*JAVA: An Introduction to Problem Solving & Programming, 7*th Ed. By Walter Savitch.
ISBN 0133862119 © 2015 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Listing 4.1

```
import java.util.Scanner;
public class WhileDemo
{
    public static void main (String [] args)
    {
        int count, number;
        System.out.println ("Enter a number");
        Scanner keyboard = new Scanner (System.in);
        number = keyboard.nextInt ();
        count = 1;
        while (count <= number)
        {
            System.out.print (count + ", ");
            count++;
        }
        System.out.println ();
        System.out.println ("Buckle my shoe.");
    }
}
```

Listing 4.2

```
import java.util.Scanner;
public class DoWhileDemo
{
    public static void main (String [] args)
    {
        int count, number;
        System.out.println ("Enter a number");
        Scanner keyboard = new Scanner (System.in);
        number = keyboard.nextInt ();
        count = 1;
        do
        {
            System.out.print (count + ", ");
            count++;
        }
        while (count <= number);
        System.out.println ();
        System.out.println ("Buckle my shoe.");
    }
}
```

Algorithm for Roach Population Program

```
Algorithm for roach population program
1. Read houseVolume
2. Read startPopulation
3. population = startPopulation
4. totalBugVolume = population * ONE_BUG_VOLUME
5. countWeeks = 0
6. while (totalBugVolume < houseVolume)
{
        newBugs = population * GROWTH_RATE
        newBugVolume = newBugs * ONE_BUG_VOLUME
        population = population + newBugs
        totalBugVolume = totalBugVolume + newBugVolume
        countWeeks = countWeeks + 1
}
7. Display startPopulation, houseVolume, countWeeks,
population, and
totalBugVolume
```

Listing 4.3

```
import java.util.Scanner;
/**
Program to calculate how long it will take a population of
roaches to completely fill a house from floor to ceiling.
*/
public class BugTrouble
{
    public static final double GROWTH_RATE = 0.95; //95% per
week
    public static final double ONE_BUG_VOLUME = 0.002;
//cubic feet
    public static void main (String [] args)
    {
        System.out.println ("Enter the total volume of your
house");
        System.out.print ("in cubic feet: ");
        Scanner keyboard = new Scanner (System.in);
        double houseVolume = keyboard.nextDouble ();
        System.out.println ("Enter the estimated number
of");
        System.out.print ("roaches in your house: ");
        int startPopulation = keyboard.nextInt ();
        int countWeeks = 0;
        double population = startPopulation;
        double totalBugVolume = population * ONE_BUG_VOLUME;
        double newBugs, newBugVolume;
        while (totalBugVolume < houseVolume)
        {
            newBugs = population * GROWTH_RATE;
            newBugVolume = newBugs * ONE_BUG_VOLUME;
            population = population + newBugs;
            totalBugVolume = totalBugVolume + newBugVolume;
            countWeeks++;
        }
        System.out.println ("Starting with a roach
population of " +
                startPopulation);
        System.out.println ("and a house with a volume of "
+ houseVolume +
                " cubic feet,");
        System.out.println ("after " + countWeeks + "
weeks,");
```

```
        System.out.println ("the house will be filled with "
+
                (int) population + " roaches.");
        System.out.println ("They will fill a volume of " +
                (int) totalBugVolume + " cubic feet.");
        System.out.println ("Better call Debugging Experts
Inc.");
    }
}
```

Listing 4.4

```
import java.util.Scanner;
/**
Computes the average of a list of (nonnegative) exam scores.
Repeats computation for more exams until the user says to
stop.
*/
public class ExamAverager
{
    public static void main (String [] args)
    {
        System.out.println ("This program computes the
average of");
        System.out.println ("a list of (nonnegative) exam
scores.");
        double sum;
        int numberOfStudents;
        double next;
        String answer;
        Scanner keyboard = new Scanner (System.in);
        do
        {
            System.out.println ();
            System.out.println ("Enter all the scores to be
averaged.");
            System.out.println ("Enter a negative number
after");
            System.out.println ("you have entered all the
scores.");
            sum = 0;
            numberOfStudents = 0;
            next = keyboard.nextDouble ();
            while (next >= 0)
            {
                sum = sum + next;
                numberOfStudents++;
                next = keyboard.nextDouble ();
            }
            if (numberOfStudents > 0)
                System.out.println ("The average is " +
                        (sum / numberOfStudents));
            else
                System.out.println ("No scores to
average.");
            System.out.println ("Want to average another
exam?");
            System.out.println ("Enter yes or no.");
            answer = keyboard.next ();
        }
        while (answer.equalsIgnoreCase ("yes"));
    }
}
```

Listing 4.5

```
public class ForDemo
{
    public static void main (String [] args)
    {
        int countDown;
        for (countDown = 3 ; countDown >= 0 ; countDown--)
        {
            System.out.println (countDown);
            System.out.println ("and counting.");
        }
        System.out.println ("Blast off!");
    }
}
```

Listing 4.6

```
import java.util.Scanner;
/**
Illustrates the use of a boolean variable to end loop
iteration.
*/
public class BooleanDemo
{
    public static void main (String [] args)
    {
        System.out.println ("Enter nonnegative numbers.");
        System.out.println ("Place a negative number at the
end");
        System.out.println ("to serve as an end marker.");
        int sum = 0;
        boolean areMore = true;
        Scanner keyboard = new Scanner (System.in);
        while (areMore)
        {
            int next = keyboard.nextInt ();
            if (next < 0)
                areMore = false;
            else
                sum = sum + next;
        }
        System.out.println ("The sum of the numbers is " +
sum);
    }
}
```

Spending Spree Algorithm

```
    1. amountRemaining = amount of gift certificate
    2. totalSpent = 0
    3. itemNumber = 1
    4. while (we have money left to spend and (itemNumber <=
max number of items))
    {
    Display amount of money left and number of items that
can be bought.
        Read cost of proposed purchase.
        if (we can afford the purchase)
        {
            Display a message.
            totalSpent = totalSpent + cost of item
            Update amountRemaining
            if (amountRemaining > 0)
            {
                Display amount of money left.
                itemNumber++
            }
            else
            {
                Display a message (no more money).
                Make this the last loop iteration.
            }
        }

        else
        Display a message (item is too expensive).
        }

        Display amount of money spent and farewell message.
```

Listing 4.7

```
import java.util.Scanner;
public class SpendingSpree
{
    public static final int SPENDING_MONEY = 100;
    public static final int MAX_ITEMS = 3;
    public static void main (String [] args)
    {
        Scanner keyboard = new Scanner (System.in);
        boolean haveMoney = true;
        int leftToSpend = SPENDING_MONEY;
        int totalSpent = 0;
        int itemNumber = 1;
        while (haveMoney && (itemNumber <= MAX_ITEMS))
        {
            System.out.println ("You may buy up to " +
                    (MAX_ITEMS - itemNumber + 1) +
                    " items");
            System.out.println ("costing no more than $" +
                    leftToSpend + ".");
            System.out.print ("Enter cost of item #" +
                    itemNumber + ": $");
            int itemCost = keyboard.nextInt ();
            if (itemCost <= leftToSpend)
            {
                System.out.println ("You may buy this item.
");
                totalSpent = totalSpent + itemCost;
                System.out.println ("You spent $ +
totalSpent +
                        so far. ");
                        leftToSpend = SPENDING_MONEY -
totalSpent;
                if (leftToSpend > 0)
                    itemNumber++;
                else
                {
                    System.out.println ("You are out of
money.);
                            haveMoney = false;
                }
            }
            else
                System.out.println ("You cannot buy that
item.");
```

```
        }
        System.out.println ("You spent $" + totalSpent +
                ", and are done shopping.");
    }
}
```

Listing 4.9

```
import javax.swing.JApplet;
import java.awt.Graphics;
import java.awt.Color;
public class MultipleFaces extends JApplet
{
    public static final int FACE_DIAMETER = 50;
    public static final int X_FACE0 = 10;
    public static final int Y_FACE0 = 5;
    public static final int EYE_WIDTH = 5;
    public static final int EYE_HEIGHT = 10;
    public static final int X_RIGHT_EYE0 = 20;
    public static final int Y_RIGHT_EYE0 = 15;
    public static final int X_LEFT_EYE0 = 45;
    public static final int Y_LEFT_EYE0 = Y_RIGHT_EYE0;
    public static final int NOSE_DIAMETER = 5;
    public static final int X_NOSE0 = 32;
    public static final int Y_NOSE0 = 25;
    public static final int MOUTH_WIDTH = 30;
    public static final int MOUTH_HEIGHT0 = 0;
    public static final int X_MOUTH0 = 20;
    public static final int Y_MOUTH0 = 35;
    public static final int MOUTH_START_ANGLE = 180;
    public static final int MOUTH_EXTENT_ANGLE = 180;
    public void paint (Graphics canvas)
    {
        super.paint(canvas);
        int i, xOffset, yOffset; //Want i to exist after the
loop ends
        for (i = 0 ; i <= 4 ; i++)
        { //Draw one face:
            xOffset = 50 * i;
            yOffset = 30 * i;
            //Draw face interior and outline:
            if (i % 2 == 0) //if i is even,
            { //Make face yellow
                canvas.setColor (Color.YELLOW);
                canvas.fillOval (X_FACE0 + xOffset, Y_FACE0
+ yOffset,
                        FACE_DIAMETER, FACE_DIAMETER);
            }
            canvas.setColor (Color.BLACK);
            canvas.drawOval (X_FACE0 + xOffset, Y_FACE0 +
yOffset,
                    FACE_DIAMETER, FACE_DIAMETER);
            //Draw eyes:
            canvas.setColor (Color.BLUE);
            canvas.fillOval (X_RIGHT_EYE0 + xOffset,
Y_RIGHT_EYE0 + yOffset,
                    EYE_WIDTH, EYE_HEIGHT);
            canvas.fillOval (X_LEFT_EYE0 + xOffset,
Y_LEFT_EYE0 + yOffset,
                    EYE_WIDTH, EYE_HEIGHT);
            //Draw nose:
            canvas.setColor (Color.BLACK);
            canvas.fillOval (X_NOSE0 + xOffset, Y_NOSE0 +
yOffset,
                    NOSE_DIAMETER, NOSE_DIAMETER);
            //Draw mouth:
            canvas.setColor (Color.RED);
            canvas.drawArc (X_MOUTH0 + xOffset, Y_MOUTH0 +
yOffset,
                    MOUTH_WIDTH, MOUTH_HEIGHT0 + 3 * i,
                    MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE);
        }
        //i is 5 when the previous loop ends
        xOffset = 50 * i;
        yOffset = 30 * i;
        //Draw kissing face:
        //Draw face outline:
        canvas.setColor (Color.BLACK);
        canvas.drawOval (X_FACE0 + xOffset, Y_FACE0 +
yOffset,
                FACE_DIAMETER, FACE_DIAMETER);
        //Draw eyes:
        canvas.setColor (Color.BLUE);
        canvas.fillOval (X_RIGHT_EYE0 + xOffset,
Y_RIGHT_EYE0 + yOffset,
                EYE_WIDTH, EYE_HEIGHT);
        canvas.fillOval (X_LEFT_EYE0 + xOffset, Y_LEFT_EYE0
+ yOffset,
                EYE_WIDTH, EYE_HEIGHT);
        //Draw nose:
        canvas.setColor (Color.BLACK);
        canvas.fillOval (X_NOSE0 + xOffset, Y_NOSE0 +
yOffset,
                NOSE_DIAMETER, NOSE_DIAMETER);
        //Draw mouth in shape of a kiss:
        canvas.setColor (Color.RED);
        canvas.fillOval (X_MOUTH0 + xOffset + 10, Y_MOUTH0 +
yOffset,
                MOUTH_WIDTH - 20, MOUTH_WIDTH - 20);
        //Add text:
        canvas.drawString ("Kiss, Kiss.",
                X_FACE0 + xOffset + FACE_DIAMETER, Y_FACE0 +
yOffset);
        //Draw blushing face:
        i++;
        xOffset = 50 * i;
        yOffset = 30 * i;
        //Draw face interior and outline:
        canvas.setColor (Color.PINK);
        canvas.fillOval (X_FACE0 + xOffset, Y_FACE0 +
yOffset,
                FACE_DIAMETER, FACE_DIAMETER);
        canvas.setColor (Color.BLACK);
        canvas.drawOval (X_FACE0 + xOffset, Y_FACE0 +
yOffset,
                FACE_DIAMETER, FACE_DIAMETER);
        //Draw eyes:
        canvas.setColor (Color.BLUE);
        canvas.fillOval (X_RIGHT_EYE0 + xOffset,
Y_RIGHT_EYE0 + yOffset,
                EYE_WIDTH, EYE_HEIGHT);
        canvas.fillOval (X_LEFT_EYE0 + xOffset, Y_LEFT_EYE0
+ yOffset,
                EYE_WIDTH, EYE_HEIGHT);
        //Draw nose:
        canvas.setColor (Color.BLACK);
        canvas.fillOval (X_NOSE0 + xOffset, Y_NOSE0 +
yOffset,
                NOSE_DIAMETER, NOSE_DIAMETER);
        //Draw mouth: (same as on face before kissing one)
        canvas.setColor (Color.RED);
        canvas.drawArc (X_MOUTH0 + xOffset, Y_MOUTH0 +
yOffset,
                MOUTH_WIDTH, MOUTH_HEIGHT0 + 3 * (i - 2),
                MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE);
        //Add text:
        canvas.drawString ("Tee Hee.",
                X_FACE0 + 50 * i + FACE_DIAMETER, Y_FACE0 +
yOffset);
    }
}
```

Listing 5.1

```
public class Dog
{
        public String name;
        public String breed;
        public int age;

        public void writeOutput()
        {
                System.out.println("Name:  " + name);
                System.out.println("Breed: " + breed);
                System.out.println("Age in calendar years: "
+ age);
                System.out.println("Age in human years: " +
getAgeInHumanYears());
                System.out.println();
        }

        public int getAgeInHumanYears()
        {
                int humanYears = 0;
                if (age <= 2)
                {
                        humanYears = age * 11;
                }
                else
                {
                        humanYears = 22 + ((age-2) * 5);
                }
                return humanYears;
        }
}
```

Listing 5.2

```
public class DogDemo
{
        public static void main(String[] args)
        {
                Dog balto = new Dog();
                balto.name = "Balto";
                balto.age = 8;
                balto.breed = "Siberian Husky";
                balto.writeOutput();

                Dog scooby = new Dog();
                scooby.name = "Scooby";
                scooby.age = 42;
                scooby.breed = "Great Dane";
                System.out.println(scooby.name + " is a " +
scooby.breed + ".");
                System.out.print("He is " + scooby.age + "
years old, or ");
                int humanYears = scooby.getAgeInHumanYears();
                System.out.println(humanYears + " in human
years.");
        }
}
```

Listing 5.3

```
import java.util.Scanner;
public class SpeciesFirstTry
{
    public String name;
    public int population;
    public 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 ();
        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 +
"%");
    }


    public int getPopulationIn10 ()
    {
        int result = 0;
        double populationAmount = population;
        int count = 10;
        while ((count > 0) && (populationAmount > 0))
        {
            populationAmount = populationAmount +
                (growthRate / 100) * populationAmount;
            count - - ;
        }
        if (populationAmount > 0)
```

```
            result = (int) populationAmount;
        return result;
    }
}
```

Listing 5.4

```
public class SpeciesFirstTryDemo
{
    public static void main (String [] args)
    {
        SpeciesFirstTry speciesOfTheMonth = new
SpeciesFirstTry ();
        System.out.println ("Enter data on the Species of
the Month:");
        speciesOfTheMonth.readInput ();
        speciesOfTheMonth.writeOutput ();
        int futurePopulation =
speciesOfTheMonth.getPopulationIn10 ();
        System.out.println ("In ten years the population
will be " +
                futurePopulation);
        //Change the species to show how to change
        //the values of instance variables:
        speciesOfTheMonth.name = "Klingon ox";
        speciesOfTheMonth.population = 10;
        speciesOfTheMonth.growthRate = 15;
        System.out.println ("The new Species of the
Month:");
        speciesOfTheMonth.writeOutput ();
        System.out.println ("In ten years the population
will be " +
                speciesOfTheMonth.getPopulationIn10 ());
    }
}
```

Listing 5.5A

```
/**
This class is used in the program LocalVariablesDemoProgram.
*/
public class BankAccount
```

```
{
    public double amount;
    public double rate;
    public void showNewBalance ()
    {
        double newAmount = amount + (rate / 100.0) * amount;
        System.out.println ("With interest added, the new
amount is $" +
                    newAmount);
    }
}
```

Listing 5.5B

```
/**
A toy program to illustrate how local variables behave.
*/
public class LocalVariablesDemoProgram
{
    public static void main (String [] args)
    {
        BankAccount myAccount = new BankAccount ();
        myAccount.amount = 100.00;
        myAccount.rate = 5;
        double newAmount = 800.00;
        myAccount.showNewBalance ();
        System.out.println ("I wish my new amount were $" +
newAmount);
    }
}
```

Listing 5.6

```
import java.util.Scanner;
public class SpeciesSecondTry
{
    public String name;
    public int population;
    public 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 ();
        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 +
"%");
    }


    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;
    }
}
```

Listing 5.7

```
/**
Demonstrates the use of a parameter
with the method predictPopulation.
*/
public class SpeciesSecondTryDemo
{
    public static void main (String [] args)
    {
        SpeciesSecondTry speciesOfTheMonth = new
SpeciesSecondTry ();
        System.out.println ("Enter data on the Species of
the Month:");
        speciesOfTheMonth.readInput ();
        speciesOfTheMonth.writeOutput ();
        int futurePopulation =
speciesOfTheMonth.predictPopulation (10);
        System.out.println ("In ten years the population
will be " +
                futurePopulation);
        //Change the species to show how to change
        //the values of instance variables:
        speciesOfTheMonth.name = "Klingon ox";
        speciesOfTheMonth.population = 10;
        speciesOfTheMonth.growthRate = 15;
        System.out.println ("The new Species of the
Month:");
        speciesOfTheMonth.writeOutput ();
        System.out.println ("In ten years the population
will be " +
                speciesOfTheMonth.predictPopulation (10));
    }
}
```

Listing 5.8

```
import java.util.Scanner;
public class SpeciesThirdTry
{
    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 ();
        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 +
"%");
    }


    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;
    }
}
```

Listing 5.9

```
/**
Class that represents a rectangle.
*/
public class Rectangle
{
    private int width;
    private int height;
    private int area;
    public void setDimensions (int newWidth, int newHeight)
    {
        width = newWidth;
        height = newHeight;
        area = width * height;
    }


    public int getArea ()
    {
        return area;
    }
}
```

Listing 5.10

```
/**
Another class that represents a rectangle.
*/
public class Rectangle2
{
    private int width;
    private int height;

    public void setDimensions (int newWidth, int newHeight)
    {
        width = newWidth;
        height = newHeight;
    }


    public int getArea ()
    {
        return width * height;
    }
}
```

Listing 5.11

```
import java.util.Scanner;
public class SpeciesFourthTry
{
    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 ();
        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 +
"%");
    }


    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;
    }
}
```

Listing5.12

```
import java.util.Scanner;
/**
Demonstrates the use of the mutator method setSpecies.
*/
public class SpeciesFourthTryDemo
{
    public static void main (String [] args)
    {
        SpeciesFourthTry speciesOfTheMonth =
            new SpeciesFourthTry ();
        System.out.println ("Enter number of years to
project:");
        Scanner keyboard = new Scanner (System.in);
        int numberOfYears = keyboard.nextInt ();
        System.out.println (
                "Enter data on the Species of the Month:");
        speciesOfTheMonth.readInput ();
        speciesOfTheMonth.writeOutput ();
        int futurePopulation =
            speciesOfTheMonth.predictPopulation
(numberOfYears);
        System.out.println ("In " + numberOfYears +
                " years the population will be " +
                futurePopulation);
        //Change the species to show how to change
        //the values of instance variables:
        speciesOfTheMonth.setSpecies ("Klingon ox", 10, 15);
        System.out.println ("The new Species of the
Month:");
        speciesOfTheMonth.writeOutput ();
        futurePopulation =
            speciesOfTheMonth.predictPopulation
(numberOfYears);
        System.out.println ("In " + numberOfYears +
                " years the population will be " +
                futurePopulation);
    }
}
```

Listing 5.13

```
import java.util.Scanner;
/**
Class for the purchase of one kind of item, such as 3
oranges.
Prices are set supermarket style, such as 5 for $1.25.
*/
public class Purchase
{
    private String name;
    private int groupCount; //Part of a price, like the 2 in
2 for $1.99.
    private double groupPrice; //Part of a price, like the
$1.99
    // in 2 for $1.99.
    private int numberBought; //Number of items bought.

    public void setName (String newName)
    {
        name = newName;
    }


    /**
    Sets price to count pieces for $costForCount.
    For example, 2 for $1.99.
    */
    public void setPrice (int count, double costForCount)
    {
        if ((count <= 0) || (costForCount <= 0))
        {
            System.out.println ("Error: Bad parameter in
setPrice.");
            System.exit (0);
        }
        else
        {
            groupCount = count;
            groupPrice = costForCount;
        }
    }


    public void setNumberBought (int number)
    {
        if (number <= 0)
        {
            System.out.println ("Error: Bad parameter in
setNumberBought.");
            System.exit (0);
        }
        else
            numberBought = number;
    }


    /**
    Reads from keyboard the price and number of a purchase.
    */
    public void readInput ()
    {
        Scanner keyboard = new Scanner (System.in);
        System.out.println ("Enter name of item you are
purchasing:");
        name = keyboard.nextLine ();
        System.out.println ("Enter price of item as two
numbers.");
        System.out.println ("For example, 3 for $2.99 is
entered as");
        System.out.println ("3 2.99");
        System.out.println ("Enter price of item as two
numbers, now:");
        groupCount = keyboard.nextInt ();
        groupPrice = keyboard.nextDouble ();
        while ((groupCount <= 0) || (groupPrice <= 0))
        { //Try again:
            System.out.println (
                    "Both numbers must be positive. Try
again.");
            System.out.println ("Enter price of item as two
numbers.");
            System.out.println ("For example, 3 for $2.99 is
entered as");
            System.out.println ("3 2.99");
            System.out.println (
                    "Enter price of item as two numbers,
now:");
            groupCount = keyboard.nextInt ();
            groupPrice = keyboard.nextDouble ();
        }
        System.out.println ("Enter number of items
purchased:");
        numberBought = keyboard.nextInt ();
        while (numberBought <= 0)
        { //Try again:
            System.out.println ("Number must be positive.
Try again.");
            System.out.println ("Enter number of items
purchased:");
            numberBought = keyboard.nextInt ();
        }
    }


    /**
    Displays price and number being purchased.
    */
    public void writeOutput ()
    {
        System.out.println (numberBought + " " + name);
        System.out.println ("at " + groupCount +
                " for $" + groupPrice);
    }


    public String getName ()
    {
        return name;
    }


    public double getTotalCost ()
    {
        return (groupPrice / groupCount) * numberBought;
    }


    public double getUnitCost ()
    {
        return groupPrice / groupCount;
    }


    public int getNumberBought ()
    {
        return numberBought;
    }
}
```

Listing 5.14

```
public class PurchaseDemo
{
    public static void main (String [] args)
    {
        Purchase oneSale = new Purchase ();
        oneSale.readInput ();
        oneSale.writeOutput ();
        System.out.println ("Cost each $" +
oneSale.getUnitCost ());
        System.out.println ("Total cost $" +
oneSale.getTotalCost ());
    }
}
```

Listing5.15

```
import java.util.Scanner;
public class Oracle
{
    private String oldAnswer = "The answer is in your
heart.";
    private String newAnswer;
    private String question;

    public void chat ()
    {
        System.out.print ("I am the oracle. ");
        System.out.println ("I will answer any one-line
question.");
        Scanner keyboard = new Scanner (System.in);
        String response;
        do
        {
            answer ();
            System.out.println ("Do you wish to ask another
question?");
            response = keyboard.next ();
        }
        while (response.equalsIgnoreCase ("yes"));
        System.out.println ("The oracle will now rest.");
    }


    private void answer ()
    {
        System.out.println ("What is your question?");
        Scanner keyboard = new Scanner (System.in);
        question = keyboard.nextLine ();
        seekAdvice ();
        System.out.println ("You asked the question:");
        System.out.println ("  " + question);
        System.out.println ("Now, here is my answer:");
        System.out.println ("  " + oldAnswer);
        update ();
    }


    private void seekAdvice ()
    {
        System.out.println ("Hmm, I need some help on
that.");
        System.out.println ("Please give me one line of
advice.");
        Scanner keyboard = new Scanner (System.in);
        newAnswer = keyboard.nextLine ();
        System.out.println ("Thank you. That helped a
lot.");
    }
    private void update ()
    {
        oldAnswer = newAnswer;
    }
}
```

Listing 5.16

```
public class OracleDemo
{
    public static void main (String [] args)
    {
        Oracle delphi = new Oracle ();
        delphi.chat ();
    }
}
```

Listing 5.17

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

    /*The definition of the methods readInput, writeOutput,
and predictPopulation
        go here.They are the same as in Listing 5.3 and
Listing 5.6 . >
        < The definition of the methods setSpecies, getName,
getPopulation,
        and getGrowthRate go here.They are the same as in
Listing 5.11 . >  */

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

Listing 5.18

```
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);
        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 5.19

```
import java.util.Scanner;
/**
Class for data on endangered species.
*/
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);
    }
}
```

Listing 5.20

```
public class SpeciesTest
{
        public static void main(String[] args)
        {
                Species testSpecies = new Species();

                // Test the setSpecies method
                testSpecies.setSpecies("Tribbles", 100, 50);
                if (testSpecies.getName().equals("Tribbles")
&&
                    (testSpecies.getPopulation() == 100) &&
                    (testSpecies.getGrowthRate() >= 49.99) &&
                    (testSpecies.getGrowthRate() <= 50.01))
                {
                        System.out.println("Pass: setSpecies
test.");
                }
                else
                {
                        System.out.println("FAIL: setSpecies
test.");
                }


                // Test the predictPopulation method
                if ((testSpecies.predictPopulation(-1) ==
100) &&
                    (testSpecies.predictPopulation(1) == 150)
&&
                    (testSpecies.predictPopulation(5) ==
759))

                {
                        System.out.println("Pass:
predictPopulation test.");
                }
                else
                {
                        System.out.println("FAIL:
predictPopulation test.");
                }
        }
}
```

Listing 5.21

```
import java.util.Scanner;
/**
This version of the class Species is only a toy example
designed
```

```
to demonstrate the difference between parameters of a class
type
and parameters of a primitive type.
*/
public class DemoSpecies
{
    private String name;
    private int population;
    private double growthRate;

    /**
    Tries to set intVariable equal to the population of this
object.
    But arguments of a primitive type cannot be changed.
    */
    public void tryToChange (int intVariable)
    {
        intVariable = this.population;
    }


    /**
    Tries to make otherObject reference this object.
    But arguments of a class type cannot be replaced.
    */
    public void tryToReplace (DemoSpecies otherObject)
    {
        otherObject = this;
    }


    /**
    Changes the data in otherObject to the data in this
object,
    which is unchanged.
    */
    public void change (DemoSpecies otherObject)
    {
        otherObject.name = this.name;
        otherObject.population = this.population;
        otherObject.growthRate = this.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);
    }

}
```

Listing 5.22

```
public class ParametersDemo
{
    public static void main (String [] args)
    {
        DemoSpecies s1 = new DemoSpecies (),
            s2 = new DemoSpecies ();
        s1.setSpecies ("Klingon ox", 10, 15);
        int aPopulation = 42;
        System.out.println ("aPopulation BEFORE calling
tryToChange: "
                + aPopulation);
        s1.tryToChange (aPopulation);
        System.out.println ("aPopulation AFTER calling
tryToChange: "
                + aPopulation);
        s2.setSpecies ("Ferengie Fur Ball", 90, 56);
        System.out.println ("s2 BEFORE calling tryToReplace:
");
        s2.writeOutput ();
        s1.tryToReplace (s2);
        System.out.println ("s2 AFTER calling tryToReplace: 
");
        s2.writeOutput ();
        s1.change (s2);
        System.out.println ("s2 AFTER calling change: ");
        s2.writeOutput ();
    }
}
```

Listing 5.23

```
import javax.swing.JApplet;
import java.awt.Graphics;
import java.awt.Color;
public class MultipleFaces extends JApplet
{
    public static final int FACE_DIAMETER = 50;
    public static final int X_FACE0 = 10;
    public static final int Y_FACE0 = 5;
    public static final int EYE_WIDTH = 5;
    public static final int EYE_HEIGHT = 10;
    public static final int X_RIGHT_EYE0 = 20;
    public static final int Y_RIGHT_EYE0 = 15;
    public static final int X_LEFT_EYE0 = 45;
    public static final int Y_LEFT_EYE0 = Y_RIGHT_EYE0;
    public static final int NOSE_DIAMETER = 5;
    public static final int X_NOSE0 = 32;
    public static final int Y_NOSE0 = 25;
    public static final int MOUTH_WIDTH = 30;
    public static final int MOUTH_HEIGHT0 = 0;
    public static final int X_MOUTH0 = 20;
    public static final int Y_MOUTH0 = 35;
    public static final int MOUTH_START_ANGLE = 180;
    public static final int MOUTH_EXTENT_ANGLE = 180;

    /**
    g is the drawing area. pos indicates the position of the 
face.
    As pos increases, the face is drawn lower and further to 
the right.
    */
    private void drawFaceSansMouth (Graphics g, int pos)
    {
        g.setColor (Color.BLACK);
        g.drawOval (X_FACE0 + 50 * pos, Y_FACE0 + 30 * pos,
                FACE_DIAMETER, FACE_DIAMETER);
        //Draw eyes:
        g.setColor (Color.BLUE);
        g.fillOval (X_RIGHT_EYE0 + 50 * pos, Y_RIGHT_EYE0 +
30 * pos,
                EYE_WIDTH, EYE_HEIGHT);
        g.fillOval (X_LEFT_EYE0 + 50 * pos, Y_LEFT_EYE0 + 30
* pos,
                EYE_WIDTH, EYE_HEIGHT);
        //Draw nose:
        g.setColor (Color.BLACK);
        g.fillOval (X_NOSE0 + 50 * pos, Y_NOSE0 + 30 * pos,
                NOSE_DIAMETER, NOSE_DIAMETER);
    }


    public void paint (Graphics canvas)
    {
        super.paint(canvas);
        int i;
        for (i = 0 ; i < 5 ; i++)
        { //Draw one face:
            if (i % 2 == 0) //If i is even,
            { //make face yellow
                canvas.setColor (Color.YELLOW);
                canvas.fillOval (X_FACE0 + 50 * i, Y_FACE0 +
30 * i,
                        FACE_DIAMETER, FACE_DIAMETER);
            }
            drawFaceSansMouth (canvas, i);
            //Draw mouth:
            canvas.setColor (Color.RED);
            canvas.drawArc (X_MOUTH0 + 50 * i, Y_MOUTH0 + 30
* i,
                    MOUTH_WIDTH, MOUTH_HEIGHT0 + 3 * i,
                    MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE);
        }
        //i == 5
        //Draw kissing face:
        drawFaceSansMouth (canvas, i);
        //Draw mouth in shape of a kiss:
        canvas.setColor (Color.RED);
        canvas.fillOval (X_MOUTH0 + 50 * i + 10, Y_MOUTH0 +
30 * i,
                MOUTH_WIDTH - 20, MOUTH_WIDTH - 20);
        //Add text:
        canvas.setColor (Color.BLACK);
        canvas.drawString ("Kiss, Kiss.",
                X_FACE0 + 50 * i + FACE_DIAMETER, Y_FACE0 +
30 * i);
        //Draw blushing face:
        i++;
        //Draw face circle:
        canvas.setColor (Color.PINK);
        canvas.fillOval (X_FACE0 + 50 * i, Y_FACE0 + 30 * i,
                FACE_DIAMETER, FACE_DIAMETER);
        drawFaceSansMouth (canvas, i);
        //Draw mouth:
        canvas.setColor (Color.RED);
        canvas.drawArc (X_MOUTH0 + 50 * i, Y_MOUTH0 + 30 *
i,
                MOUTH_WIDTH, MOUTH_HEIGHT0 + 3 * (i - 2),
                MOUTH_START_ANGLE, MOUTH_EXTENT_ANGLE);
        //Add text:
        canvas.setColor (Color.BLACK);
        canvas.drawString ("Tee Hee.",
                X_FACE0 + 50 * i + FACE_DIAMETER, Y_FACE0 +
30 * i);
    }
}
```

Listing 5.24

```
import javax.swing.JFrame;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.awt.image.RescaleOp;
import javax.imageio.ImageIO;
import java.io.File;
import java.io.IOException;

public class Graphics2DExample extends JFrame
{
 public void paint(Graphics canvas)
 {
   super.paint(canvas);
   try
   {
     // Load image from default location on disk. This is
inefficient
     // because the image will be re-loaded everytime the
JFrame is
     // displayed.  A better technique would be to load the
image
     // once in the constructor (discussed in a later
chapter).
     BufferedImage img = ImageIO.read(new File("java.jpg"));
     // Draw the image at coordinate 50,50
     canvas.drawImage(img, 50, 50, null);

     // Copy the image to another buffer with a
     // color model (ARGB) to support alpha blending
     // that allows translucency
     int w = img.getWidth(null);
     int h = img.getHeight(null);
     BufferedImage img2 = new
       BufferedImage(w, h, BufferedImage.TYPE_INT_ARGB);
     Graphics g = img2.getGraphics();
     g.drawImage(img, 0, 0, null);


     // Create a rescale filter operation that
     // makes the image 30% opaque
     float[] scales = { 1f, 1f, 1f, 0.3f };
     float[] offsets = new float[4];
     RescaleOp rop = new RescaleOp(scales, offsets, null);
     // Draw the image, applying the filter
     Graphics2D g2 = (Graphics2D) canvas;
     g2.drawImage(img2, rop, 150, 50);
   }
   catch (IOException e)
   {
     System.out.println("Error reading the image.");
   }
 }
 public Graphics2DExample()
 {
   setSize(275,175);
   setDefaultCloseOperation(EXIT_ON_CLOSE);
 }
 public static void main(String[] args)
 {
   Graphics2DExample guiWindow = new Graphics2DExample();
   guiWindow.setVisible(true);
 }
}
```

Listing 5.25

```
import javax.swing.JApplet;
import javax.swing.JLabel;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
/**
An applet that uses a label to display text.
*/
public class LabelDemo extends JApplet
{
    public void init ()
    {
        Container contentPane = getContentPane ();
        contentPane.setBackground (Color.WHITE);
        //Create labels:
        JLabel label1 = new JLabel ("Hello ");
        JLabel label2 = new JLabel ("out there!");
        //Add labels:
        contentPane.setLayout (new FlowLayout ());
        contentPane.add (label1);
        contentPane.add (label2);
    }
}
```