following syntax:

```plaintext
for ( initialization ; condition ; increment )
  statement;
```

- **initialization** is executed once before the loop begins.
- **statement** is executed until the **condition** becomes false.
- **increment** portion is executed at the end of each iteration.
Logic of a for loop

1. **Initialization**
   - Code is executed once before the loop starts.

2. **Condition Evaluation**
   - The condition is checked before the loop body.
   - If the condition is `true`, the loop body is executed.
   - If the condition is `false`, the loop ends.

3. **Statement**
   - The loop body is executed.

4. **Increment**
   - The loop counter is updated or modified.

The process repeats until the condition becomes `false`.
Example of a for loop

- i is initialized to 0
- i is compared to 5
  - if this is true x is incremented with i
- i is incremented by 1
- i is compared again with 5
  - ...

```c
  for (i=0; i<5; i++)
    x += i;
```
The for Statement

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

```plaintext
initialization;
while ( condition )
{
    statement;
    increment;
}
```
The for Statement

• An example of a for loop:

   ```java
   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
The for Statement

- The increment section can perform any calculation:

```java
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
    }
}
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();
}
}
```java
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 + "	");
    // 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
```
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();
        }
    }
}
public class Stars
{
    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();
        }
    }
}
Quiz

Given that s is a String, what does the following loop do?

```java
for (int j = s.length(); j > 0; j--)
    System.out.print(s.charAt(j-1));
```

A) it prints s out backwards
B) it prints s out forwards
C) it prints s out backwards after skipping the last character
D) it prints s out backwards but does not print the 0th character
E) it yields a run-time error because there is no character at s.charAt(j-1) for j = 0
Quiz

Given that s is a String, what does the following loop do?

```java
for (int j = s.length(); j > 0; j--)
    System.out.print(s.charAt(j-1));
```

A) it prints s out backwards
B) it prints s out forwards
C) it prints s out backwards after skipping the last character
D) it prints s out backwards but does not print the 0th character
E) it yields a run-time error because there is no character at s.charAt(j-1) for j = 0
Quick Check

Write a code fragment that rolls a die 100 times and counts the number of times a 3 comes up. Use the `for` loop!
Quick Check

Write a code fragment that rolls a die 100 times and counts the number of times a 3 comes up. Use the **for** loop!

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

Copyright © 2012 Pearson Education, Inc.
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:

```java
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` (from the `Iterable`) 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.

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

• Transform the following while loop in a program with the same semantic but using for

```java
while (x > 2) {
    System.out.println(x);
    x++;
}
```
Quiz

• Transform the following while loop in a program with the same semantic but using for

```java
while (x > 2) {
    system.out.println(x);
    x++;
}
```

```java
for(;x > 2; x++) {
    system.out.println(x);
}
```
Outline

The `switch` Statement
The Conditional Operator
The `do` Statement
The `for` Statement
Drawing with Loops and Conditionals
Dialog Boxes
Drawing Techniques

• Conditionals and loops enhance our ability to generate interesting graphics

• See Bullseye.java
• See BullseyePanel.java

• See Boxes.java
• See BoxesPanel.java
import javax.swing.JFrame;

public class Bullseye
{
    public static void main (String[] args)
    {
        JFrame frame = new JFrame ("Bullseye");
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

        BullseyePanel panel = new BullseyePanel();

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JFrame;

public class Bullseye {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Bullseye");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        BullseyePanel panel = new BullseyePanel();
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JPanel;
import java.awt.*;

public class BullseyePanel extends JPanel
{
    private final int MAX_WIDTH = 300, NUM_RINGS = 5, RING_WIDTH = 25;

    // Sets up the bullseye panel.
    public BullseyePanel ()
    {
        setBackground (Color.cyan);
        setPreferredSize (new Dimension(300,300));
    }
}

continue
public void paintComponent(Graphics page) {
    super.paintComponent(page);
    int x = 0, y = 0, diameter = MAX_WIDTH;
    page.setColor(Color.white);

    for (int count = 0; count < NUM_RINGS; count++) {
        if (page.getColor() == Color.black) // alternate colors
            page.setColor(Color.white);
        else
            page.setColor(Color.black);

        page.fillOval(x, y, diameter, diameter);
        diameter -= (2 * RING_WIDTH);
        x += RING_WIDTH;
        y += RING_WIDTH;
    }

    // Draw the red bullseye in the center
    page.setColor(Color.red);
    page.fillOval(x, y, diameter, diameter);
}
import javax.swing.JFrame;

public class Boxes {
    public static void main (String[] args) {
        JFrame frame = new JFrame("Boxes");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        BoxesPanel panel = new BoxesPanel();

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JFrame;

public class Boxes {
    public static void main (String[] args) {
        JFrame frame = new JFrame("Boxes");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        BoxesPanel panel = new BoxesPanel();
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
import javax.swing.JPanel;
import java.awt.*;
import java.util.Random;

public class BoxesPanel extends JPanel {
    private final int NUM_BOXES = 50, THICKNESS = 5, MAX_SIDE = 50;
    private final int MAX_X = 350, MAX_Y = 250;
    private Random generator;

    //--                               
    //  Sets up the drawing panel.  
    //--                               

    public BoxesPanel () {
        generator = new Random();

        setBackground (Color.black);
        setPreferredSize (new Dimension(400, 300));
    }

    continue
public void paintComponent(Graphics page)
{
    super.paintComponent (page);

    int x, y, width, height;

    for (int count = 0; count < NUM_BOXES; count++)
    {
        x = generator.nextInt(MAX_X) + 1;
        y = generator.nextInt(MAX_Y) + 1;

        width = generator.nextInt(MAX_SIDE) + 1;
        height = generator.nextInt(MAX_SIDE) + 1;

        continue
continue

    if (width <= THICKNESS)  // check for narrow box
    {
        page.setColor (Color.yellow);
        page.fillRect (x, y, width, height);
    }
    else
    {
        if (height <= THICKNESS)  // check for short box
        {
            page.setColor (Color.green);
            page.fillRect (x, y, width, height);
        }
        else
        {
            page.setColor (Color.white);
            page.drawRect (x, y, width, height);
        }
    }
}
Outline

The switch Statement
The Conditional Operator
The do Statement
The for Statement
Drawing with Loops and Conditionals
Dialog Boxes
Dialog Boxes

• A *dialog box* is a window that appears on top of any currently active window.

• It may be used to:
  – convey information
  – confirm an action
  – allow the user to enter data
  – pick a color
  – choose a file

• A dialog box usually has a specific, solitary purpose, and the user interaction with it is brief.
Dialog Boxes

• The `JOptionPane` class provides methods that simplify the creation of some types of dialog boxes

• See `EvenOdd.java`

• Specialized dialog boxes for choosing colors and files are covered in Chapter 9
public class EvenOdd
{
    // Determines if the value input by the user is even or odd.
    // Uses multiple dialog boxes for user interaction.
    public static void main (String[] args)
    {
        String numStr, result;
        int num, again;
        continue
continue

do{
    numStr = JOptionPane.showInputDialog("Enter an integer: ");
    num = Integer.parseInt(numStr);

    result = "That number is " + ((num%2 == 0) ? "even" : "odd");

    JOptionPane.showMessageDialog(null, result);
    again = JOptionPane.showConfirmDialog(null, "Do Another?");
} while (again == JOptionPane.YES_OPTION);
continue
do{
    numStr = JOptionPane.showInputDialog ("Enter an integer: ");
    num = Integer.parseInt (numStr);
    result = "That number is " + ((num%2 == 0) ? "even" : "odd");
    JOptionPane.showMessageDialog (null, result);
    again = JOptionPane.showConfirmDialog (null, "Do Another?");}
while(again == JOptionPane.YES_OPTION);
Summary

• Chapter 6 focused on:
  – the switch statement
  – the conditional operator
  – the do loop
  – the for loop
  – drawing with the aid of conditionals and loops
  – dialog boxes