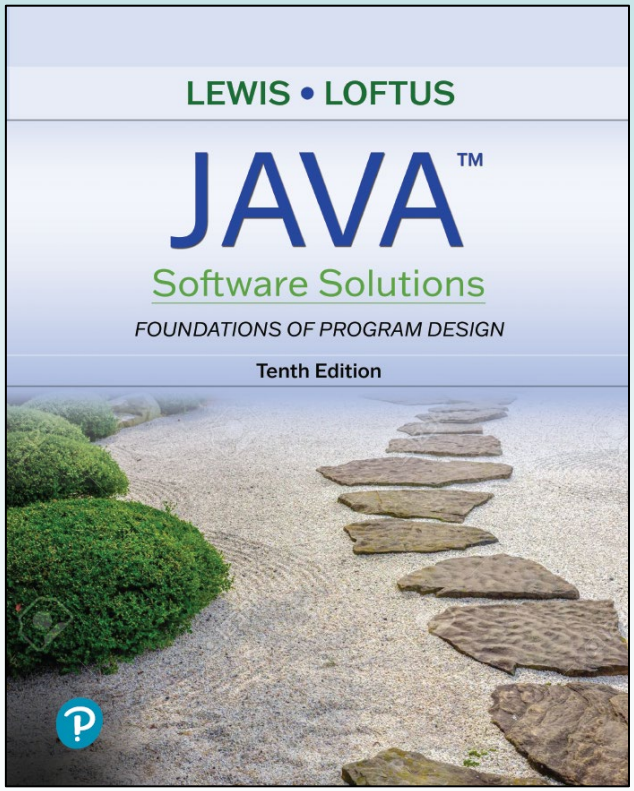
Chapter 6
More Conditionals and Loops



Java Software Solutions

Foundations of Program Design

10th Edition

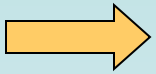
edits: 10/27/2025
08/12/2026

John Lewis
William Loftus

More Conditionals and Loops

- Additional 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**

Switching Yard

The switch Statement

- The *switch statement* provides a way to pick one of several choices
  - Same thing can be done with nested ifs
  - The switch statement is not needed in a language
- The `switch` statement evaluates an expression, then attempts to match the result to one of several *cases*
- Each `case` has a value and a list of statements
- Control transfers to the statement following the first match

The switch Statement

Syntax of a `switch` statement:

switch and *case* are reserved words

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

}
```

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

The switch Statement

- A ***`break`*** *statement* is often the last statement in a 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 continues into the next case
  - Sometimes this is what you want, but usually not
  - A frequent bug

Example switch Statement

```
char option = 'B' ;

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

This will increment only bCount

Example switch Statement

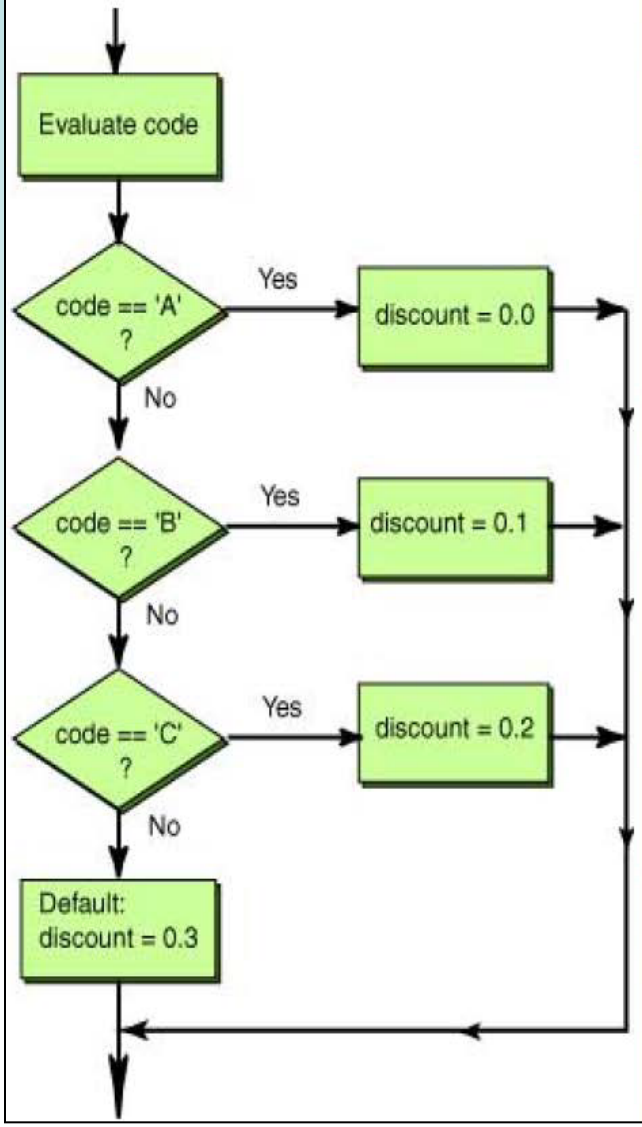
```
char option = 'B' ;

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

This will increment `bCount` and then go on to increment `cCount`.

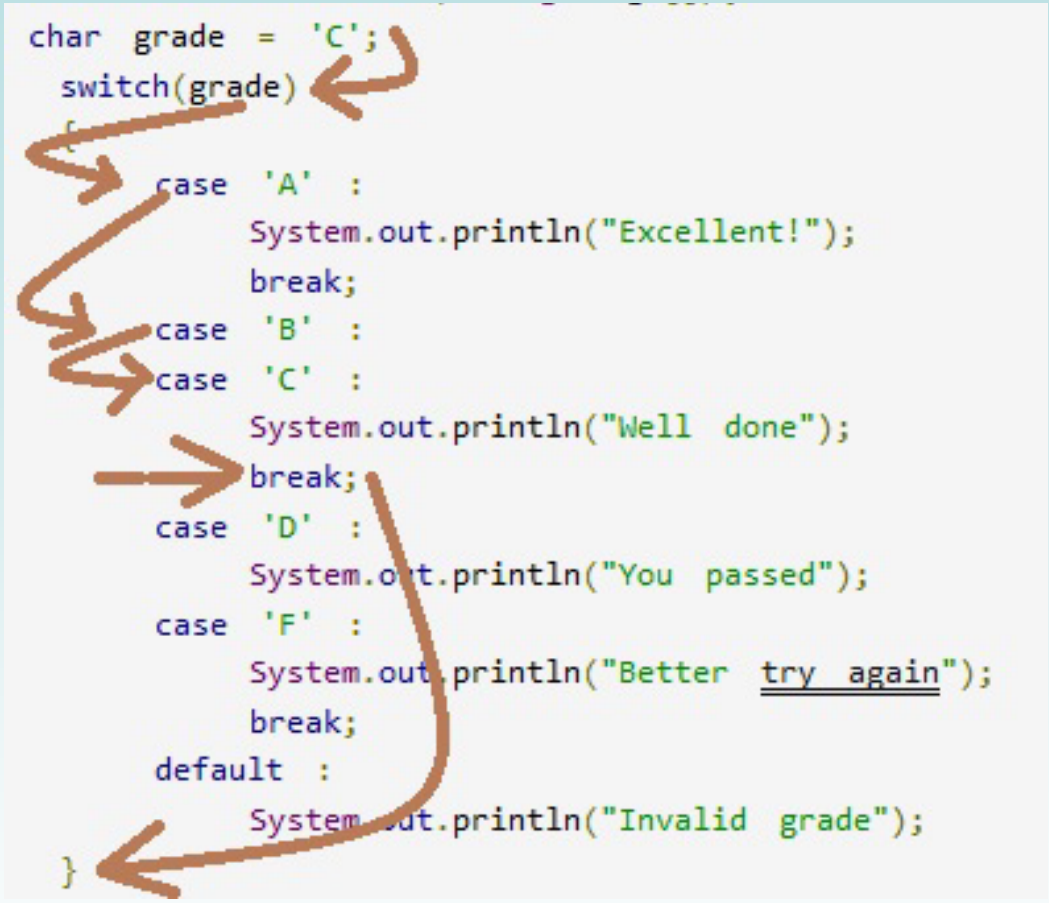
The switch Statement default case

- 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

Flowchart of switch




```
double discount;
// Usually code would come from data
char   code = 'B' ;

switch ( code )
{

  case 'A':
    discount = 0.0;
    break;


  case 'B':
    discount = 0.1;
    break;


  case 'C':
    discount = 0.2;
    break;


  default:
    discount = 0.3;
}

System.out.println( "discount is: " +
```



The switch Statement

- The type of a switch expression must be integers, characters, or enumerated types
- A switch can also be used with strings
- You cannot use a switch with floating point values
- You cannot perform relational checks with a **`switch`** statement
- Further notes: https://chortle.ccsu.edu/javaLessons/chap34/ch34_05.html
- See `GradeReport.java`

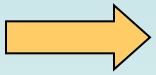
```
//********************************************************************
//  GradeReport.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a switch statement.
//********************************************************************

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;  // integer division

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

Sample Run

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

```
        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.");
        }
     }
}
```

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:

```
condition ? expression1 : expression2
```

- If the **`condition`** is true, **`expression1`** is evaluated; if it is false, **`expression2`** is evaluated
- The value of the entire conditional expression is the value of the selected expression
- Usually best to use an ordinary IF 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

Quick Check

Express the following logic 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.");
```

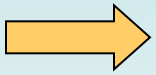
Quick Check

Express the following logic 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**
  - ☺ **Drawing with Loops and Conditionals**
  - ☺ **Dialog Boxes**

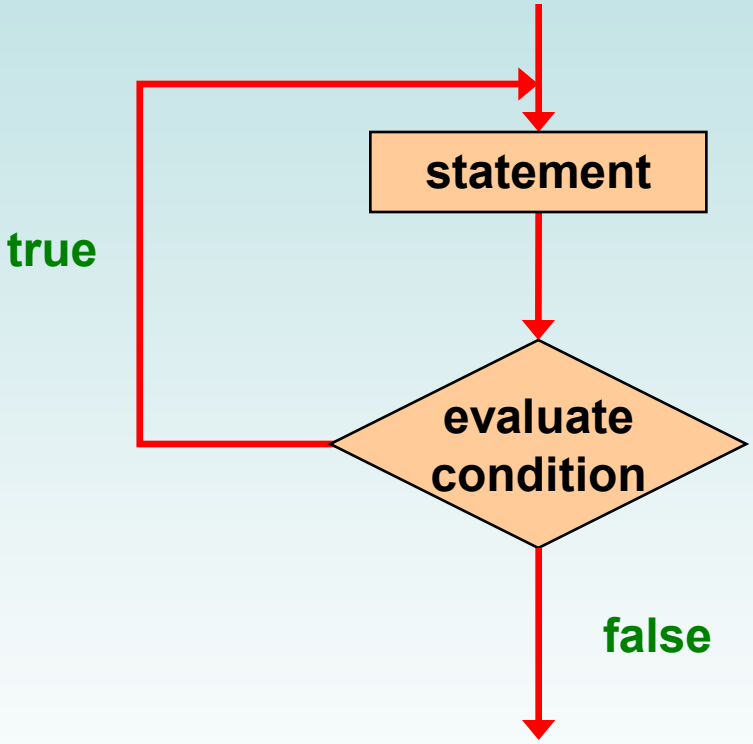
The do Statement

- A *do statement* has the following syntax:

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

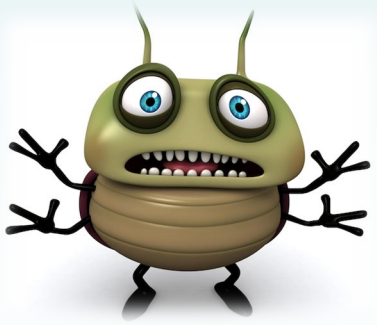
- The ***statement-list*** is always executed once initially, and then the **condition** is evaluated

Logic of a do Loop



The do Statement

- The statement is executed repeatedly until the condition becomes false
- The do statement adds no power to Java
  - anything written with a do can be written with a while (or with a for)
- Frequently a bug



The do Statement

- An example of a `do` loop:

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

Sample Run

```
1
2
3
4
5
```

- The body of a `do` loop executes at least once
- See `ReverseNumber.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);
        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);
    }
}
```

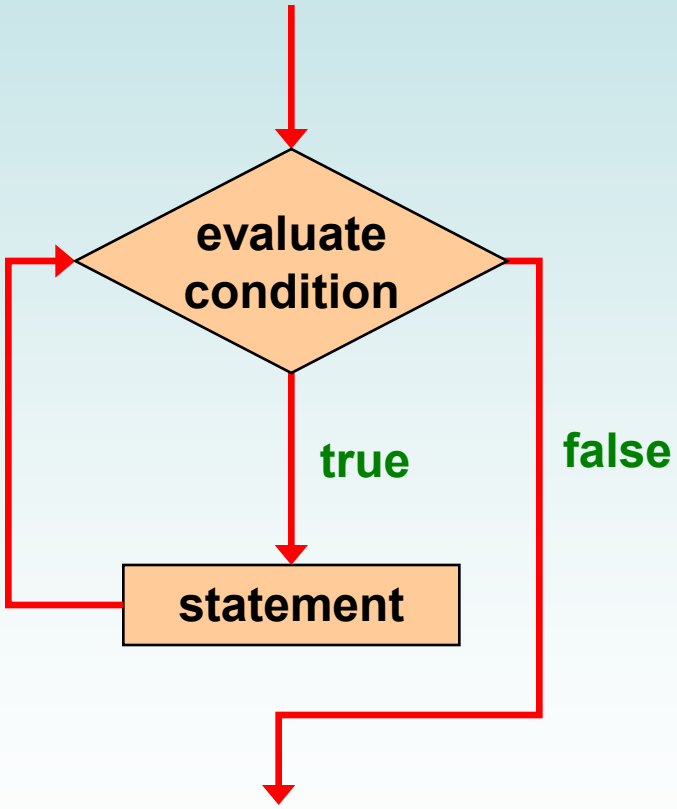
Sample Run

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

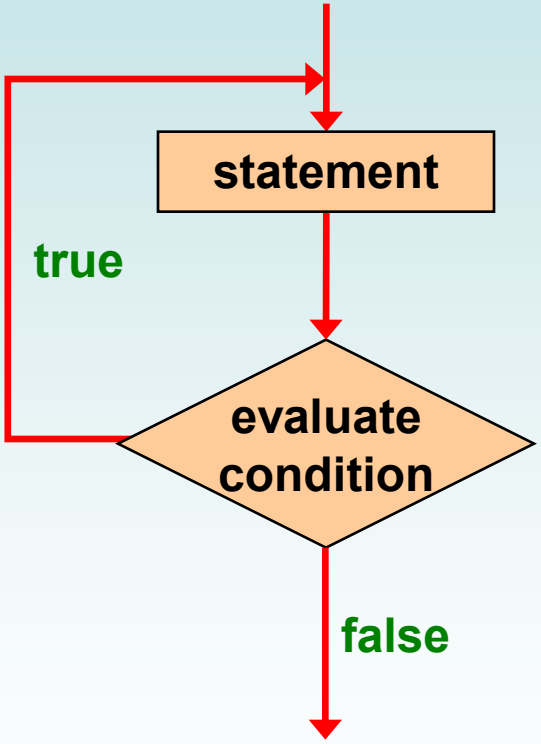
The program could just as easily be written with a while.

Comparing while and do

The while Loop



The do Loop



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:

The *initialization* is executed once before the loop begins

The *statement* is executed until the *condition* becomes false

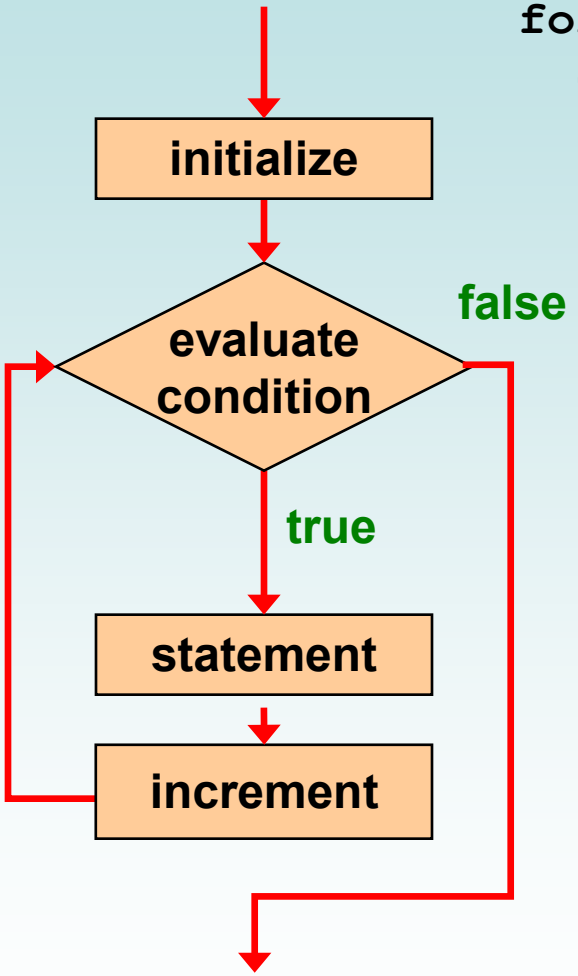
```
for ( initialization ; condition ; increment )
   statement;
```

The *increment* portion is executed at the end of each iteration.

Often *increment* increments a counter, but can be any statement.

All three things a loop must do are contained in one statement.

Logic of a for loop




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

The for Statement

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

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

- It puts the three things that must be coordinated for a loop all in one statement
- It can be used to build any of the three types of loops

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 may execute zero or more times (same is true for `while`)

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 counting loops
- 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();
```

```
        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)     // common trick
              System.out.println();
        }
     }
}
```

The loop just generates a sequence of integers with linefeeds inserted every 5 integers.

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();
      }
   }
}
```

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

public class Stars
{
   //-----------------------------------------------------
   //  Prints a triangle shape using asterisk (star) cha
   //-----------------------------------------------------
   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();
      }
   }
}
```

Output

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

Quick Check

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

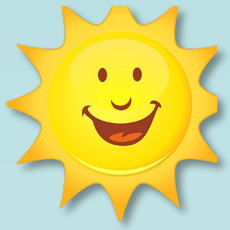
Quick Check

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



For-each Loops

- A variant of the `for` loop simplifies the repetitive processing of items in an iterator
- For example, suppose `bookList` refers to 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 create an iterator and call the `hasNext` and `next` methods explicitly
- Helpful when processing arrays (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);
```

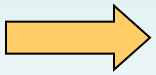
Outline

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

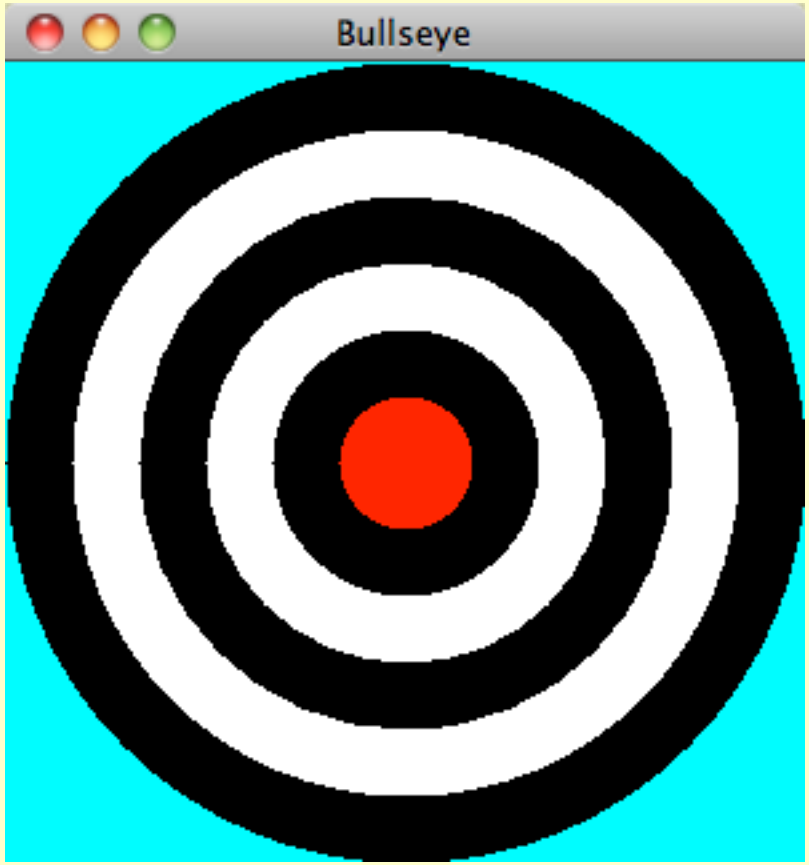
- See `Bullseye.java`
- See `BullseyePanel.java`

- See `Boxes.java`
- See `BoxesPanel.java`

☺

```
//********************************************************************
//  Bullseye.java       Author: Lewis/Loftus
//
//  Demonstrates the use of loops to draw.
//********************************************************************

import javax.swing.JFrame;

public class Bullseye
{
   //-----------------------------------------------------------------
   //  Creates the main frame of the program.
   //-----------------------------------------------------------------
   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);
   }
}
```

```
//************
//  Bullseye.j
//
//  Demonstrat
//************

import javax.s

public class B
{
   //---------
   //  Creates
   //---------
   public stat
   {
      JFrame f
      frame.se

      Bullseye

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```



```
//********************************************************************
//  BullseyePanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of conditionals and loops to guide drawing.
//********************************************************************

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));
   }
```

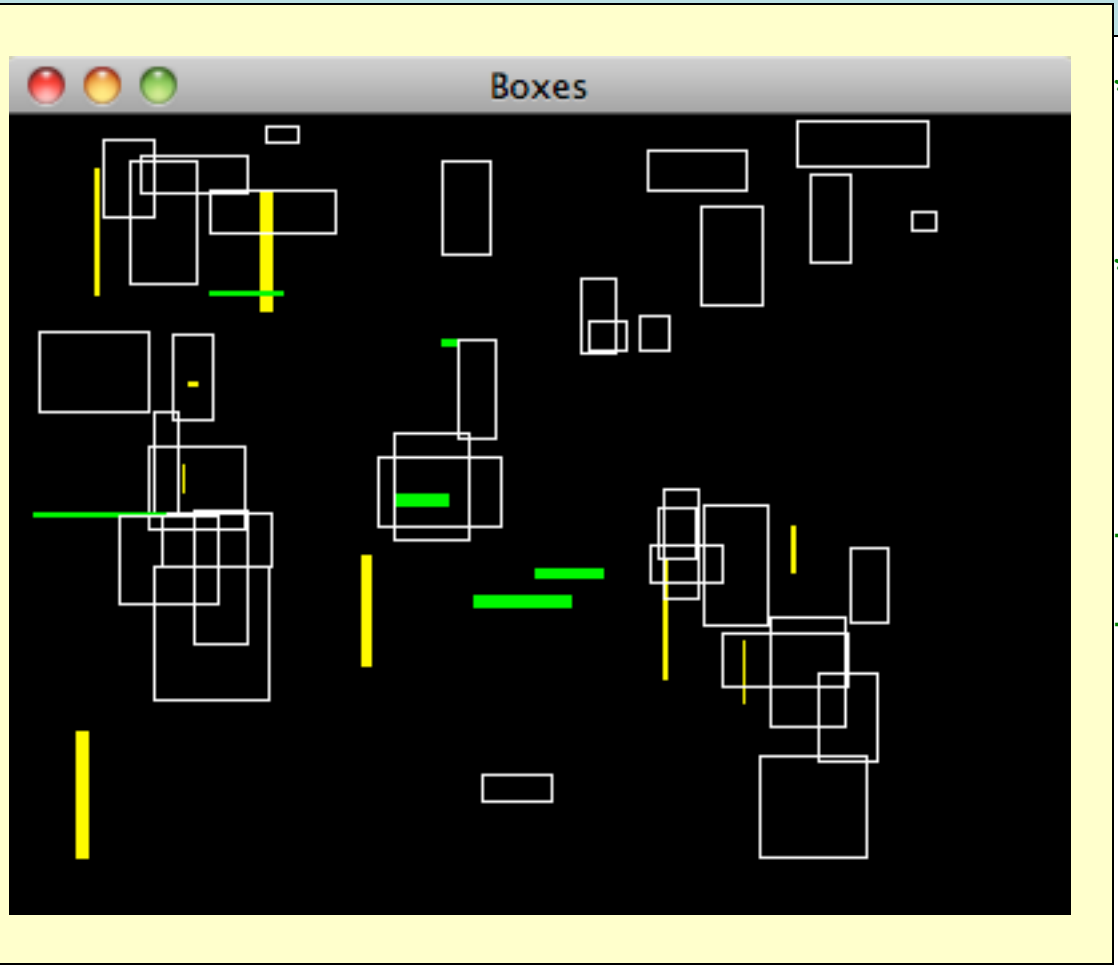
continue

continue

```
    //-----------------------------------------------------------------
    //  Paints a bullseye target.
    //-----------------------------------------------------------------
    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);
    }
}
```

```
//********************************************************************
//  Boxes.java       Author: Lewis/Loftus
//
//  Demonstrates the use of loops to draw.
//********************************************************************

import javax.swing.JFrame;

public class Boxes
{
   //-----------------------------------------------------------------
   //  Creates the main frame of the program.
   //-----------------------------------------------------------------
   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);
   }
}
```

```
//********
//  Boxes.
//
//  Demonst
//********

import java

public clas
{
   //------
   //  Crea
   //------
   public s
   {
      JFrar
      frame

      Boxes

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```



```
//********************************************************************
//  BoxesPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of conditionals and loops to guide drawing.
//********************************************************************

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

continue

```
    //-----------------------------------------------------------------
    //  Paints boxes of random width and height in a random location.
    //  Narrow or short boxes are highlighted with a fill color.
    //-----------------------------------------------------------------
    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

```
/********************************************************************
//  EvenOdd.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the JOptionPane class.
//********************************************************************

import javax.swing.JOptionPane;

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;
```

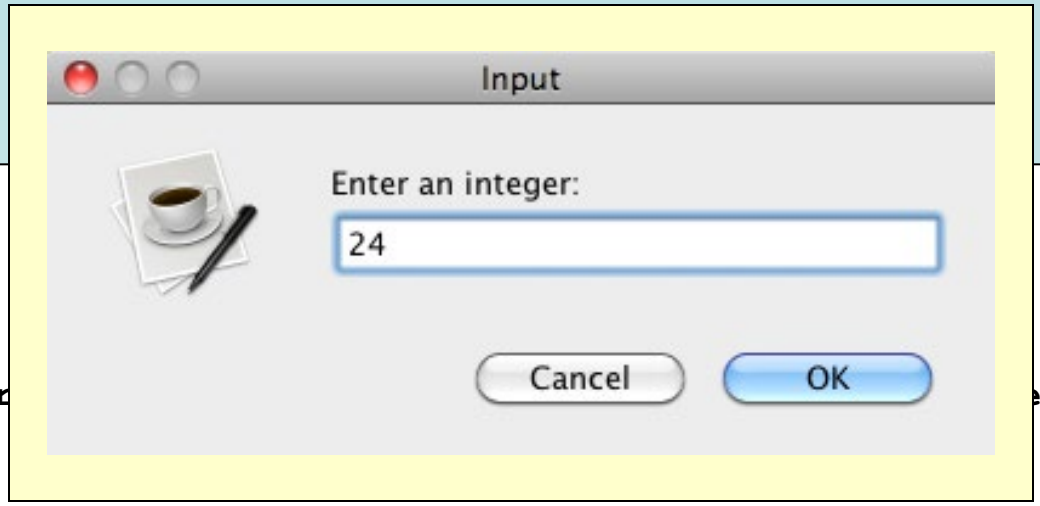
continue

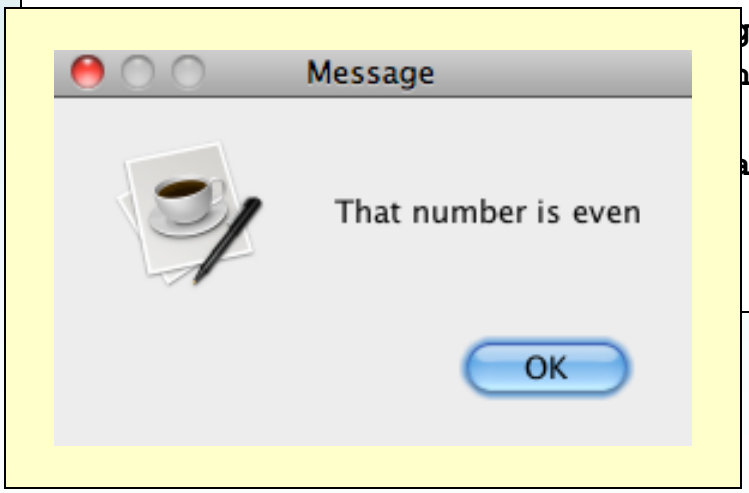
continue

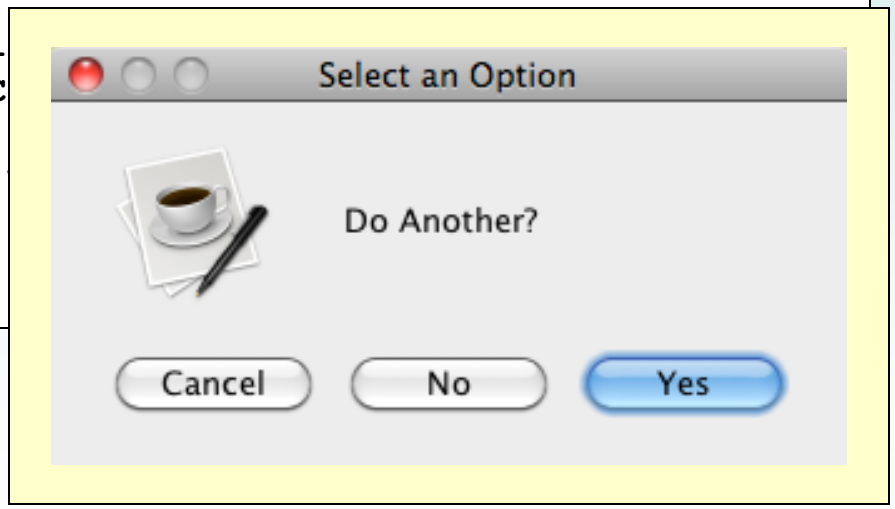
```
        do
        {
            numStr = JOptionPane.showInputDialog("Enter an integer: ");
            num = Integer.parseInt(numStr);

            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                                                    eger: ");
    num =

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







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