*JAVA: An Introduction to Problem Solving & Programming, 7*th Ed. By Walter Savitch.
ISBN 0133862119

Listing 6.1

```
/**
Class for basic pet data: name, age, and
weight.
*/
public class Pet
{
```

```
    private String name;
    private int age; //in years
    private double weight; //in pounds

    public Pet ()   // default constructor
    {
        name = "No name yet.";
        age = 0;
        weight = 0;
    }


    public Pet (String initialName, int
initialAge,
            double initialWeight)
    {
        name = initialName;
        if ((initialAge < 0) ||
(initialWeight < 0))
        {
            System.out.println ("Error:
Negative age or weight.");
            System.exit (0);
        }
        else
        {
            age = initialAge;
            weight = initialWeight;
        }
    }


    public void setPet (String newName, int
newAge,
            double newWeight)
    {
        name = newName;
        if ((newAge < 0) || (newWeight < 0))
        {
            System.out.println ("Error:
Negative age or weight.");
```

```
            System.exit (0);
        }
        else
        {
            age = newAge;
            weight = newWeight;
        }
    }


    public Pet (String initialName)
    {
        name = initialName;
        age = 0;
        weight = 0;
    }


    public void setName (String newName)
    {
        name = newName; //age and weight are
unchanged.
    }


    public Pet (int initialAge)
    {
        name = "No name yet.";
        weight = 0;
        if (initialAge < 0)
        {
            System.out.println ("Error:
Negative age.");
            System.exit (0);
        }
        else
            age = initialAge;
    }


    public void setAge (int newAge)
```

```
    {
        if (newAge < 0)
        {
            System.out.println ("Error:
Negative age.");
            System.exit (0);
        }
        else
            age = newAge;
        //name and weight are unchanged.
    }


    public Pet (double initialWeight)
    {
        name = "No name yet";
        age = 0;
        if (initialWeight < 0)
        {
            System.out.println ("Error:
Negative weight.");
            System.exit (0);
        }
        else
            weight = initialWeight;
    }


    public void setWeight (double newWeight)
    {
        if (newWeight < 0)
        {
            System.out.println ("Error:
Negative weight.");
            System.exit (0);
        }
        else
            weight = newWeight; //name and
age are unchanged.
    }
```

```
    public String getName ()
    {
        return name;
    }


    public int getAge ()
    {
        return age;
    }


    public double getWeight ()
    {
        return weight;
    }


    public void writeOutput ()
    {
        System.out.println ("Name: " +
name);
        System.out.println ("Age: " + age +
" years");
        System.out.println ("Weight: " +
weight + " pounds");
    }
}
```

Listing 6.2

```
import java.util.Scanner;
public class PetDemo
{
    public static void main (String [] args)
```

```
    {
        Pet yourPet = new Pet ("Jane Doe");
        System.out.println ("My records on
your pet are inaccurate.");
        System.out.println ("Here is what
they currently say:");
        yourPet.writeOutput ();
        Scanner keyboard = new Scanner
(System.in);
        System.out.println ("Please enter
the correct pet name:");
        String correctName =
keyboard.nextLine ();
        yourPet.setName (correctName);
        System.out.println ("Please enter
the correct pet age:");
        int correctAge = keyboard.nextInt
();
        yourPet.setAge (correctAge);
        System.out.println ("Please enter
the correct pet weight:");
        double correctWeight =
keyboard.nextDouble ();
        yourPet.setWeight (correctWeight);
        System.out.println ("My updated
records now say:");
        yourPet.writeOutput ();
    }
}
```

Listing 6.3

```
/**
Revised class for basic pet data: name, age,
and weight.
*/
public class Pet2
{
```

```
    private String name;
    private int age; //in years
    private double weight; //in pounds

    public Pet2 (String initialName, int
initialAge,
            double initialWeight)
    {
        set (initialName, initialAge,
initialWeight);
    }


    public Pet2 (String initialName)
    {
        set (initialName, 0, 0);
    }


    public Pet2 (int initialAge)
    {
        set ("No name yet.", initialAge, 0);
    }


    public Pet2 (double initialWeight)
    {
        set ("No name yet.", 0,
initialWeight);
    }


    public Pet2 ()
    {
        set ("No name yet.", 0, 0);
    }


    public void setPet (String newName, int
newAge,
            double newWeight)
```

```
    {
        set (newName, newAge, newWeight);
    }


    public void setName (String newName)
    {
        set (newName, age, weight); //age
and weight unchanged
    }


    public void setAge (int newAge)
    {
        set (name, newAge, weight); //name
and weight unchanged
    }


    public void setWeight (double newWeight)
    {
        set (name, age, newWeight); //name
and age unchanged
    }


    private void set (String newName, int
newAge,
            double newWeight)
    {
        name = newName;
        if ((newAge < 0) || (newWeight < 0))
        {
            System.out.println ("Error:
Negative age or weight.");
            System.exit (0);
        }
        else
        {
            age = newAge;
            weight = newWeight;
```

```
        }
    }


    /*   The methods getName, getAge,
getWeight, and writeOutput are
         the same as in Listing 6.1. >  */
}
```

Listing 6.4

```
/**
Revised class for basic pet data: name, age,
and weight.
*/
public class Pet3
{
    private String name;
    private int age; //in years
    private double weight; //in pounds

    public Pet3 (String initialName, int
initialAge,
             double initialWeight)
    {
        set (initialName, initialAge,
initialWeight);
    }


    public Pet3 (String initialName)
    {
        this (initialName, 0, 0);
    }
```

```
    public Pet3 (int initialAge)
    {
        this ("No name yet.", initialAge,
0);
    }


    public Pet3 (double initialWeight)
    {
        this ("No name yet.", 0,
initialWeight);
    }


    public Pet3 ()
    {
        this ("No name yet.", 0, 0);
    }


    /*  The rest of the class is like Pet2
in Listing 6.3.  */
}
```

Listing 6.5

```
/**
Class of static methods to perform dimension
conversions.
*/
public class DimensionConverter
{
    public static final int INCHES_PER_FOOT
= 12;
```

```
    public static double convertFeetToInches
(double feet)
    {
        return feet * INCHES_PER_FOOT;
    }


    public static double convertInchesToFeet
(double inches)
    {
        return inches / INCHES_PER_FOOT;
    }
}
```

Listing 6.6

```
import java.util.Scanner;
/**
Demonstration of using the class
DimensionConverter.
*/
public class DimensionConverterDemo
{
    public static void main (String [] args)
    {
        Scanner keyboard = new Scanner
(System.in);
        System.out.println ("Enter a
measurement in inches: ");
        double inches = keyboard.nextDouble
();
        double feet =

DimensionConverter.convertInchesToFeet
(inches);
```

```
        System.out.println (inches + "
inches = " +
                  feet + " feet.");
        System.out.print ("Enter a
measurement in feet: ");
        feet = keyboard.nextDouble ();
        inches =
DimensionConverter.convertFeetToInches
(feet);
        System.out.println (feet + " feet =
" +
                  inches + " inches.");
    }
}
```

Listing 6.7

```
import java.util.Scanner;
/**
Class with static and nonstatic members.
*/
public class SavingsAccount
{
    private double balance;
    public static double interestRate = 0;
    public static int numberOfAccounts = 0;

    public SavingsAccount ()
    {
        balance = 0;
        numberOfAccounts++;
    }


    public static void setInterestRate
(double newRate)
```

```
    {
        interestRate = newRate;
    }


    public static double getInterestRate ()
    {
        return interestRate;
    }


    public static double getNumberOfAccounts
()
    {
        return numberOfAccounts;
    }


    public void deposit (double amount)
    {
        balance = balance + amount;
    }


    public double withdraw (double amount)
    {
        if (balance >= amount)
            balance = balance - amount;
        else
            amount = 0;
        return amount;
    }


    public void addInterest ()
    {
        double interest = balance *
interestRate;
        // you can replace interestRate with
getInterestRate()
        balance = balance + interest;
```

```
    }


    public double getBalance ()
    {
        return balance;
    }


    public static void showBalance
(SavingsAccount account)
    {
        System.out.print (account.getBalance
());
    }
}
```

Listing 6.8

```
public class SavingsAccountDemo
{
    public static void main (String [] args)
    {
        SavingsAccount.setInterestRate
(0.01);
        SavingsAccount mySavings = new
SavingsAccount ();
        SavingsAccount yourSavings = new
SavingsAccount ();
        System.out.println ("I deposited
$10.75.");
        mySavings.deposit (10.75);
        System.out.println ("You deposited
$75.");
        yourSavings.deposit (75.00);
```

```
        System.out.println ("You deposited
$55.");
        yourSavings.deposit (55.00);
        double cash = yourSavings.withdraw
(15.00);
        System.out.println ("You withdrew $"
+ cash + ".");
        if (yourSavings.getBalance () >
100.00)
        {
            System.out.println ("You
received interest.");
            yourSavings.addInterest ();
        }
        System.out.println ("Your savings is
$" +
                yourSavings.getBalance ());
        System.out.print ("My savings is
$");
        SavingsAccount.showBalance
(mySavings);
        System.out.println ();
        int count =
SavingsAccount.getNumberOfAccounts ();
        System.out.println ("We opened " +
count
                " savings accounts today.");
    }
}
```

Listing 6.9

```
public class SpeciesEqualsDemo
{
    public static void main (String [] args)
```

```
    {
        Species s1 = new Species (), s2 =
new Species ();
        s1.setSpecies ("Klingon Ox", 10,
15);
        s2.setSpecies ("Klingon Ox", 10,
15);
        System.out.println ("Now change one
Klingon Ox.");
        s2.setSpecies ("klingon ox", 10,
15); //Use lowercase
    }

    if (s1 == s2)
        System.out.println ("Match with
==.");
    else
        System.out.println ("Do Not match
with ==.");

    if (s1.equals (s2))
        System.out.println ("Match with the
method equals.");
    else
        System.out.println ("Do Not match
with the method equals.");

    System.out.println ("Now change one
Klingon Ox.");
    s2.setSpecies ("klingon ox", 10, 15);
//Use lowercase

    if (s1.equals (s2))
        System.out.println ("Match with the
method equals.");
    else
        System.out.println ("Do Not match
with the method equals.");

}
```

Listing 6.10

```
public class SpeciesEqualsDemo
{
    public static void main (String [] args)
    {
        Species s1 = new Species (), s2 =
new Species ();
        s1.setSpecies ("Klingon Ox", 10,
15);
        s2.setSpecies ("Klingon Ox", 10,
15);
        testEqualsOperator (s1, s2);
        testEqualsMethod (s1, s2);
        System.out.println ("Now change one
Klingon Ox.");
        s2.setSpecies ("klingon ox", 10,
15); //Use lowercase
        testEqualsMethod (s1, s2);
    }


    private static void testEqualsOperator
(Species s1, Species s2)
    {
        if (s1 == s2)
            System.out.println ("Match with
==.");
        else
            System.out.println ("Do Not
match with ==.");
    }


    private static void testEqualsMethod
(Species s1, Species s2)
    {
        if (s1.equals (s2))
```

```
            System.out.println ("Match with
the method equals.");
        else
            System.out.println ("Do Not
match with the method equals.");
    }
}
```

Listing 6.11

```
import java.util.Scanner;
public class Species
{
    private String name;
    private int population;
    private double growthRate;

    public void readInput ()
    {
        Scanner keyboard = new Scanner
(System.in);
        System.out.println ("What is the
species' name?");
        name = keyboard.nextLine ();
        System.out.println (
                "What is the population of
the species?");
        population = keyboard.nextInt ();
        while (population < 0)
        {
            System.out.println ("Population
cannot be negative.");
            System.out.println ("Reenter
population:");
            population = keyboard.nextInt
();
```

```
        }
        System.out.println (
                "Enter growth rate (%
increase per year):");
        growthRate = keyboard.nextDouble ();
    }


    public void writeOutput ()
    {
        System.out.println ("Name = " +
name);
        System.out.println ("Population = "
+ population);
        System.out.println ("Growth rate = "
+ growthRate + "%");
    }


    /**
    Precondition: years is a nonnegative
number.
    Returns the projected population of the
receiving object
    after the specified number of years.
    */
    public int predictPopulation (int years)
    {
        int result = 0;
        double populationAmount =
population;
        int count = years;
        while ((count > 0) &&
(populationAmount > 0))
        {
            populationAmount =
(populationAmount +
                    (growthRate / 100) *
populationAmount);
            count - - ;
        }
```

```
        if (populationAmount > 0)
            result = (int) populationAmount;
        return result;
    }


    public void setSpecies (String newName,
int newPopulation,
            double newGrowthRate)
    {
        name = newName;
        if (newPopulation >= 0)
            population = newPopulation;
        else
        {
            System.out.println ("ERROR:
using a negative population.");
            System.exit (0);
        }
        growthRate = newGrowthRate;
    }


    public String getName ()
    {
        return name;
    }


    public int getPopulation ()
    {
        return population;
    }


    public double getGrowthRate ()
    {
        return growthRate;
    }
```

```
    public boolean equals (Species
otherObject)
    {
        return (name.equalsIgnoreCase
(otherObject.name)) &&
            (population ==
otherObject.population) &&
            (growthRate ==
otherObject.growthRate);
    }

    public static void main (String [] args)
    {
        Species speciesToday = new Species
();
        System.out.println ("Enter data on
today's species:");
        speciesToday.readInput ();
        speciesToday.writeOutput ();
        System.out.println ("Enter number of
years to project:");
        Scanner keyboard = new Scanner
(System.in);
        int numberOfYears = keyboard.nextInt
();
        int futurePopulation =
            speciesToday.predictPopulation
(numberOfYears);
        System.out.println ("In " +
numberOfYears +
                " years the population will
be " +
                futurePopulation);
        speciesToday.setSpecies ("Klingon
ox", 10, 15);
        System.out.println ("The new species
is:");
        speciesToday.writeOutput ();
    }
}
```

Listing 6.12

```
public class DollarFormatFirstTry
{
    /**
    Displays amount in dollars and cents
notation.
    Rounds after two decimal places.
    Does not advance to the next line after
output.
    */
    public static void write (double amount)
    {
        int allCents = (int) (Math.round
(amount * 100));
        int dollars = allCents / 100;
        int cents = allCents % 100;
        System.out.print ('$');
        System.out.print (dollars);
        System.out.print ('.');
        if (cents < 10)
        {
            System.out.print ('0');
            System.out.print (cents);
        }
        else
            System.out.print (cents);
    }


    /**
    Displays amount in dollars and cents
notation.
    Rounds after two decimal places.
    Advances to the next line after output.
    */
    public static void writeln (double
amount)
```

```
    {
        write (amount);
        System.out.println ();
    }
}
```

Listing 6.13

```
import java.util.Scanner;
public class DollarFormatFirstTryDriver
{
    public static void main (String [] args)
    {
        double amount;
        String response;
        Scanner keyboard = new Scanner
(System.in);
        System.out.println (
                "Testing
DollarFormatFirstTry.write:");
        do
        {
            System.out.println ("Enter a
value of type double:")
                amount = keyboard.nextDouble
();
            DollarFormatFirstTry.write
(amount);
            System.out.println ();
            System.out.println ("Test
again?");
            response = keyboard.next ();
        }
        while (response.equalsIgnoreCase
("yes"));
        System.out.println ("End of test.");
```

```
    }
}
```

Listing 6.14

```
public class DollarFormat
{
    /**
    Displays amount in dollars and cents
notation.
    Rounds after two decimal places.
    Does not advance to the next line after
output.
    */
    public static void write (double amount)
    {
        if (amount >= 0)
        {
            System.out.print ('$');
            writePositive (amount);
        }
        else
        {
            double positiveAmount = -amount;
            System.out.print ('$');
            System.out.print ('-');
            writePositive (positiveAmount);
        }
    }

    //Precondition: amount >= 0;
    //Displays amount in dollars and cents
notation. Rounds
    //after two decimal places. Omits the
dollar sign.
```

```
    private static void writePositive
(double amount)
    {
        int allCents = (int) (Math.round
(amount * 100));
        int dollars = allCents / 100;
        int cents = allCents % 100;
        System.out.print (dollars);
        System.out.print ('.');
    }

    if (cents < 10)
        System.out.print ('0');
    System.out.print (cents)
        /**
        Displays amount in dollars and cents
notation.
        Rounds after two decimal places.
        Advances to the next line after
output.
        */
        public static void writeln (double
amount)
    {
        write (amount);
        System.out.println ();
    }
}
```

Listing 6.15

```
/**
This class illustrates overloading.
*/
public class Overload
{
```

```
    public static void main (String [] args)
    {
        double average1 =
Overload.getAverage (40.0, 50.0);
        double average2 =
Overload.getAverage (1.0, 2.0, 3.0);
        char average3 = Overload.getAverage
('a', 'c');
        System.out.println ("average1 = " +
average1);
        System.out.println ("average2 = " +
average2);
        System.out.println ("average3 = " +
average3);
    }


    public static double getAverage (double
first, double second)
    {
        return (first + second) / 2.0;
    }


    public static double getAverage (double
first, double second,
            double third)
    {
        return (first + second + third) /
3.0;
    }


    public static char getAverage (char
first, char second)
    {
        return (char) (((int) first + (int)
second) / 2);
    }
}
```

Listing 6.16

```
import java.util.Scanner;
/**
Class representing nonnegative amounts of
money,
such as $100, $41.99, $0.05.
*/
public class Money
{
    private long dollars;
    private long cents;
    public void set (long newDollars)
    {
        if (newDollars < 0)
        {
            System.out.println (
                    "Error: Negative amounts
of money are not allowed.");
            System.exit (0);
        }
        else
        {
            dollars = newDollars;
            cents = 0;
        }
    }


    public void set (double newAmount)
    {
        if (newAmount < 0)
        {
            System.out.println (
                    "Error: Negative amounts
of money are not allowed.");
            System.exit (0);
        }
```

```
        else
        {
            long allCents = Math.round
(newAmount * 100);
            dollars = allCents / 100;
            cents = allCents % 100;
        }
    }


    public void set (Money moneyObject)
    {
        this.dollars = moneyObject.dollars;
        this.cents = moneyObject.cents;
    }


    /**
    Precondition: The argument is an
ordinary representation
    of an amount of money, with or without a
dollar sign.
    Fractions of a cent are not allowed.
    */
    public void set (String amountString)
    {
        String dollarsString;
        String centsString;
        //Delete '$' if any:
        if (amountString.charAt (0) == '$')
            amountString =
amountString.substring (1);
        amountString = amountString.trim ();
        //Locate decimal point:
        int pointLocation =
amountString.indexOf (".");
        if (pointLocation < 0) //If no
decimal point
        {
            cents = 0;
```

```
            dollars = Long.parseLong
(amountString);
        }
        else //String has a decimal point.
        {
            dollarsString =
                amountString.substring (0,
pointLocation);
            centsString =
                amountString.substring
(pointLocation + 1);
            //one digit in cents means
tenths of a dollar
            if (centsString.length () <= 1)
                centsString = centsString +
"0";
            // convert to numeric
            dollars = Long.parseLong
(dollarsString);
            cents = Long.parseLong
(centsString);
            if ((dollars < 0) || (cents < 0)
|| (cents > 99))
            {
                System.out.println (
                        "Error: Illegal
representation of money.");
                System.exit (0);
            }
        }
    }


    public void readInput ()
    {
        System.out.println ("Enter amount on
a line by itself:");
        Scanner keyboard = new Scanner
(System.in);
        String amount = keyboard.nextLine
();
```

```
        set (amount.trim ());
    }


    /**
    Does not go to the next line after
displaying money.
    */
    public void writeOutput ()
    {
        System.out.print ("$" + dollars);
        if (cents < 10)
            System.out.print (".0" + cents);
        else
            System.out.print ("." + cents);
    }


    /**
    Returns n times the calling object.
    */
    public Money times (int n)
    {
        Money product = new Money ();
        product.cents = n * cents;
        long carryDollars = product.cents /
100;
        product.cents = product.cents % 100;
        product.dollars = n * dollars +
carryDollars;
        return product;
    }


    /**
    Returns the sum of the calling object
and the argument.
    */
    public Money add (Money otherAmount)
    {
        Money sum = new Money ();
```

```
        sum.cents = this.cents +
otherAmount.cents;
        long carryDollars = sum.cents / 100;
        sum.cents = sum.cents % 100;
        sum.dollars = this.dollars
            + otherAmount.dollars +
carryDollars;
        return sum;
    }
}
```

Listing 6.17

```
public class MoneyDemo
{
    public static void main (String [] args)
    {
        Money start = new Money ();
        Money goal = new Money ();
        System.out.println ("Enter your
current savings:");
        start.readInput ();
        goal = start.times (2);
        System.out.print (
                "If you double that, you
will have ");
        goal.writeOutput ();
        System.out.println (", or better
yet:");
        goal = start.add (goal);
        System.out.println (
                "If you triple that original
amount, you will have:");
        goal.writeOutput ();
        System.out.println ();
```

```
        System.out.println ("Remember: A
penny saved");
        System.out.println ("is a penny
earned.");
    }
}
```

Listing 6.18

```
/**
Class whose privacy can be breached.
*/
public class PetPair
{
    private Pet first, second;

    public PetPair (Pet firstPet, Pet
secondPet)
    {
        first = firstPet;
        second = secondPet;
    }


    public Pet getFirst ()
    {
        return first;
    }


    public Pet getSecond ()
    {
        return second;
    }
```

```
    public void writeOutput ()
    {
        System.out.println ("First pet in
the pair:");
        first.writeOutput ();
        System.out.println ("\nSecond pet in
the pair:");
        second.writeOutput ();
    }
}
```

Listing 6.19

```
/**
Toy program to demonstrate how a programmer
can access and
change private data in an object of the
class PetPair.
*/
public class Hacker
{
    public static void main (String [] args)
    {
        Pet goodDog = new Pet ("Faithful
Guard Dog", 5, 75.0);
        Pet buddy = new Pet ("Loyal
Companion", 4, 60.5);
        PetPair pair = new PetPair (goodDog,
buddy);
        System.out.println ("Our pair:");
        pair.writeOutput ();
        Pet badGuy = pair.getFirst ();
        badGuy.setPet ("Dominion Spy", 1200,
500);
        System.out.println ("\nOur pair
now:");
```

```
        pair.writeOutput ();
        System.out.println ("The pet wasn't
so private!");
        System.out.println ("Looks like a
security breach.");
    }
}
```

Listing 6.20

```
/** An enumeration of card suits. */
enum Suit
{
    CLUBS ("black"), DIAMONDS ("red"),
HEARTS ("red"),
        SPADES ("black");

    private final String color;

    private Suit (String suitColor)
    {
        color = suitColor;
    }


    public String getColor ()
    {
        return color;
    }
}
```

Listing 6.21

```
import javax.swing.JApplet;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Graphics;
/**
Simple demonstration of adding buttons to an
applet.
These buttons do not do anything. That comes
in a later version.
*/
public class PreliminaryButtonDemo extends
JApplet
{
    public void init ()
    {
        Container contentPane =
getContentPane ();
        contentPane.setBackground
(Color.WHITE);
        contentPane.setLayout (new
FlowLayout ());
        JButton sunnyButton = new JButton
("Sunny");
        contentPane.add (sunnyButton);
        JButton cloudyButton = new JButton
("Cloudy");
        contentPane.add (cloudyButton);
    }
}
```

Listing 6.22

```
import javax.swing.JApplet;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
Simple demonstration of adding buttons to an
applet.
These buttons do something when clicked.
*/
public class ButtonDemo extends JApplet
implements ActionListener
{
    public void init ()
    {
        Container contentPane =
getContentPane ();
        contentPane.setBackground
(Color.WHITE);
        contentPane.setLayout (new
FlowLayout ());
        JButton sunnyButton = new JButton
("Sunny");
        contentPane.add (sunnyButton);
        sunnyButton.addActionListener
(this);
        JButton cloudyButton = new JButton
("Cloudy");
        contentPane.add (cloudyButton);
        cloudyButton.addActionListener
(this);
    }


    public void actionPerformed (ActionEvent
e)
    {
```

```
        Container contentPane =
getContentPane ();
        if (e.getActionCommand ().equals
("Sunny"))
            contentPane.setBackground
(Color.BLUE);
        else if (e.getActionCommand
().equals ("Cloudy"))
            contentPane.setBackground
(Color.GRAY);
        else
            System.out.println ("Error in
button interface.");
    }
}
```

Listing 6.23

```
import javax.swing.ImageIcon;
import javax.swing.JApplet;
import javax.swing.JLabel;
public class IconDemo extends JApplet
{
    public void init ()
    {
        JLabel niceLabel = new JLabel ("Java
is fun!");
        ImageIcon dukeIcon = new ImageIcon
("duke_waving.gif");
        niceLabel.setIcon (dukeIcon);
        getContentPane ().add (niceLabel);
    }
}
```

Listing 6.24

```
import javax.swing.ImageIcon;
import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JLabel;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
Simple demonstration of changing visibility
in an applet.
*/
public class VisibilityDemo extends JApplet
implements ActionListener
{
    private JLabel response;
    private Container contentPane;
    public void init ()
    {
        contentPane = getContentPane ();
        contentPane.setBackground
(Color.WHITE);
        //Create button:
        JButton aButton = new JButton ("Push
me!");
        aButton.addActionListener (this);
        //Create label:
        response = new JLabel ("Thanks. That
felt good!");
```

```
        ImageIcon smileyFaceIcon = new
ImageIcon ("smiley.gif");
        response.setIcon (smileyFaceIcon);
        response.setVisible (false);
//Invisible until button is clicked
        //Add button:
        contentPane.setLayout (new
FlowLayout ());
        contentPane.add (aButton);
        //Add label
        contentPane.add (response);
    }


    public void actionPerformed (ActionEvent
e)
    {
        contentPane.setBackground
(Color.PINK);
        response.setVisible (true); //Show
label
    }
}
```