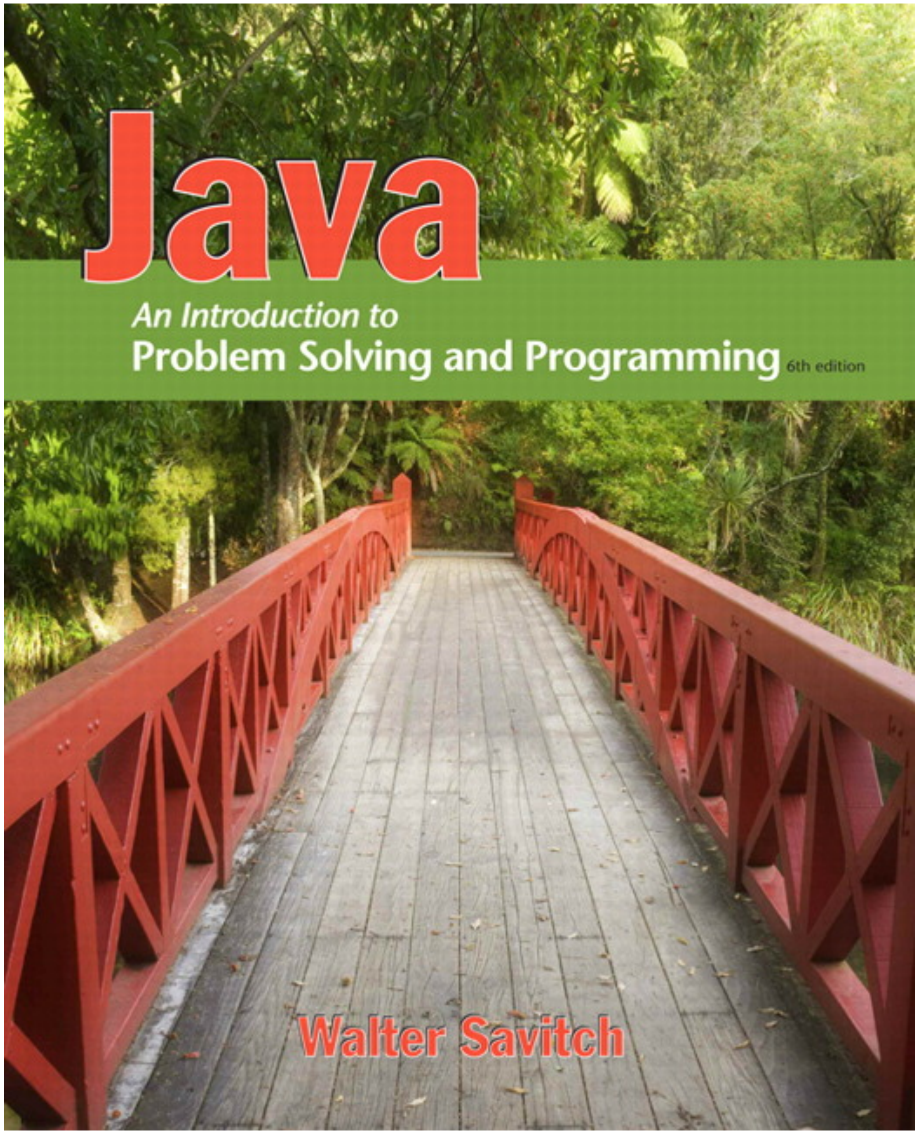
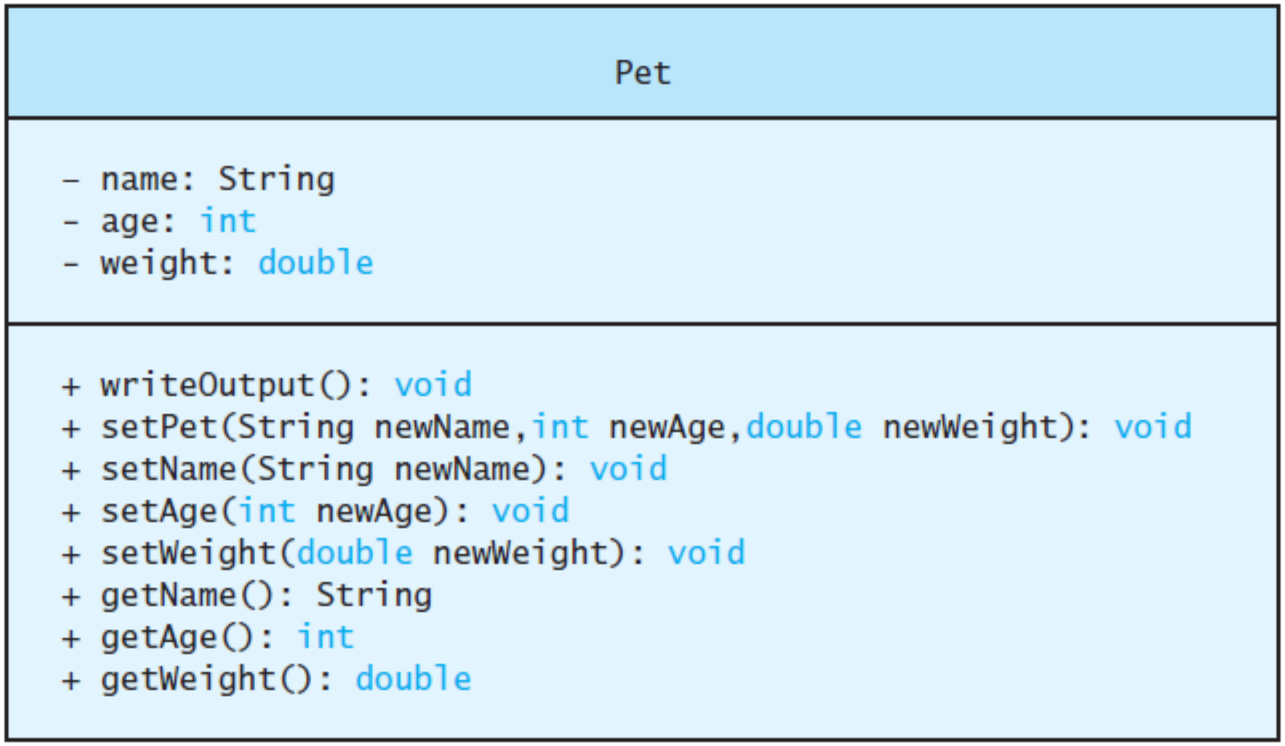


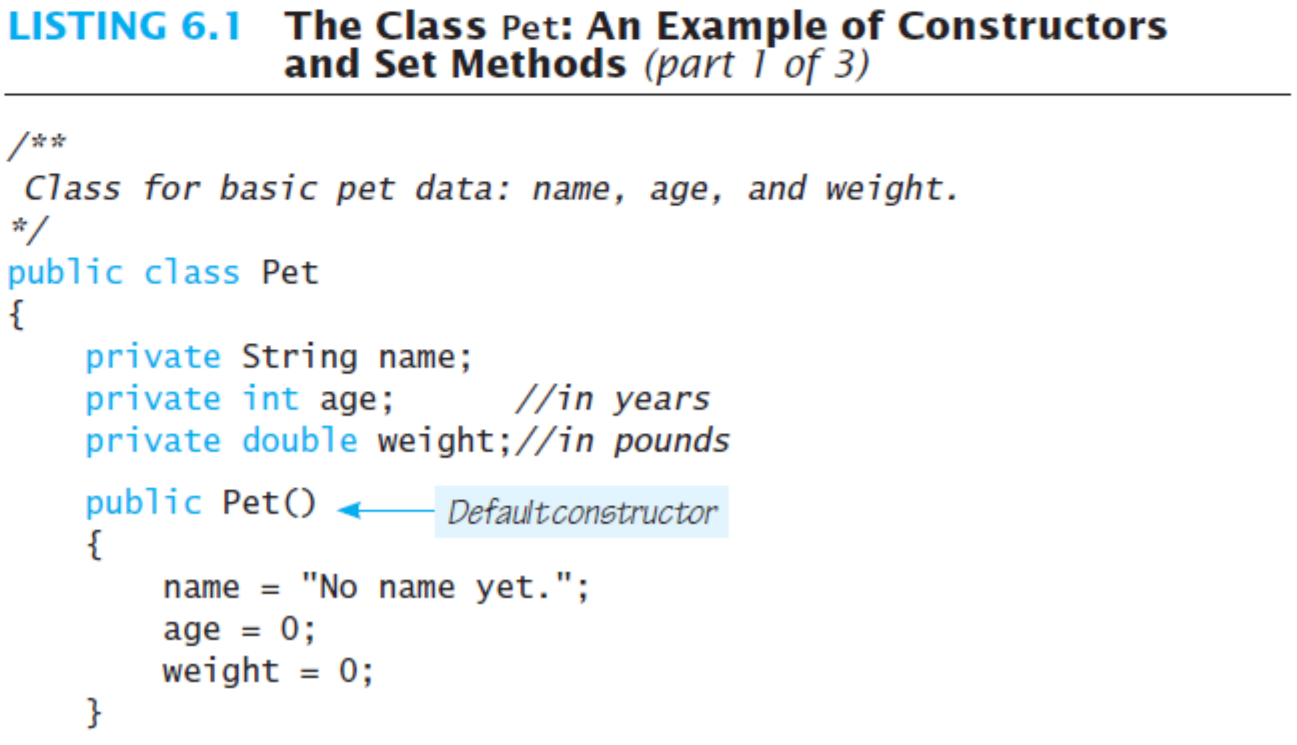
More About Objects and Methods 6

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709

FIGURE 6.1 Class Diagram for a Class Pet



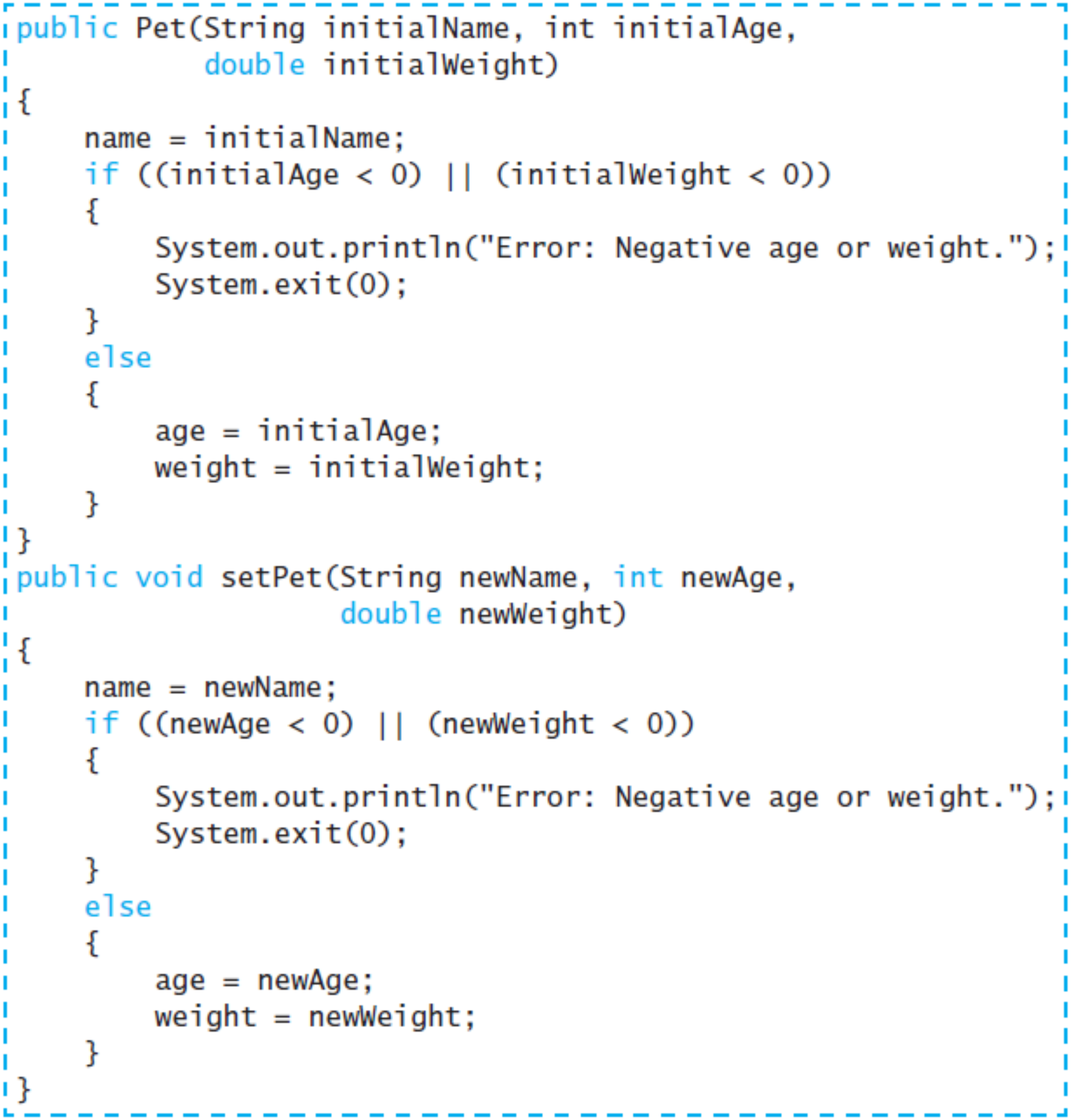
```
Pet

- name: String
- age: int
- weight: double

+ writeOutput(): void
+ setPet(String newName,int newAge,double newWeight): void
+ setName(String newName): void
+ setAge(int newAge): void
+ setWeight(double newWeight): void
+ getName(): String
+ getAge(): int
+ getWeight(): double
```



LISTING 6.1 The Class Pet: An Example of Constructors and Set Methods *(part 1 of 3)*

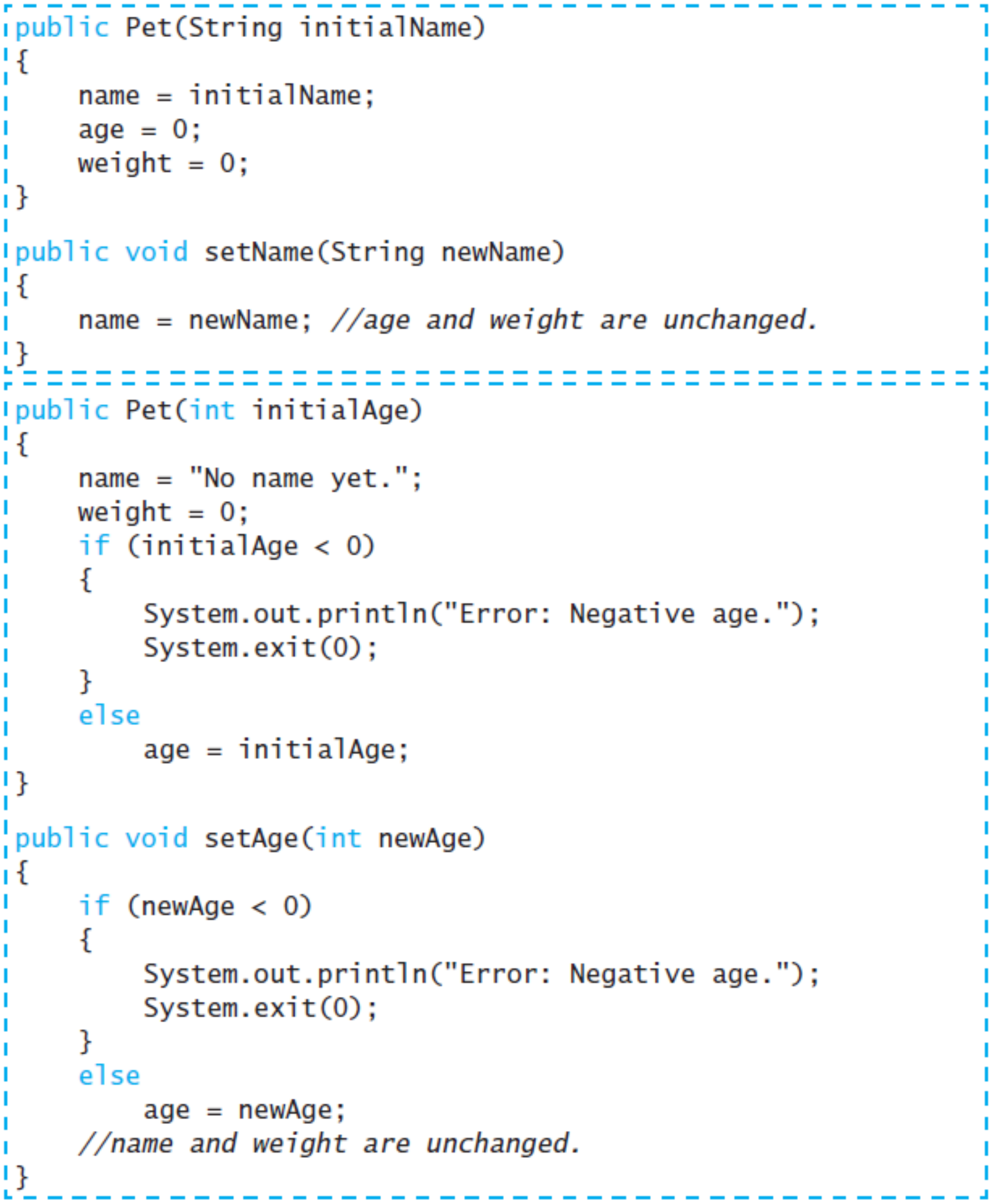
```
/**
 Class for basic pet data: name, age, and weight.
*/
public class Pet
{
    private String name;
    private int age;      //in years
    private double weight;//in pounds

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

Default constructor



```
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;
    }
}
```

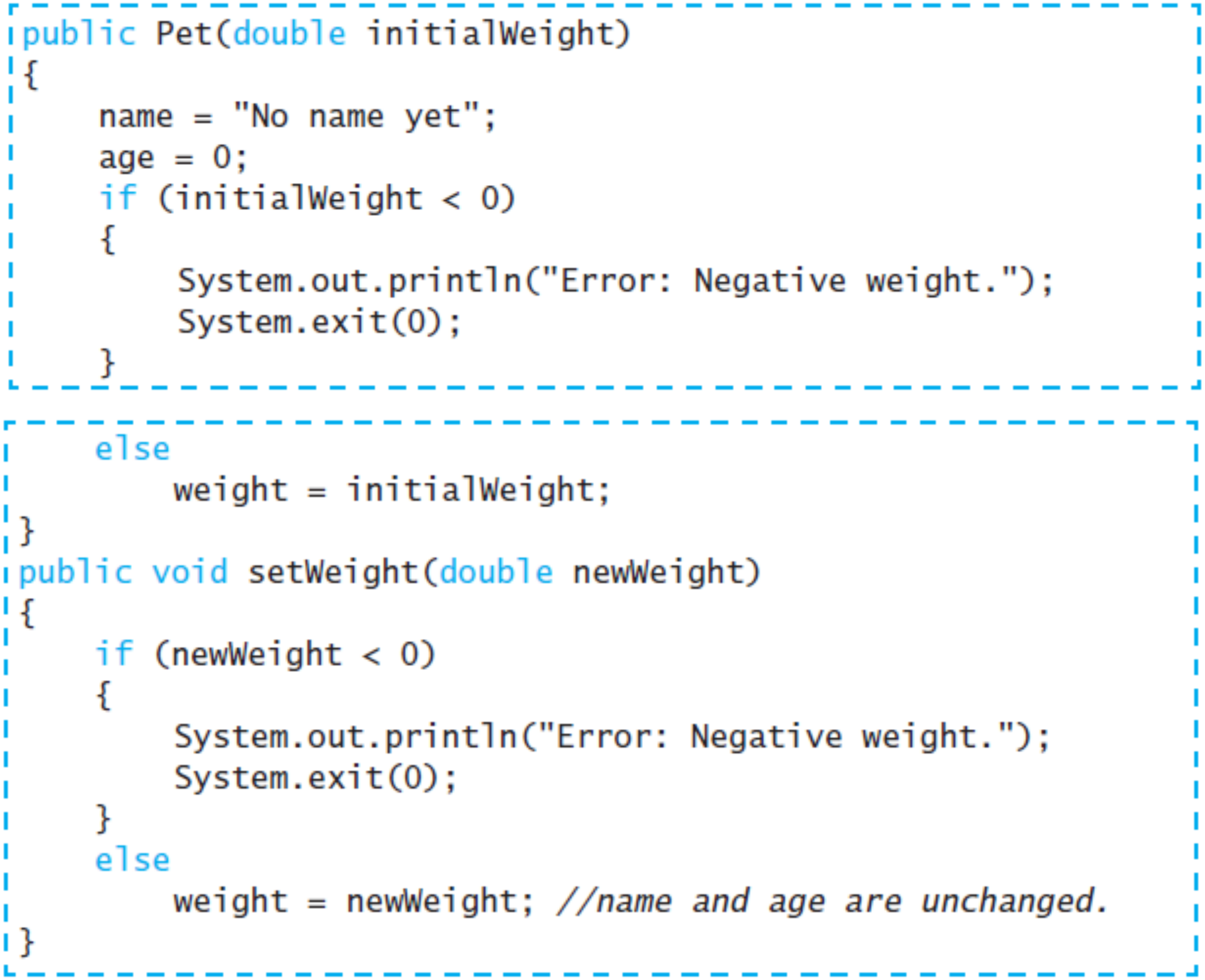
JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch



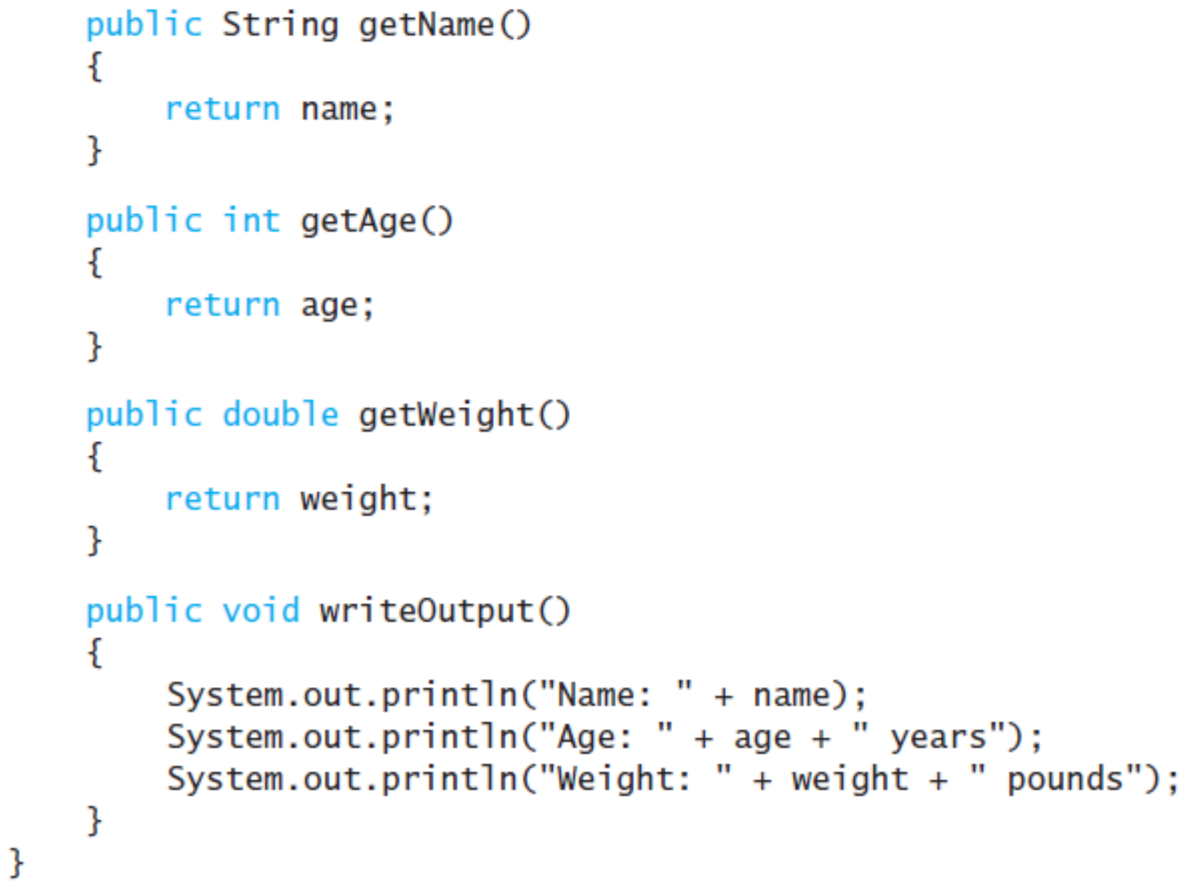
```
public Pet(String initialName)
{
    name = initialName;
    age = 0;
    weight = 0;
}

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.
}
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

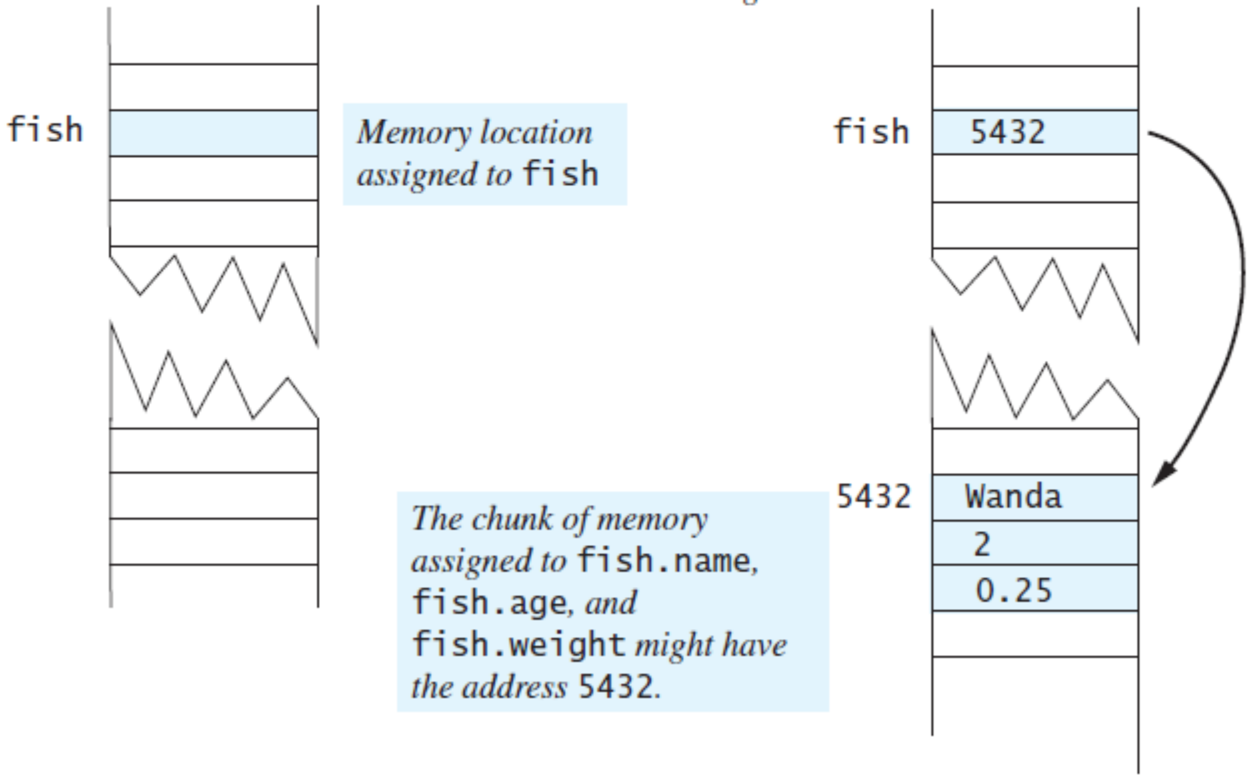


```
public Pet(double initialWeight)
{
    name = "No name yet";
    age = 0;
    if (initialWeight < 0)
    {
        System.out.println("Error: Negative weight.");
        System.exit(0);
    }
    else
        weight = initialWeight;
}
public void setWeight(double newWeight)
{
    if (newWeight < 0)
    {
        System.out.println("Error: Negative weight.");
        System.exit(0);
    }
    else
        weight = newWeight; //name and age are unchanged.
}
```



```java
    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");
    }
}
```

FIGURE 6.2 A Constructor Returning a Reference



`Pet fish;`

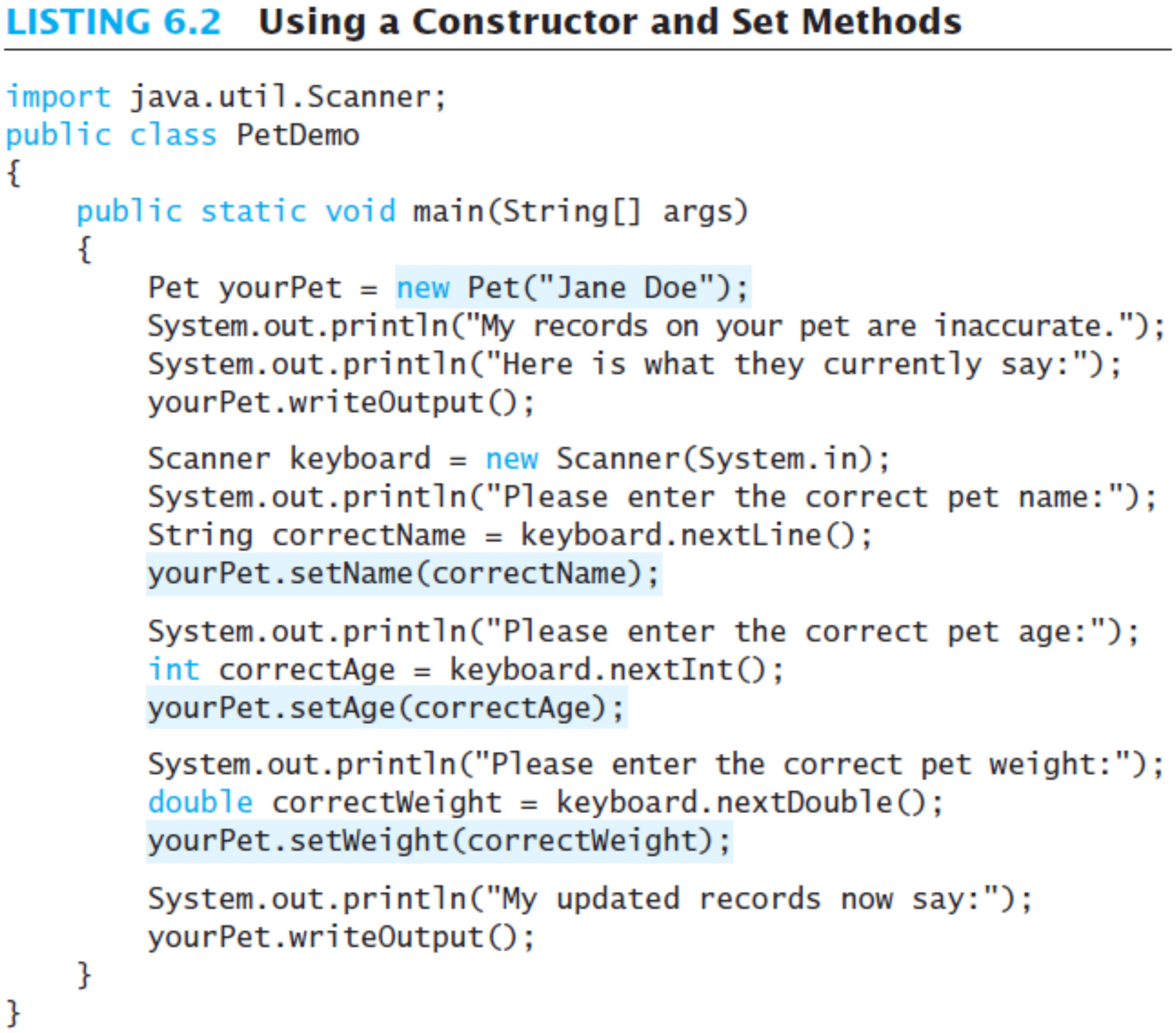
Assigns a memory location to `fish`

Memory location assigned to `fish`

`fish = new Pet();`

Assigns a chunk of memory for an object of the class `Pet`*—that is, memory for a name, an age, and a weight—and places the address of this memory chunk in the memory location assigned to* `fish`

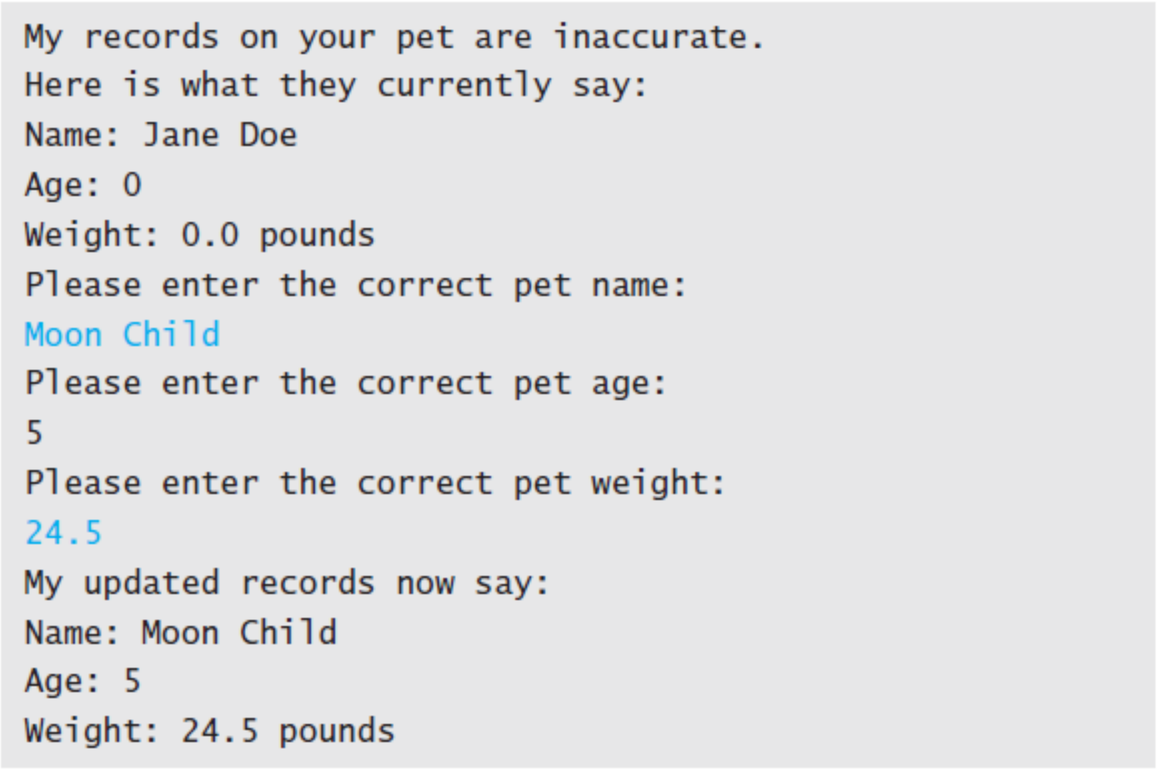
The chunk of memory assigned to `fish.name`*,* `fish.age`*, and* `fish.weight` *might have the address* `5432`*.*



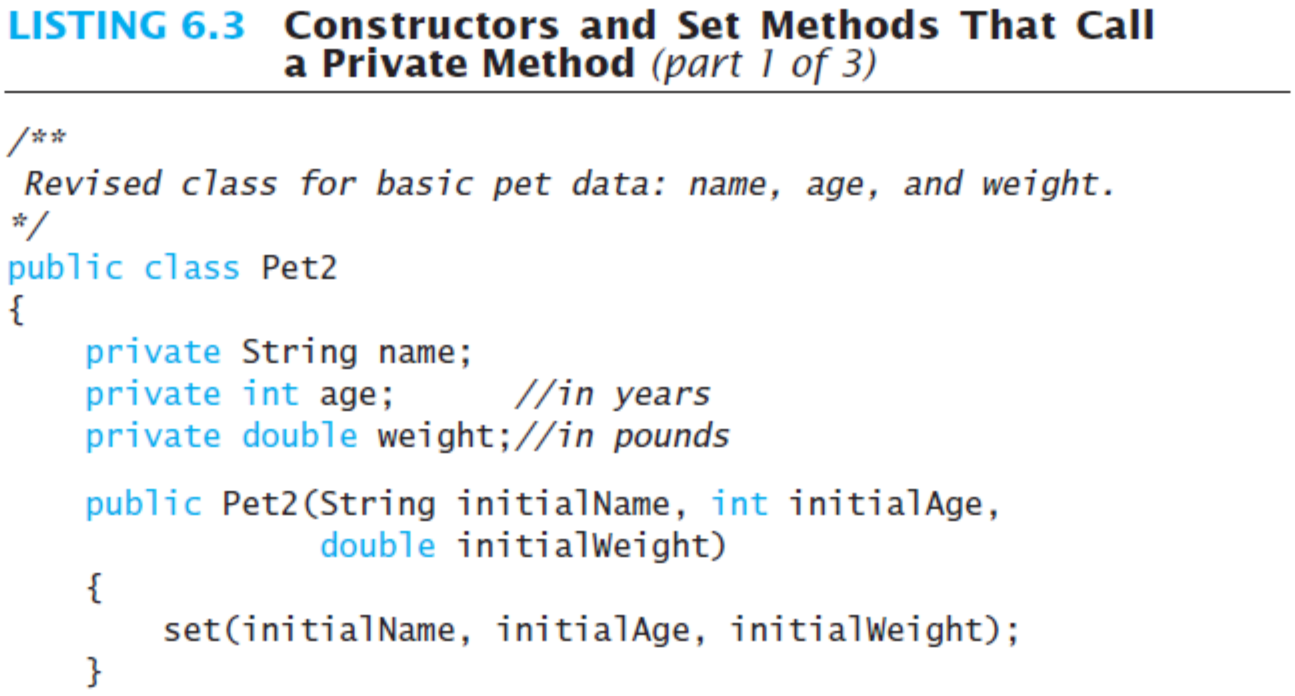
LISTING 6.2 Using a Constructor and Set Methods

```
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();
    }
}
```

Sample Screen Output

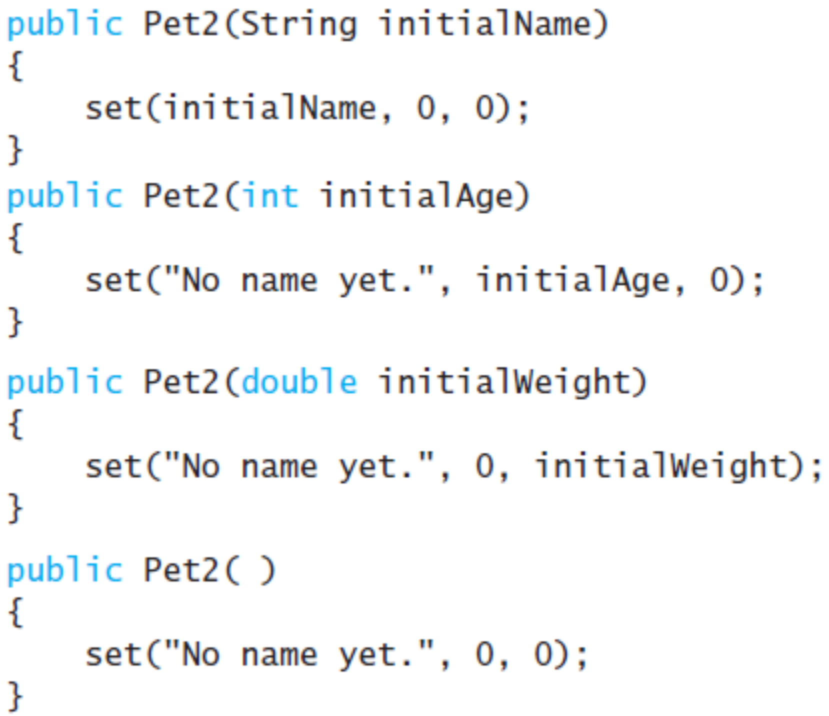


```
My records on your pet are inaccurate.
Here is what they currently say:
Name: Jane Doe
Age: 0
Weight: 0.0 pounds
Please enter the correct pet name:
Moon Child
Please enter the correct pet age:
5
Please enter the correct pet weight:
24.5
My updated records now say:
Name: Moon Child
Age: 5
Weight: 24.5 pounds
```

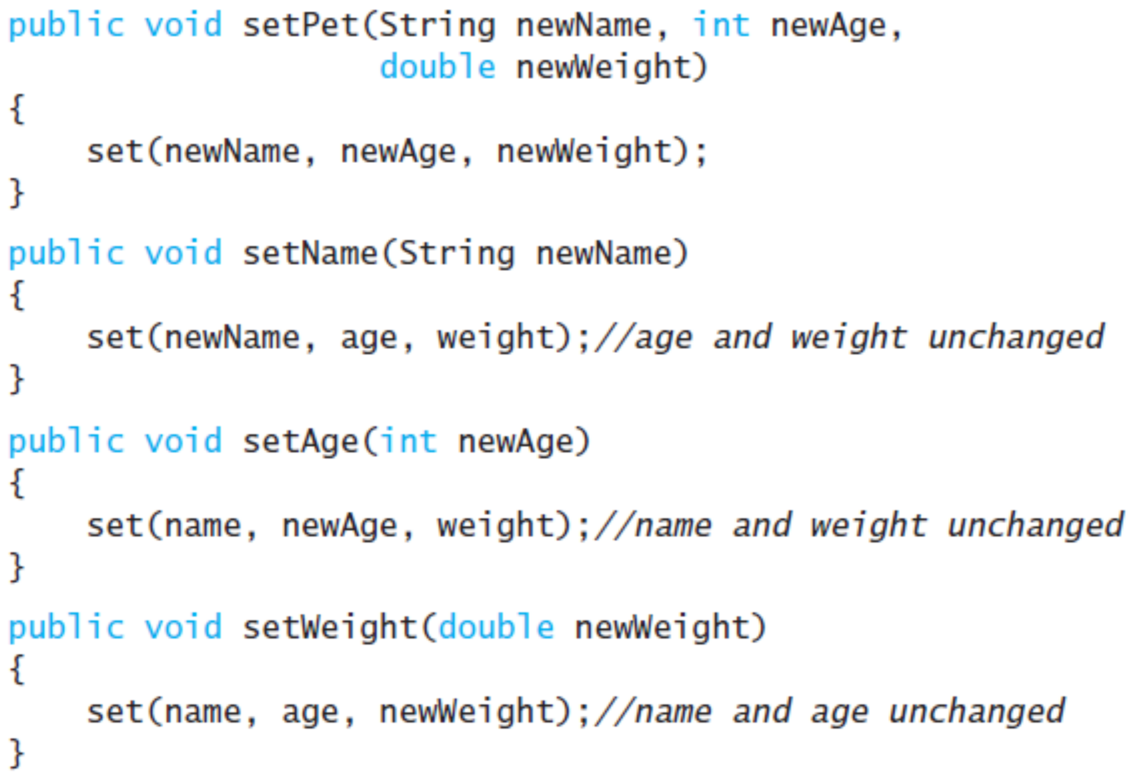


LISTING 6.3 Constructors and Set Methods That Call a Private Method *(part 1 of 3)*

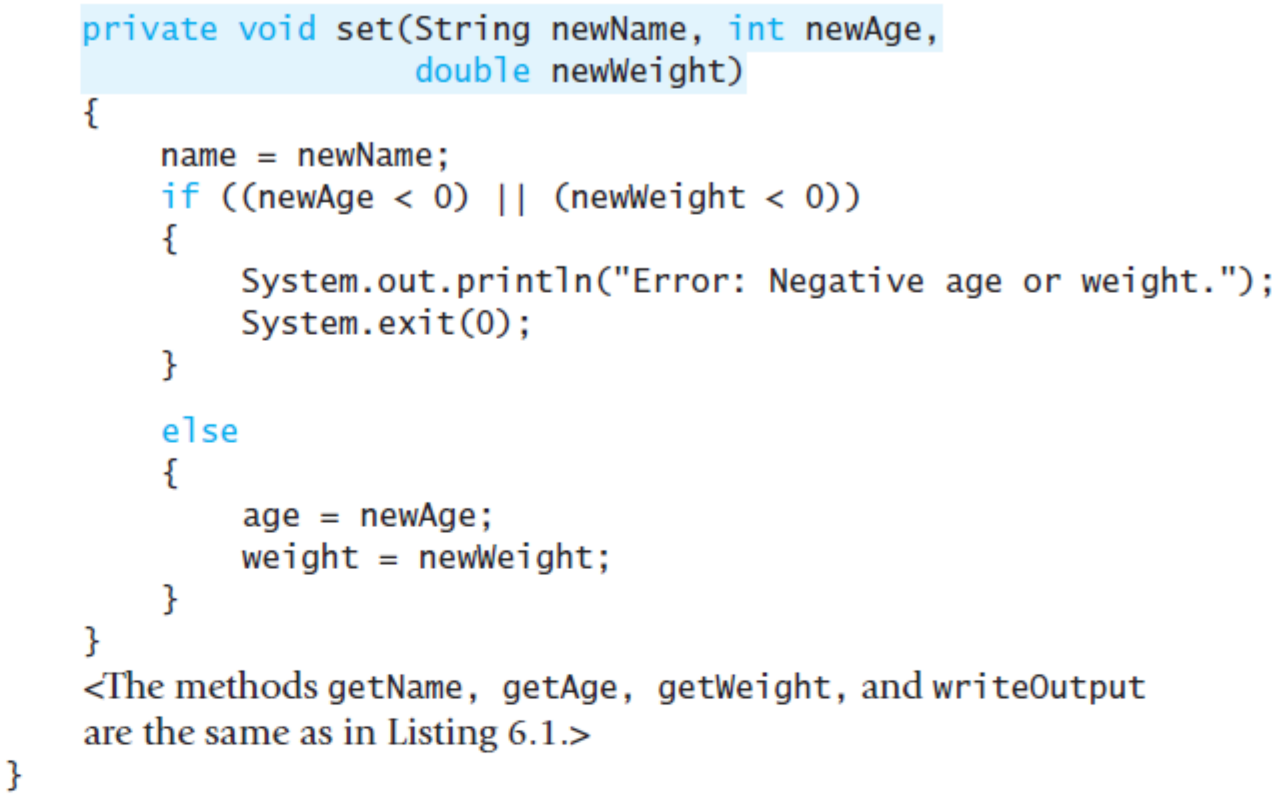
```java
/**
 Revised class for basic pet data: name, age, and weight.
*/
public class Pet2
{
    private String name;
    private int age;      //in years
    private double weight;//in pounds

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



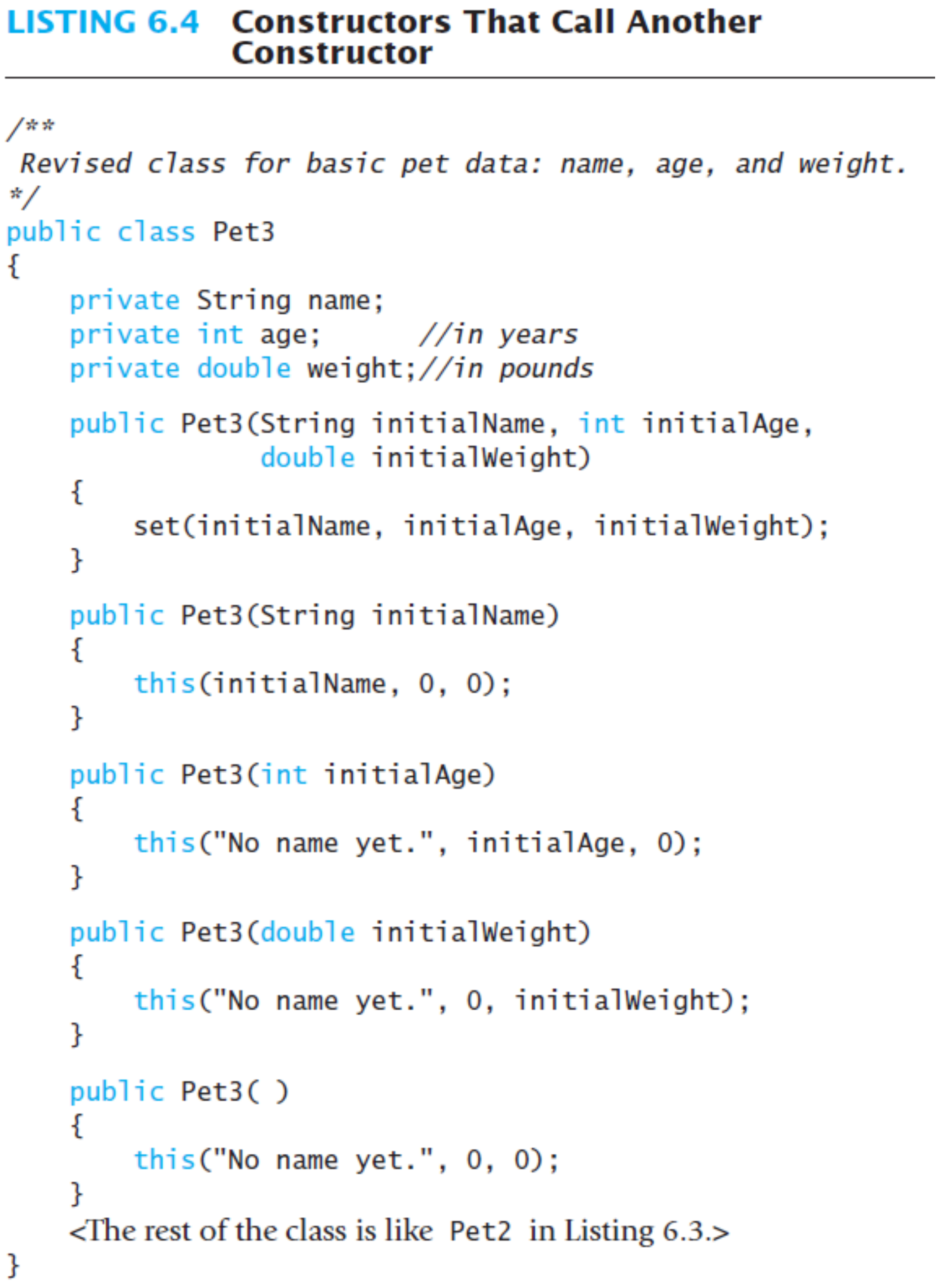
```
public Pet2(String initialName)
{
    set(initialName, 0, 0);
}
public Pet2(int initialAge)
{
    set("No name yet.", initialAge, 0);
}
public Pet2(double initialWeight)
{
    set("No name yet.", 0, initialWeight);
}
public Pet2( )
{
    set("No name yet.", 0, 0);
}
```



```java
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
}
```

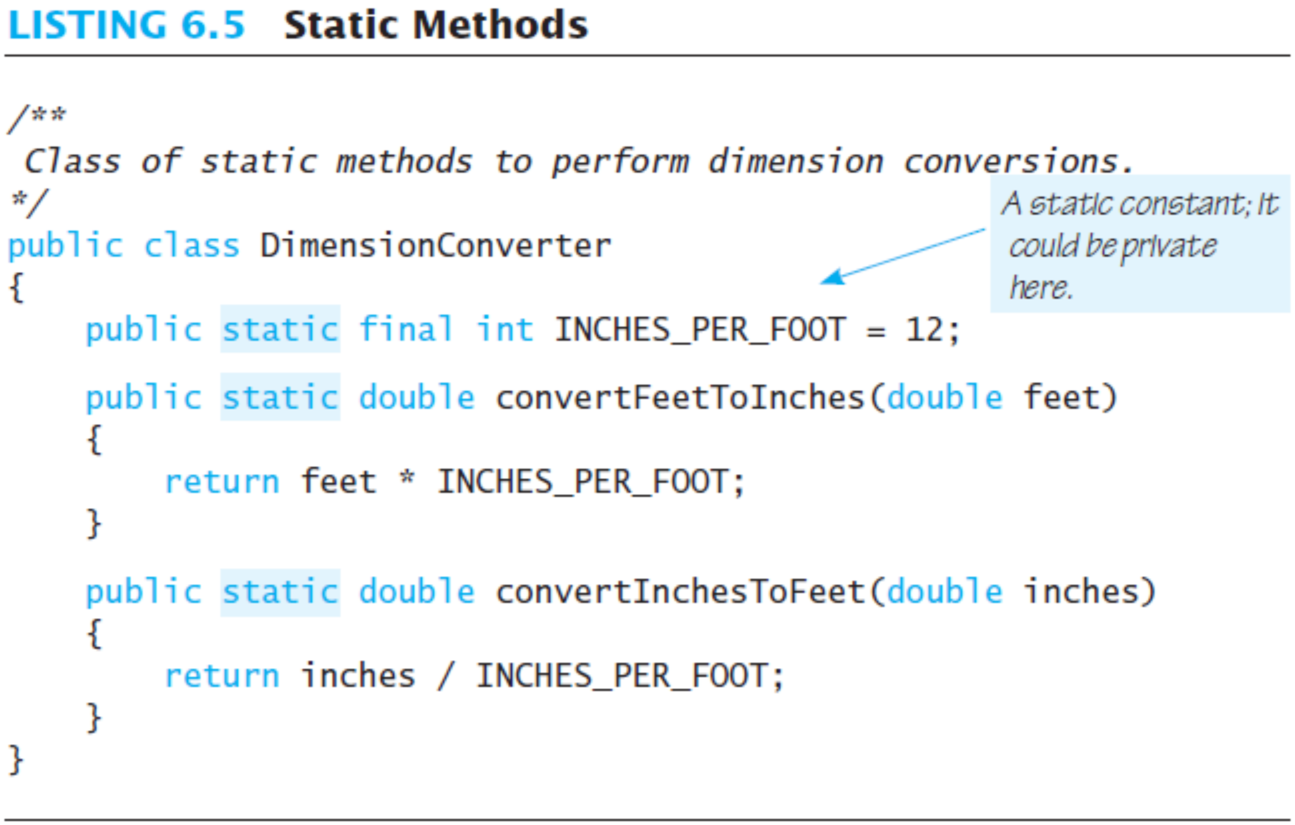


```
    private void set(String newName, int newAge,
                     double newWeight)
    {
        name = newName;
        if ((newAge < 0) || (newWeight < 0))
        {
            System.out.println("Error: Negative age or weight.");
            System.exit(0);
        }

        else
        {
            age = newAge;
            weight = newWeight;
        }
    }
    <The methods getName, getAge, getWeight, and writeOutput
    are the same as in Listing 6.1.>
}
```



LISTING 6.4 Constructors That Call Another Constructor

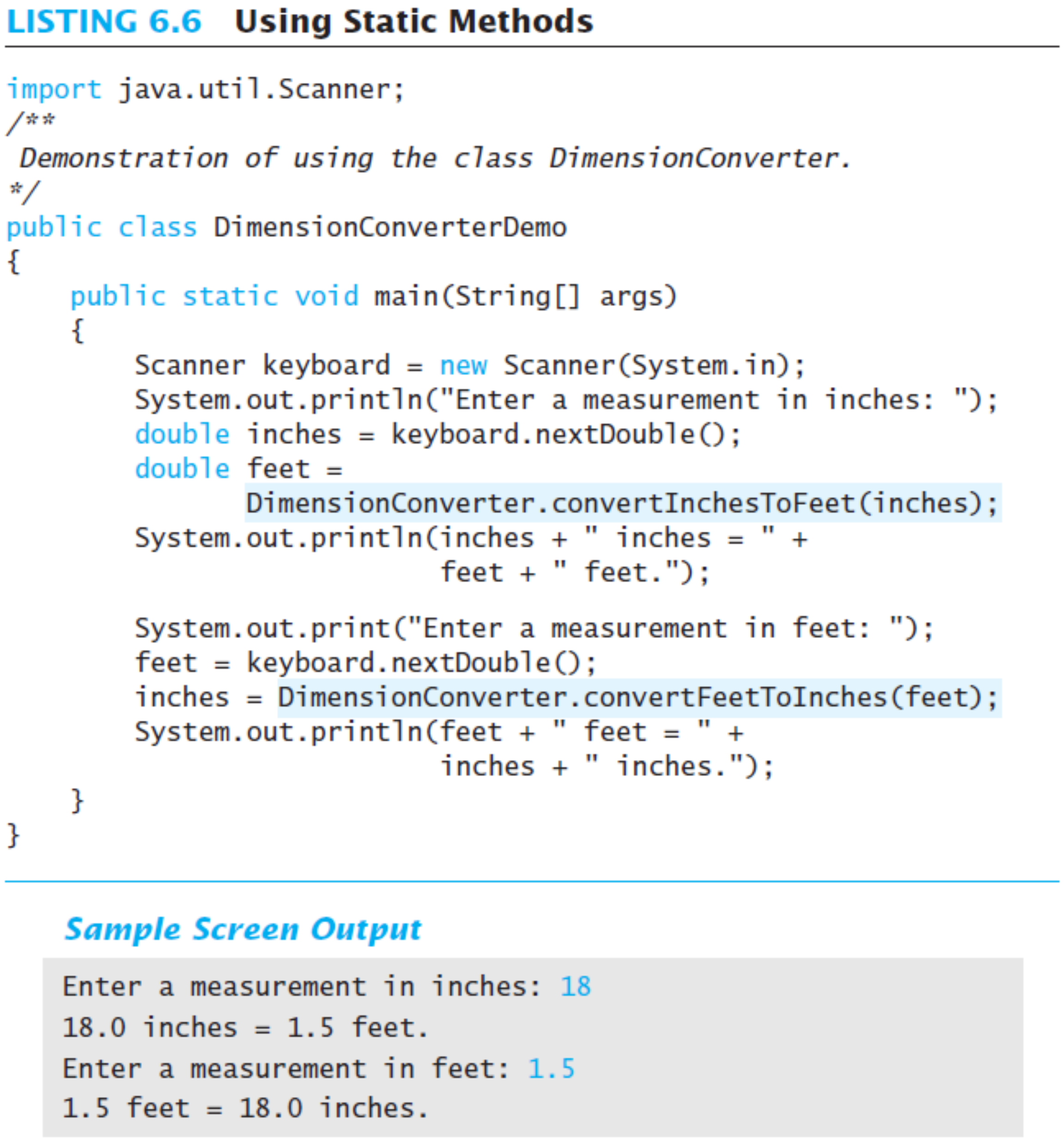
```
/**
 Revised class for basic pet data: name, age, and weight.
*/
public class Pet3
{
    private String name;
    private int age;      //in years
    private double weight;//in pounds

    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 Static Methods

```
/**
 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;
    }
}
```

A static constant; it could be private here.



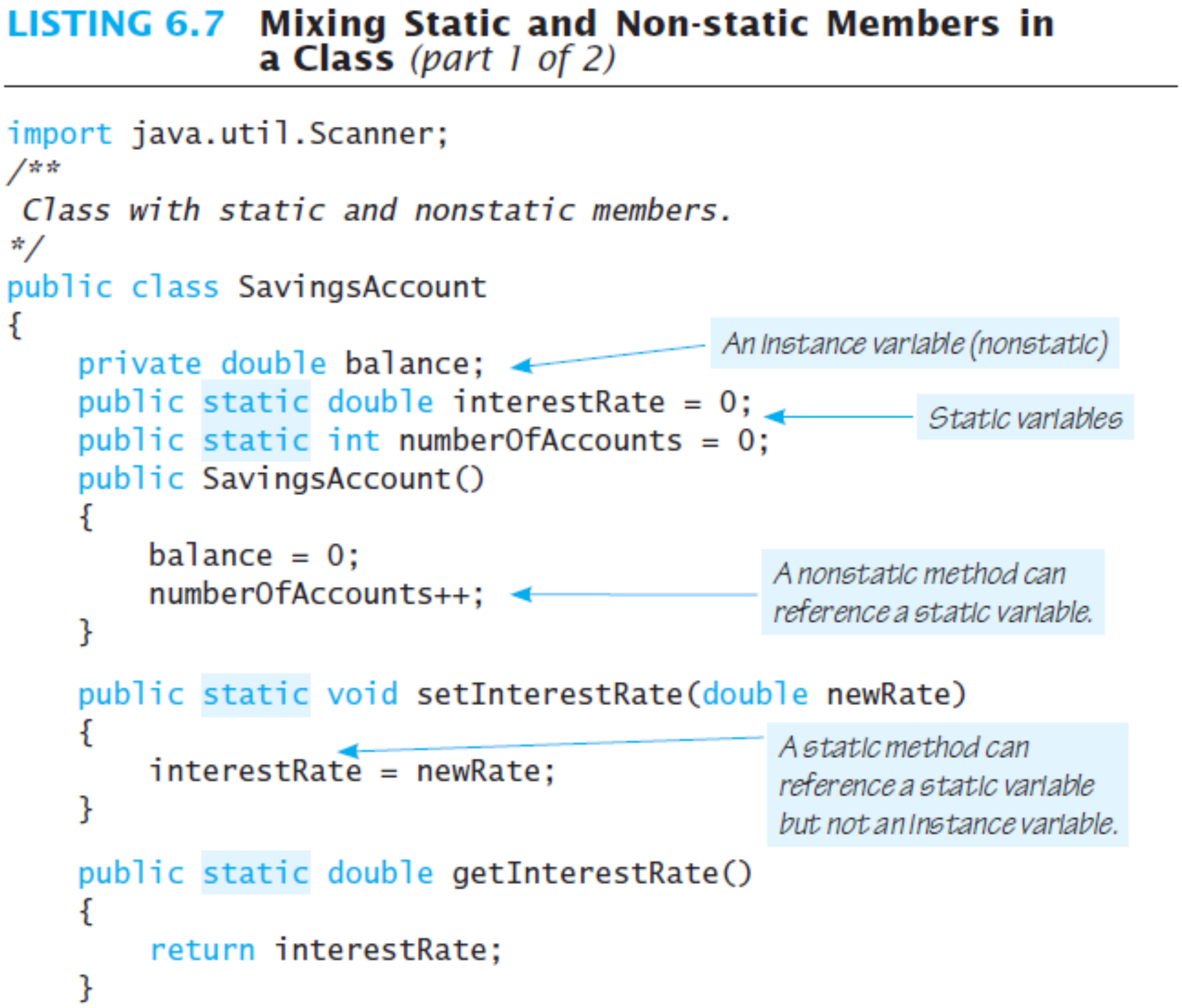
LISTING 6.6 Using Static Methods

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

Sample Screen Output

```
Enter a measurement in inches: 18
18.0 inches = 1.5 feet.
Enter a measurement in feet: 1.5
1.5 feet = 18.0 inches.
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



LISTING 6.7 Mixing Static and Non-static Members in a Class *(part 1 of 2)*

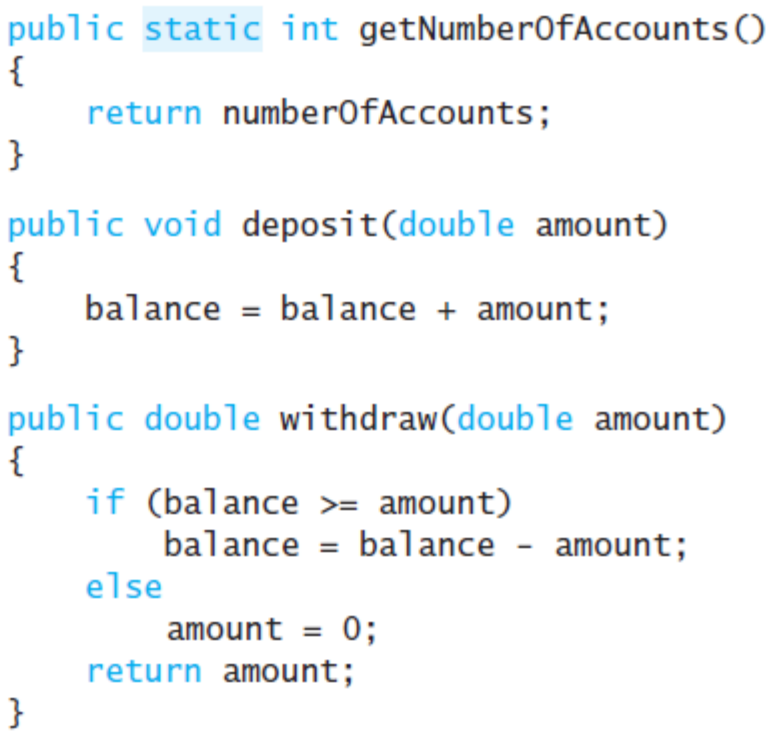
```
import java.util.Scanner;
/**
 Class with static and nonstatic members.
*/
public class SavingsAccount
{
    private double balance;
    public static double interestRate = 0;
    public static int numberOfAccounts = 0;
    public SavingsAccount()
    {
        balance = 0;
        numberOfAccounts++;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

    public static void setInterestRate(double newRate)
    {
        interestRate = newRate;
    }

    public static double getInterestRate()
    {
        return interestRate;
    }
```

An instance variable (nonstatic)

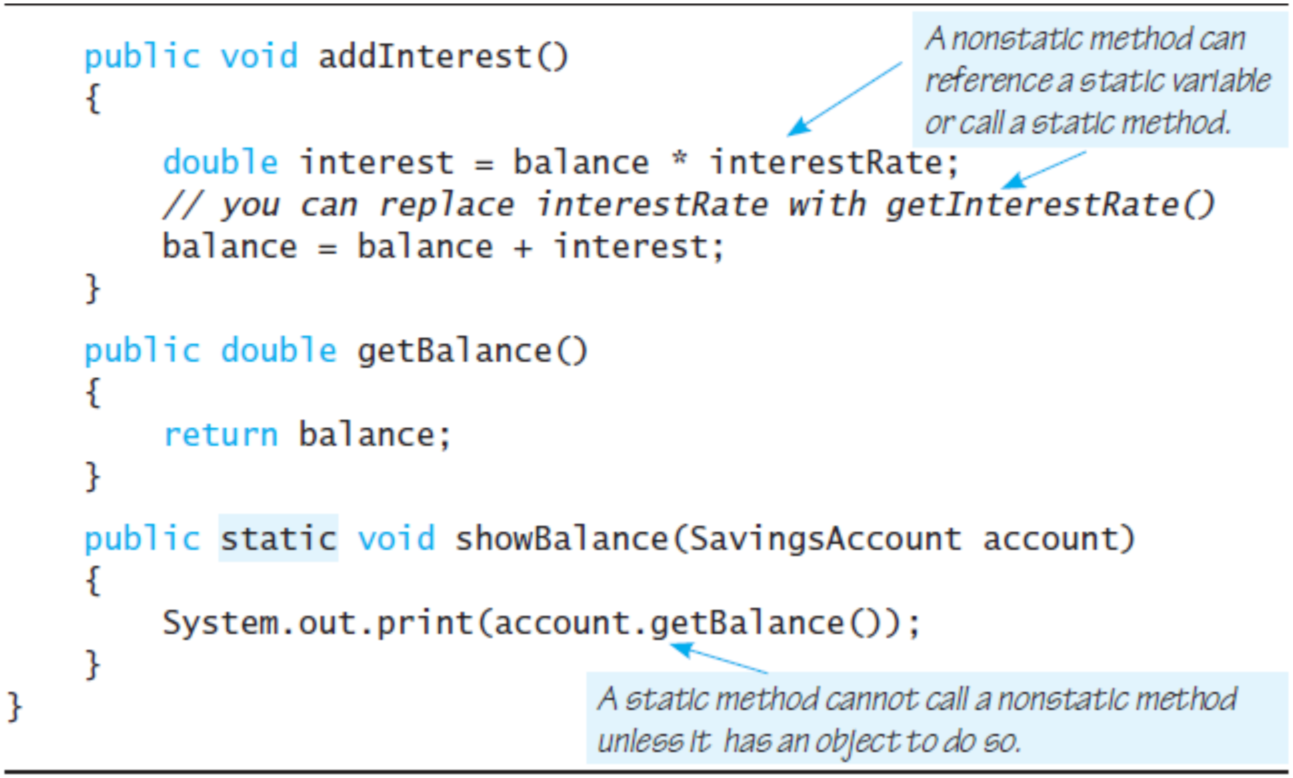
Static variables

A nonstatic method can reference a static variable.

A static method can reference a static variable but not an instance variable.



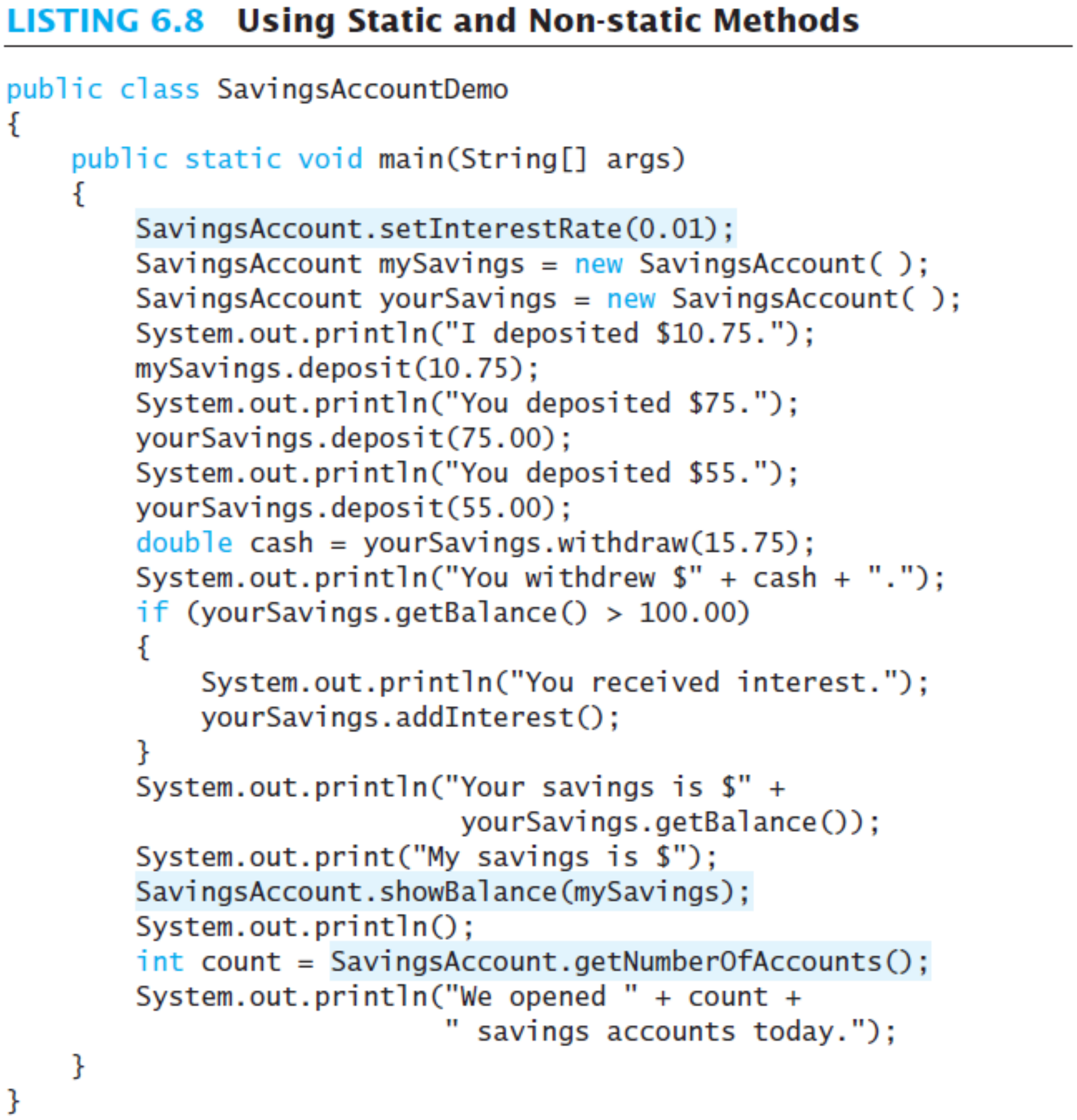
```
public static int getNumberOfAccounts()
{
    return numberOfAccounts;
}

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());
    }
}
```

A nonstatic method can reference a static variable or call a static method.

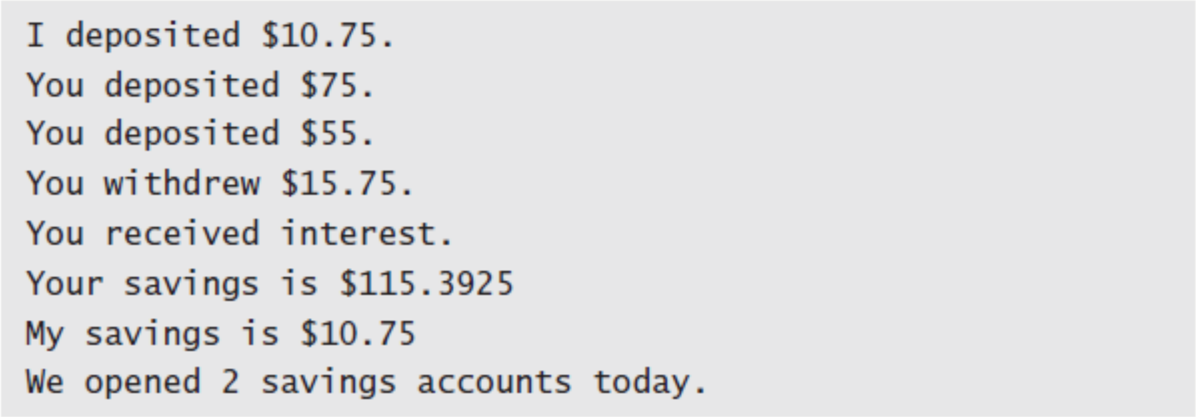
A static method cannot call a nonstatic method unless it has an object to do so.



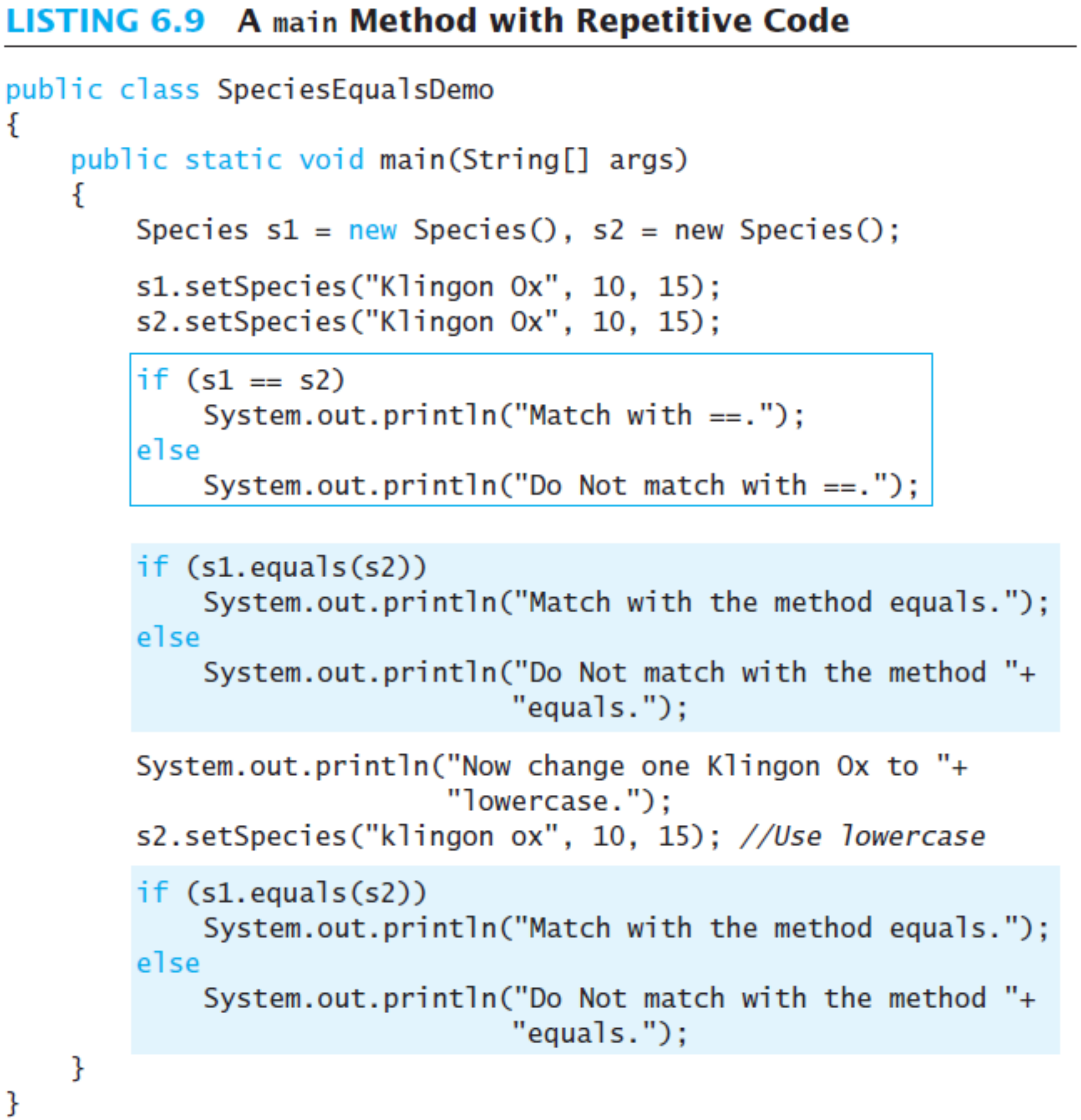
LISTING 6.8 Using Static and Non-static Methods

```java
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(15.75);
        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.");
    }
}
```

Screen Output

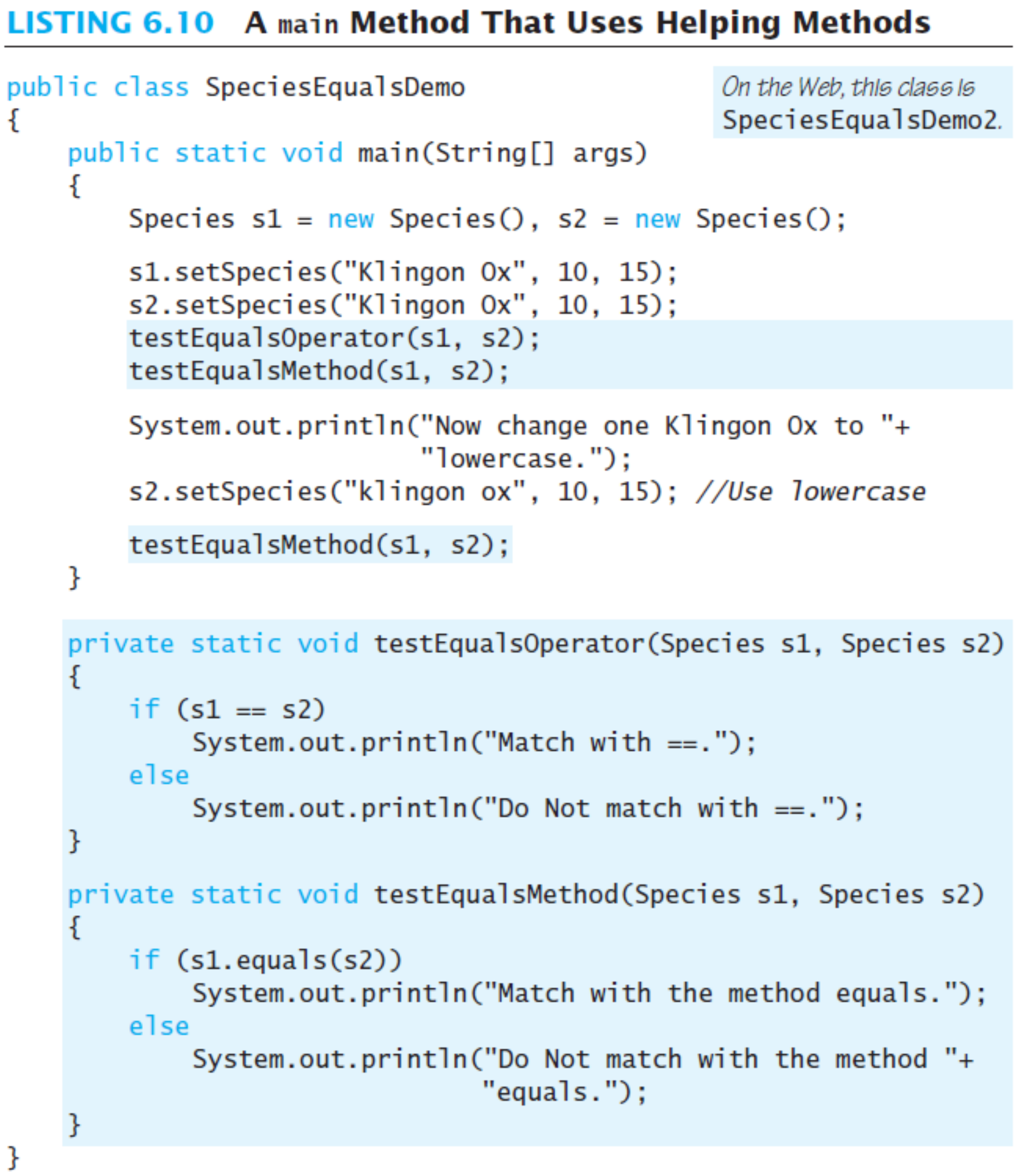


```
I deposited $10.75.
You deposited $75.
You deposited $55.
You withdrew $15.75.
You received interest.
Your savings is $115.3925
My savings is $10.75
We opened 2 savings accounts today.
```



LISTING 6.9 A `main` Method with Repetitive Code

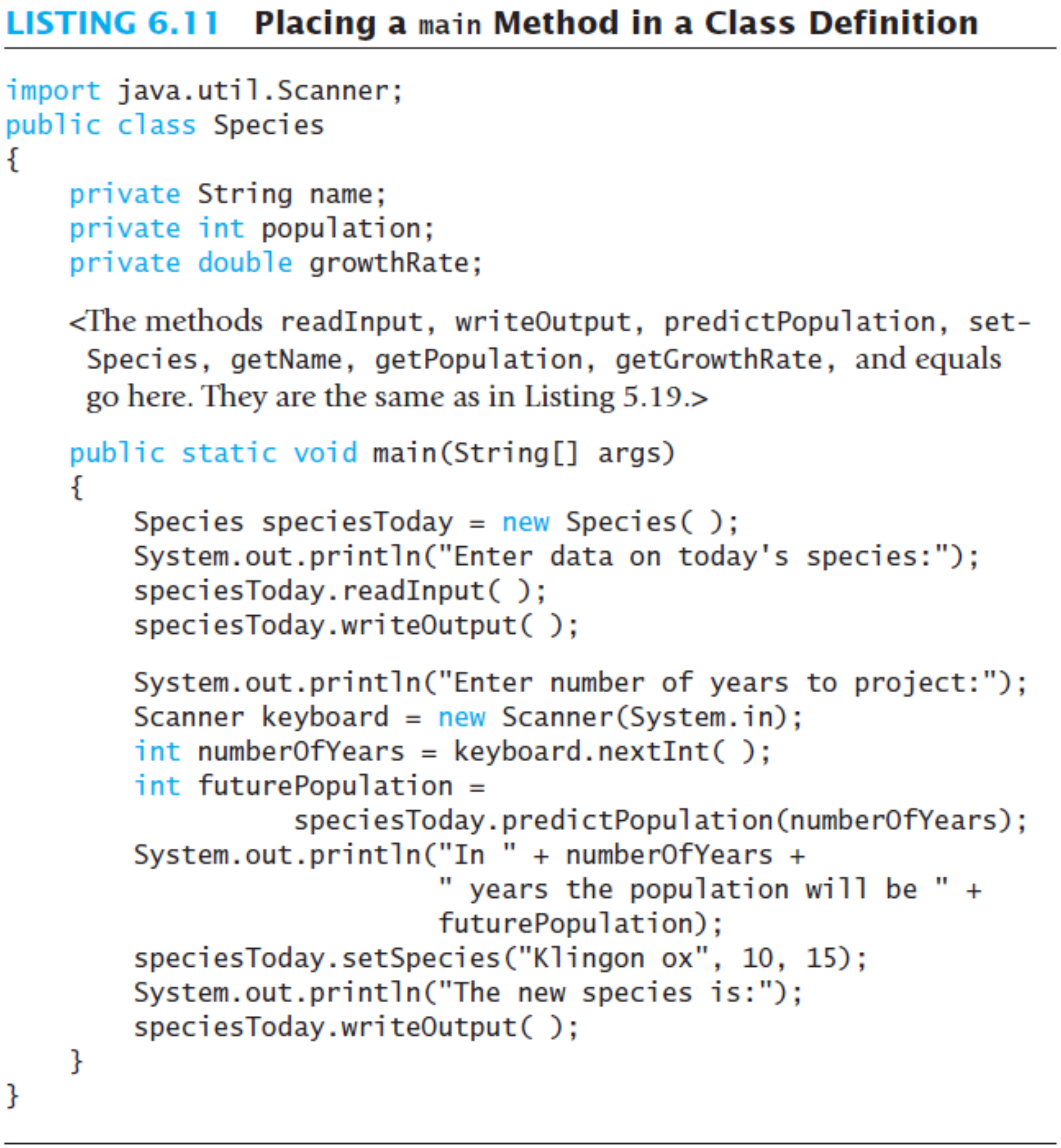
```
public class SpeciesEqualsDemo
{
    public static void main(String[] args)
    {
        Species s1 = new Species(), s2 = new Species();

        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 to "+
                              "lowercase.");
        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 A main Method That Uses Helping Methods

On the Web, this class is SpeciesEqualsDemo2.

```
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 to "+
                           "lowercase.");
        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.");
    }
}
```

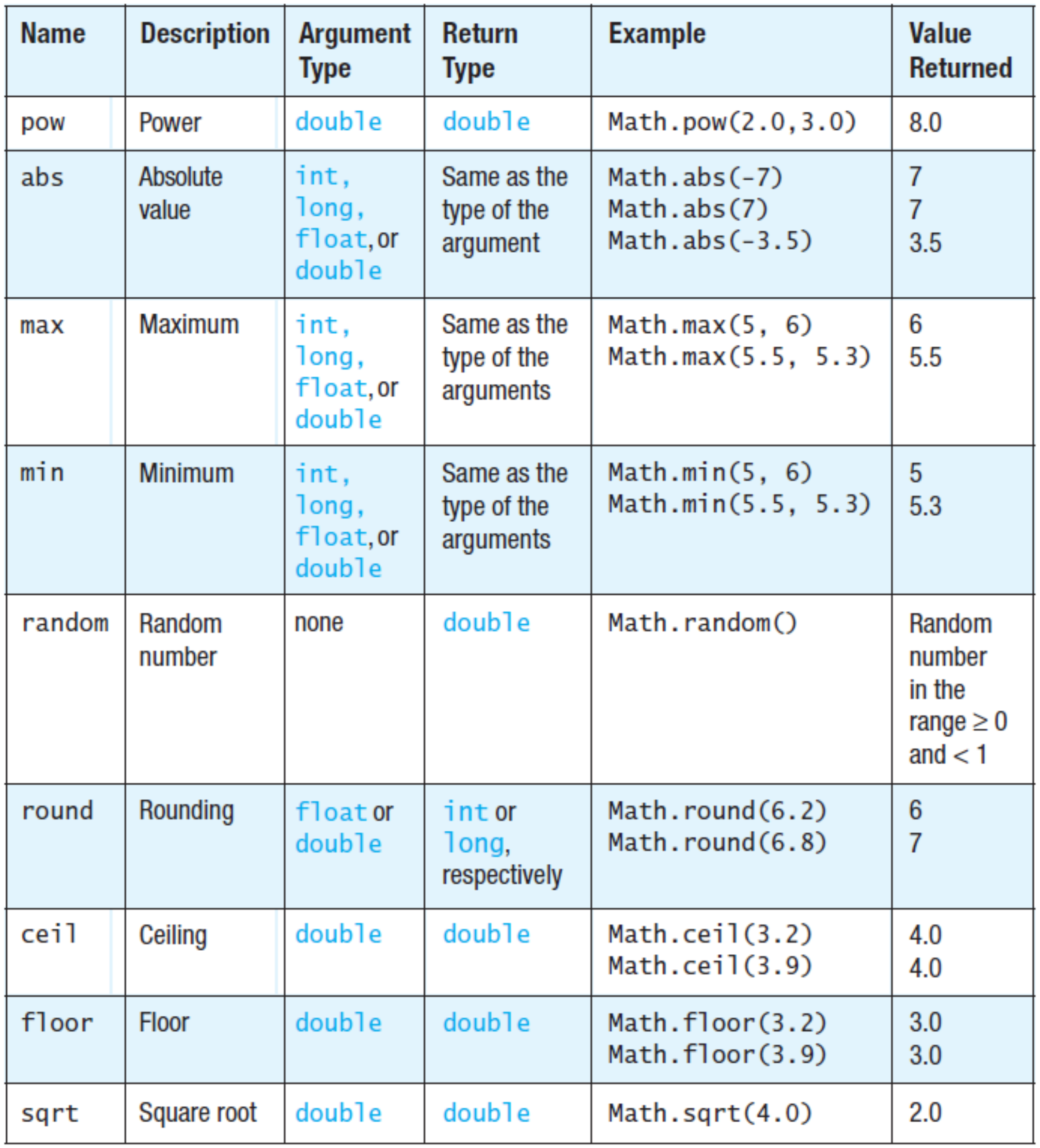


LISTING 6.11 Placing a `main` Method in a Class Definition

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

    <The methods readInput, writeOutput, predictPopulation, set-
     Species, getName, getPopulation, getGrowthRate, and equals
     go here. They are the same as in Listing 5.19.>

    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( );
    }
}
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

FIGURE 6.3 Static Methods in the Class `Math`



Name	Description	Argument Type	Return Type	Example	Value Returned
`pow`	Power	`double`	`double`	`Math.pow(2.0,3.0)`	8.0

`abs`	Absolute value	`int`, `long`, `float`, or `double`	Same as the type of the argument	`Math.abs(-7)`
				`Math.abs(7)`
				`Math.abs(-3.5)`
					7
					7
					3.5

`max`	Maximum	`int`, `long`, `float`, or `double`	Same as the type of the arguments	`Math.max(5, 6)`
				`Math.max(5.5, 5.3)`
					6
					5.5

`min`	Minimum	`int`, `long`, `float`, or `double`	Same as the type of the arguments	`Math.min(5, 6)`
				`Math.min(5.5, 5.3)`
					5
					5.3

`random`	Random number	none	`double`	`Math.random()`	Random number in the range ≥ 0 and < 1

`round`	Rounding	`float` or `double`	`int` or `long`, respectively	`Math.round(6.2)`
				`Math.round(6.8)`
					6
					7

`ceil`	Ceiling	`double`	`double`	`Math.ceil(3.2)`
				`Math.ceil(3.9)`
					4.0
					4.0

`floor`	Floor	`double`	`double`	`Math.floor(3.2)`
				`Math.floor(3.9)`
					3.0
					3.0

`sqrt`	Square root	`double`	`double`	`Math.sqrt(4.0)`	2.0

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch

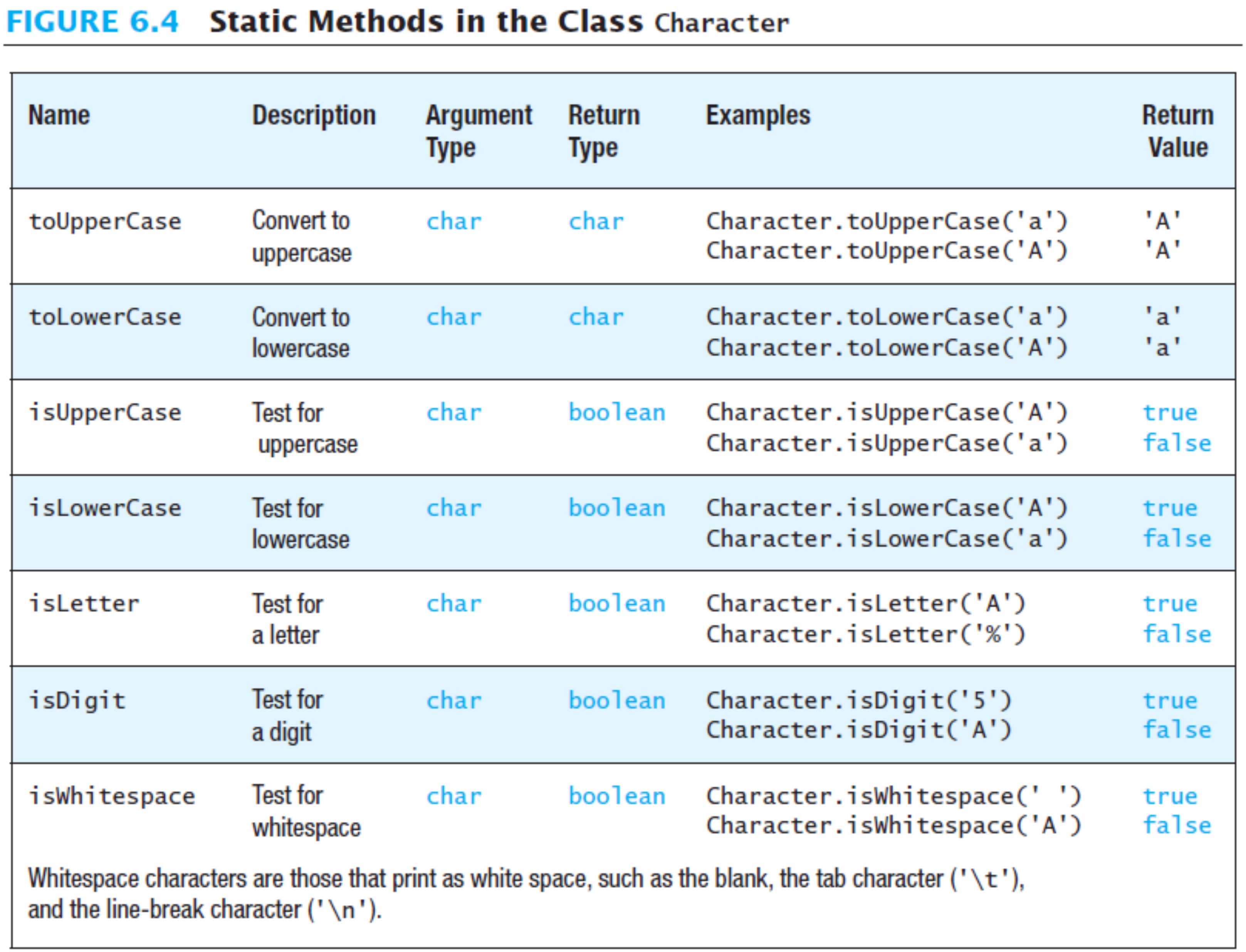


FIGURE 6.4 Static Methods in the Class `Character`

Name	Description	Argument Type	Return Type	Examples	Return Value

`toUpperCase`	Convert to uppercase	`char`	`char`	`Character.toUpperCase('a')`
				`Character.toUpperCase('A')`
					`'A'`
					`'A'`

`toLowerCase`	Convert to lowercase	`char`	`char`	`Character.toLowerCase('a')`
				`Character.toLowerCase('A')`
					`'a'`
					`'a'`

`isUpperCase`	Test for uppercase	`char`	`boolean`	`Character.isUpperCase('A')`
				`Character.isUpperCase('a')`
					`true`
					`false`

`isLowerCase`	Test for lowercase	`char`	`boolean`	`Character.isLowerCase('A')`
				`Character.isLowerCase('a')`
					`true`
					`false`

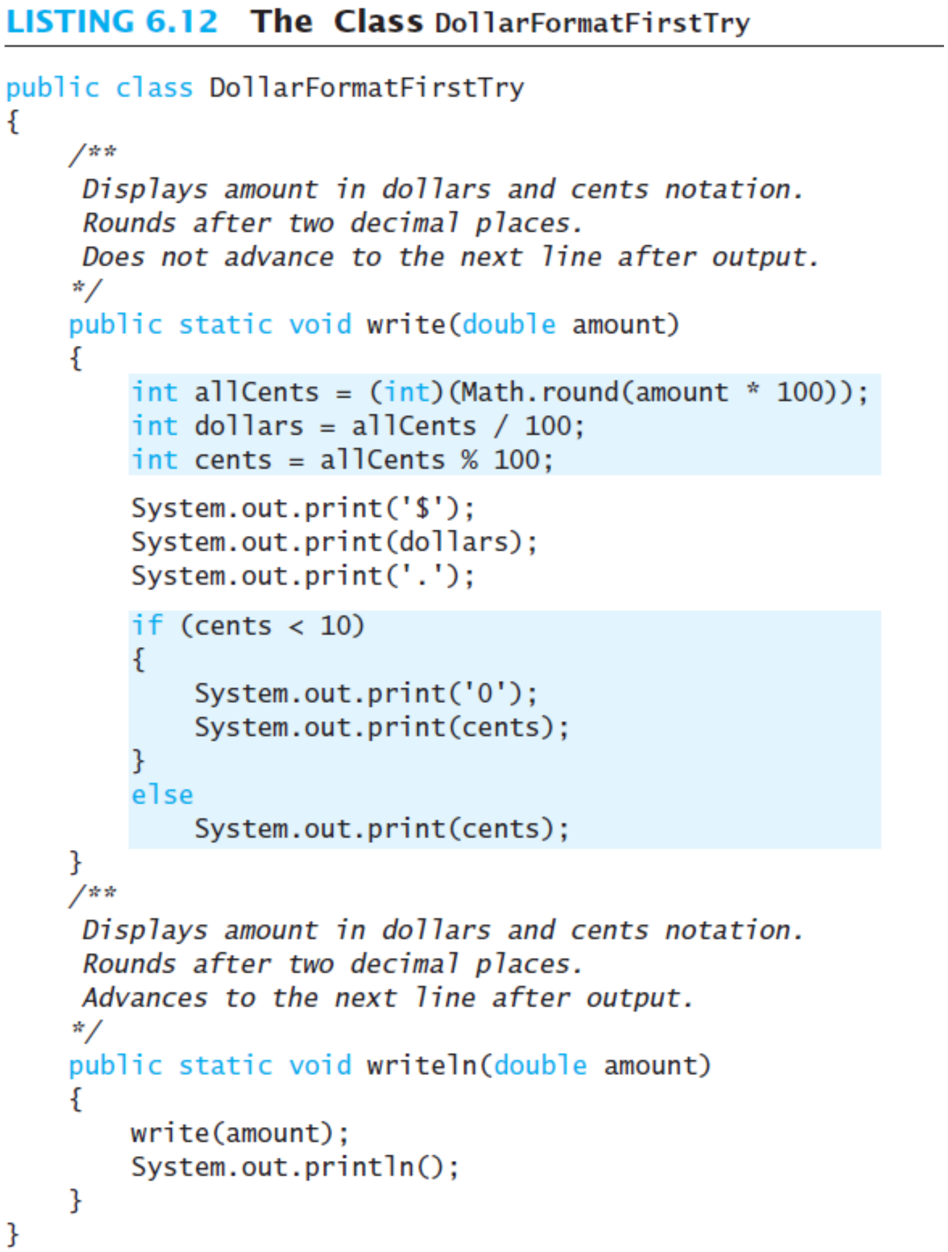
`isLetter`	Test for a letter	`char`	`boolean`	`Character.isLetter('A')`
				`Character.isLetter('%')`
					`true`
					`false`

`isDigit`	Test for a digit	`char`	`boolean`	`Character.isDigit('5')`
				`Character.isDigit('A')`
					`true`
					`false`

`isWhitespace`	Test for whitespace	`char`	`boolean`	`Character.isWhitespace(' ')`
				`Character.isWhitespace('A')`
					`true`
					`false`

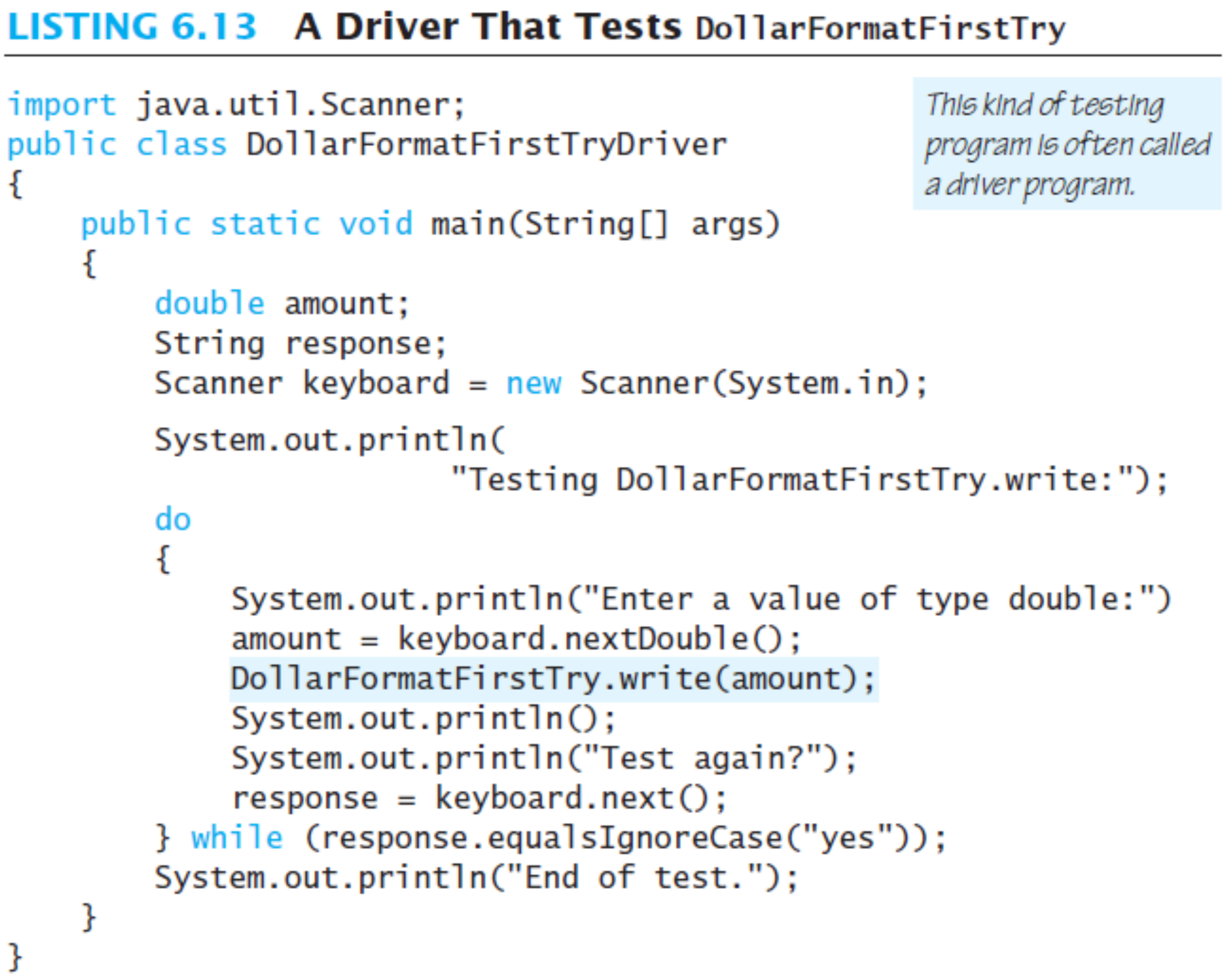
Whitespace characters are those that print as white space, such as the blank, the tab character (`'\t'`), and the line-break character (`'\n'`).

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



LISTING 6.12 The Class DollarFormatFirstTry

```
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();
    }
}
```

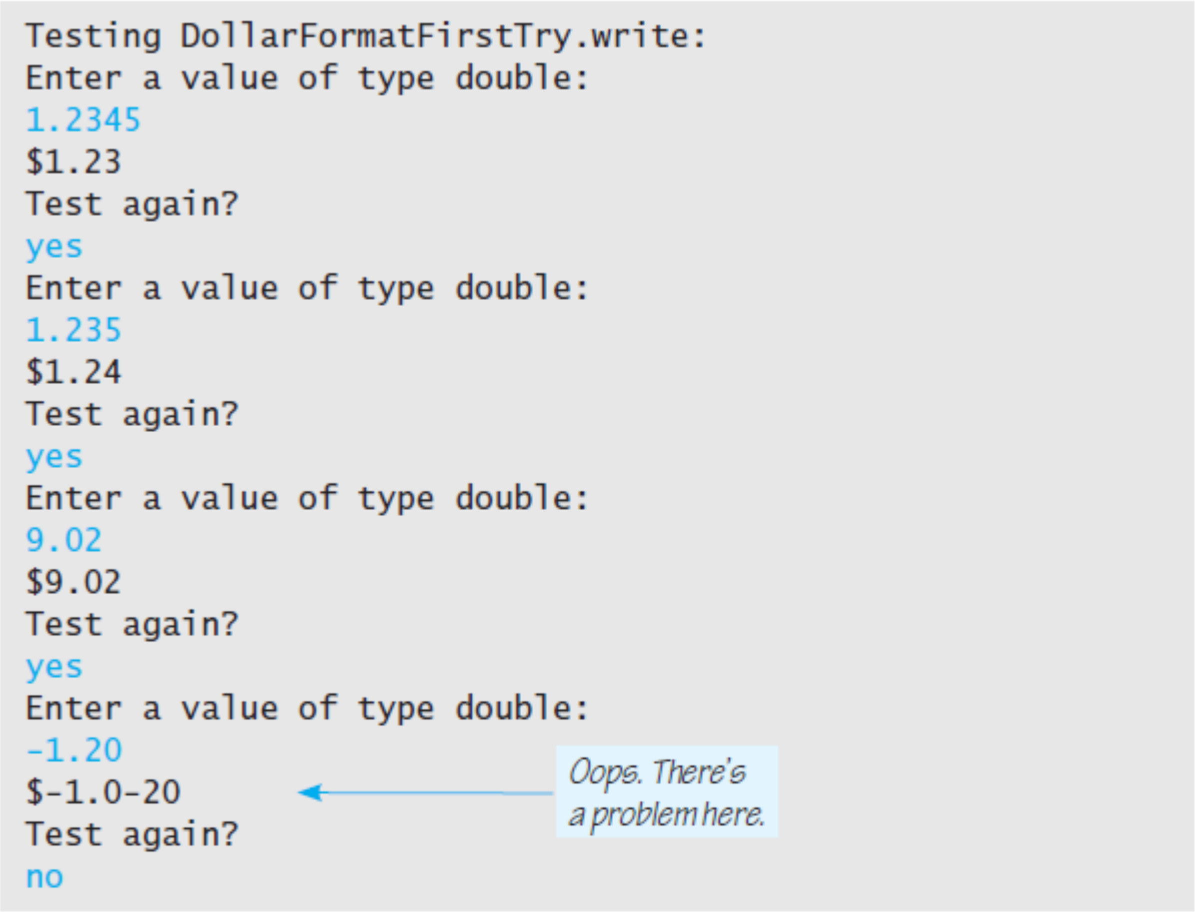


LISTING 6.13 A Driver That Tests DollarFormatFirstTry

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

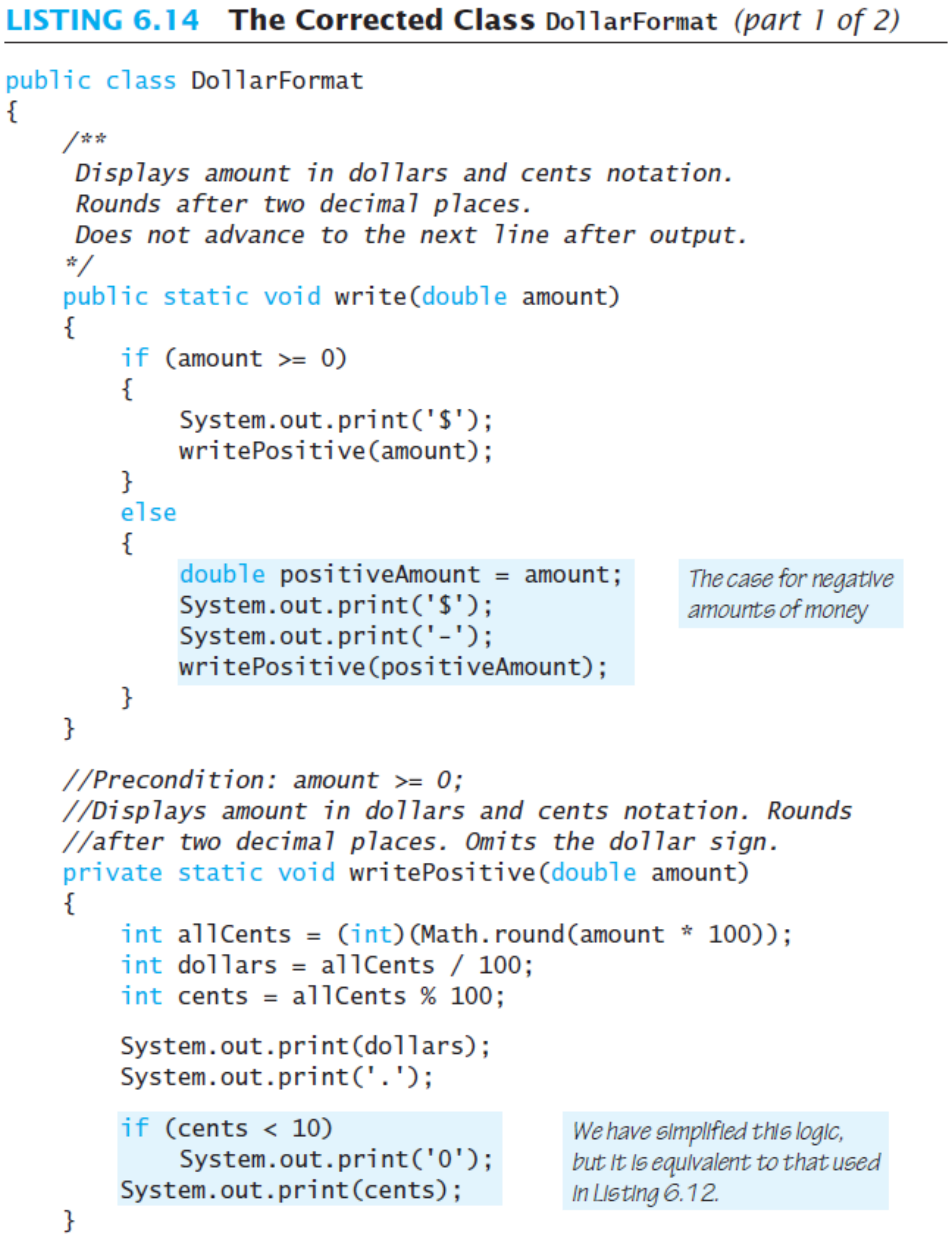
This kind of testing program is often called a driver program.

Sample Screen Output



```
Testing DollarFormatFirstTry.write:
Enter a value of type double:
1.2345
$1.23
Test again?
yes
Enter a value of type double:
1.235
$1.24
Test again?
yes
Enter a value of type double:
9.02
$9.02
Test again?
yes
Enter a value of type double:
-1.20
$-1.0-20
Test again?
no
```

Oops. There's a problem here.

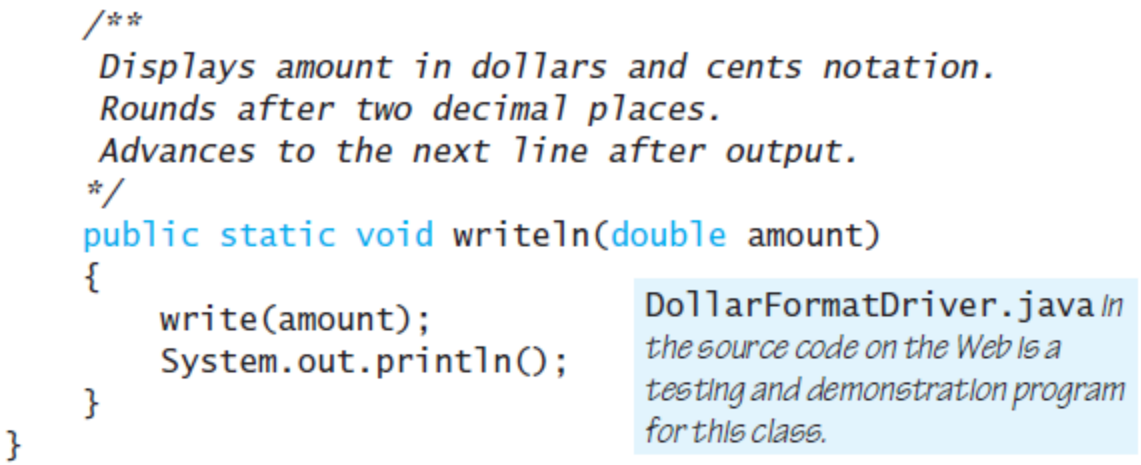


LISTING 6.14 The Corrected Class `DollarFormat` *(part 1 of 2)*

```
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);
    }
```

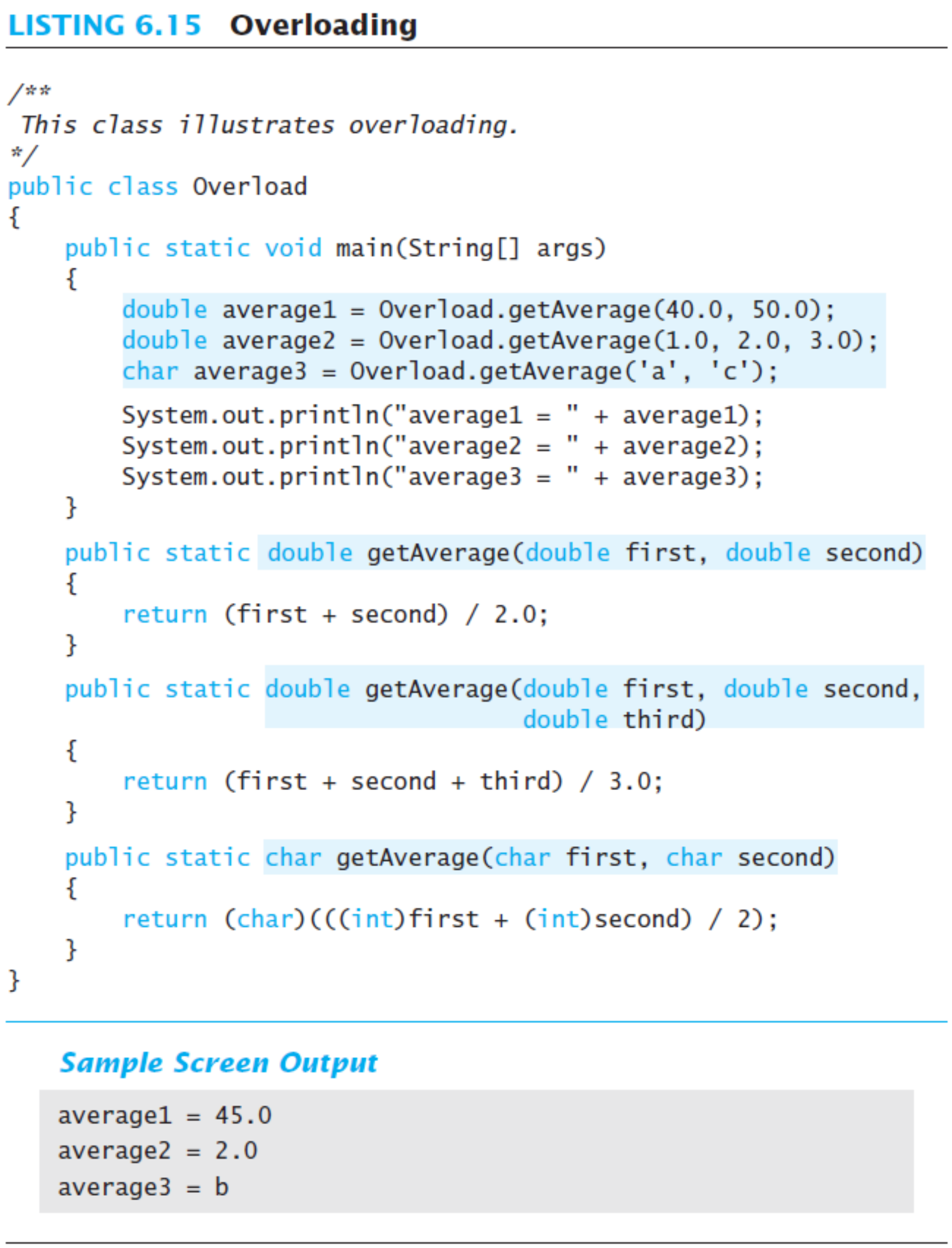
The case for negative amounts of money

We have simplified this logic, but it is equivalent to that used in Listing 6.12.



```
    /**
     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();
    }
}
```

`DollarFormatDriver.java` *In the source code on the Web is a testing and demonstration program for this class.*

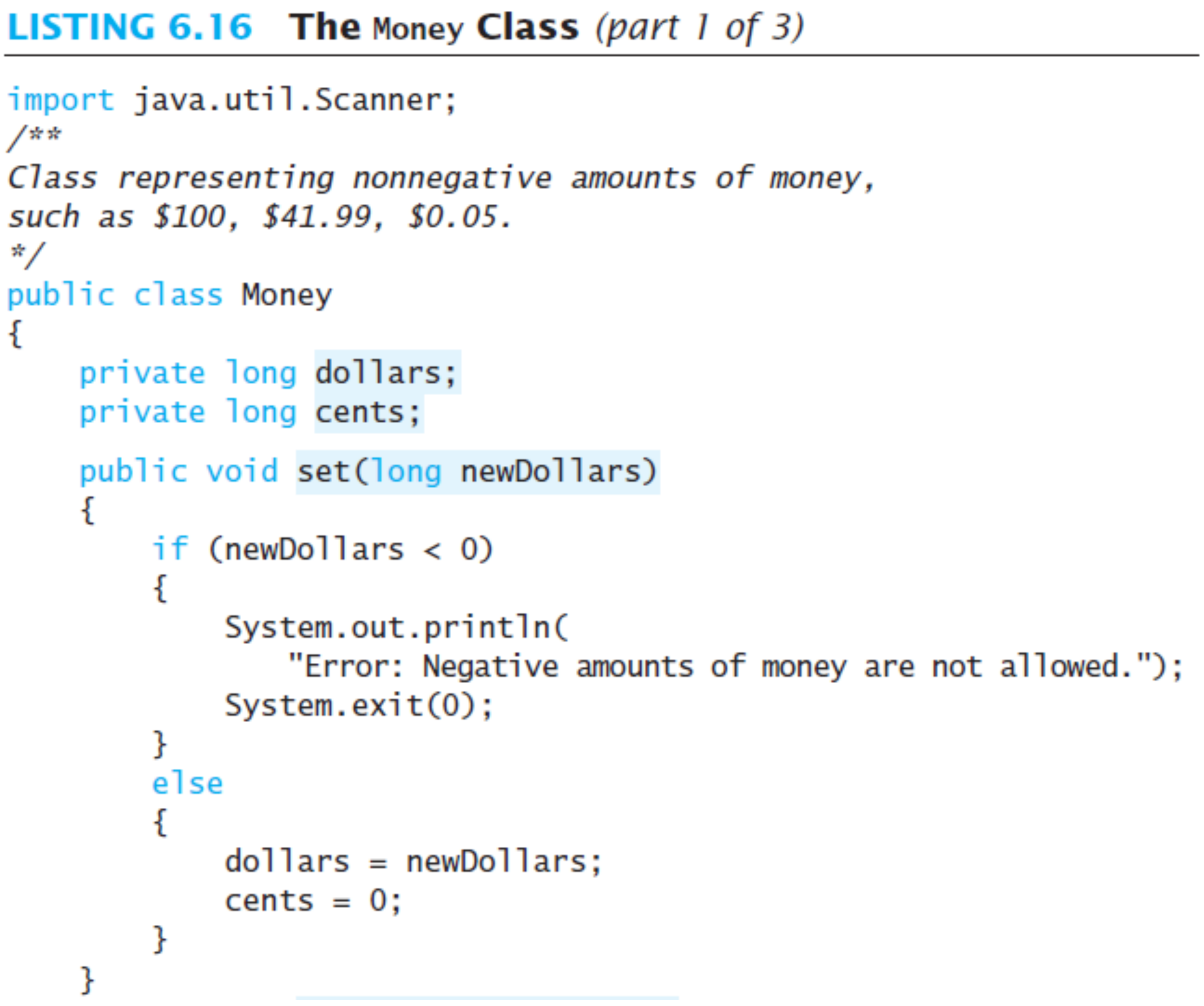


LISTING 6.15 Overloading

```
/**
 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);
    }
}
```

Sample Screen Output

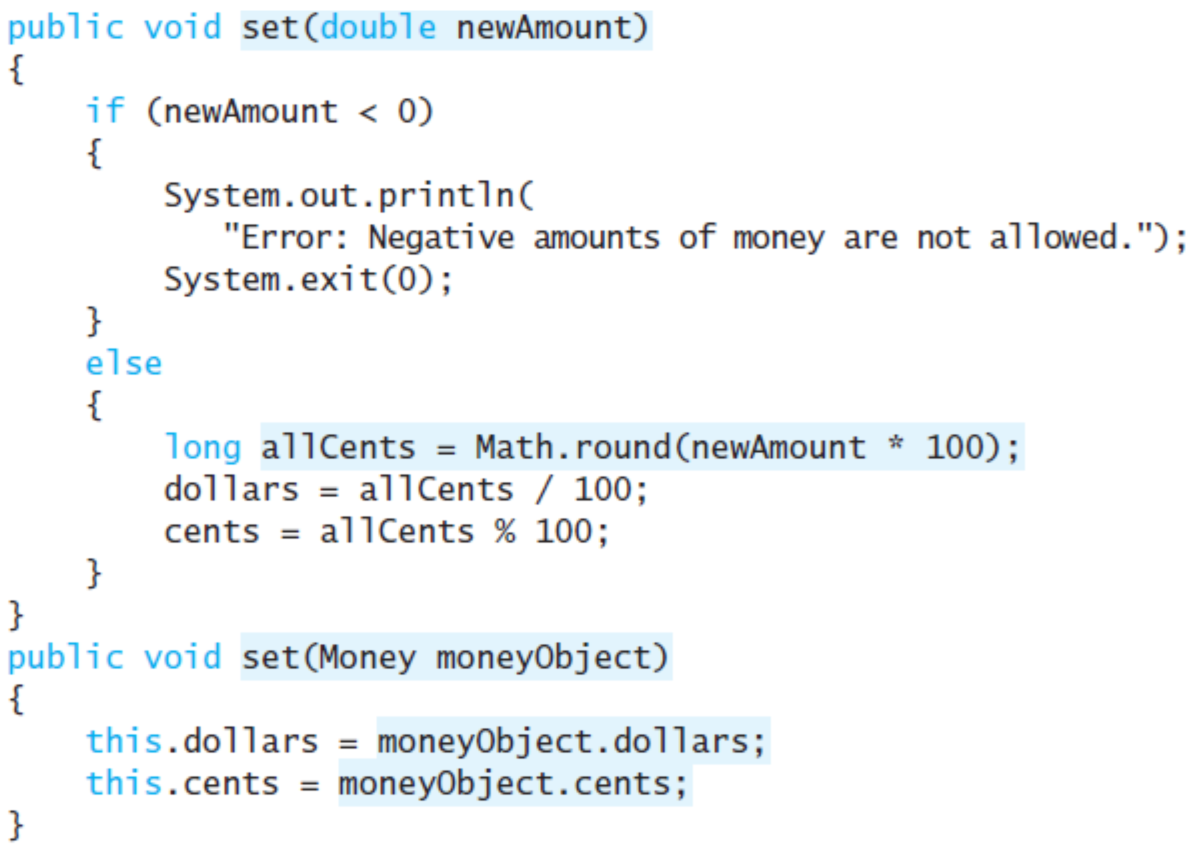
```
average1 = 45.0
average2 = 2.0
average3 = b
```



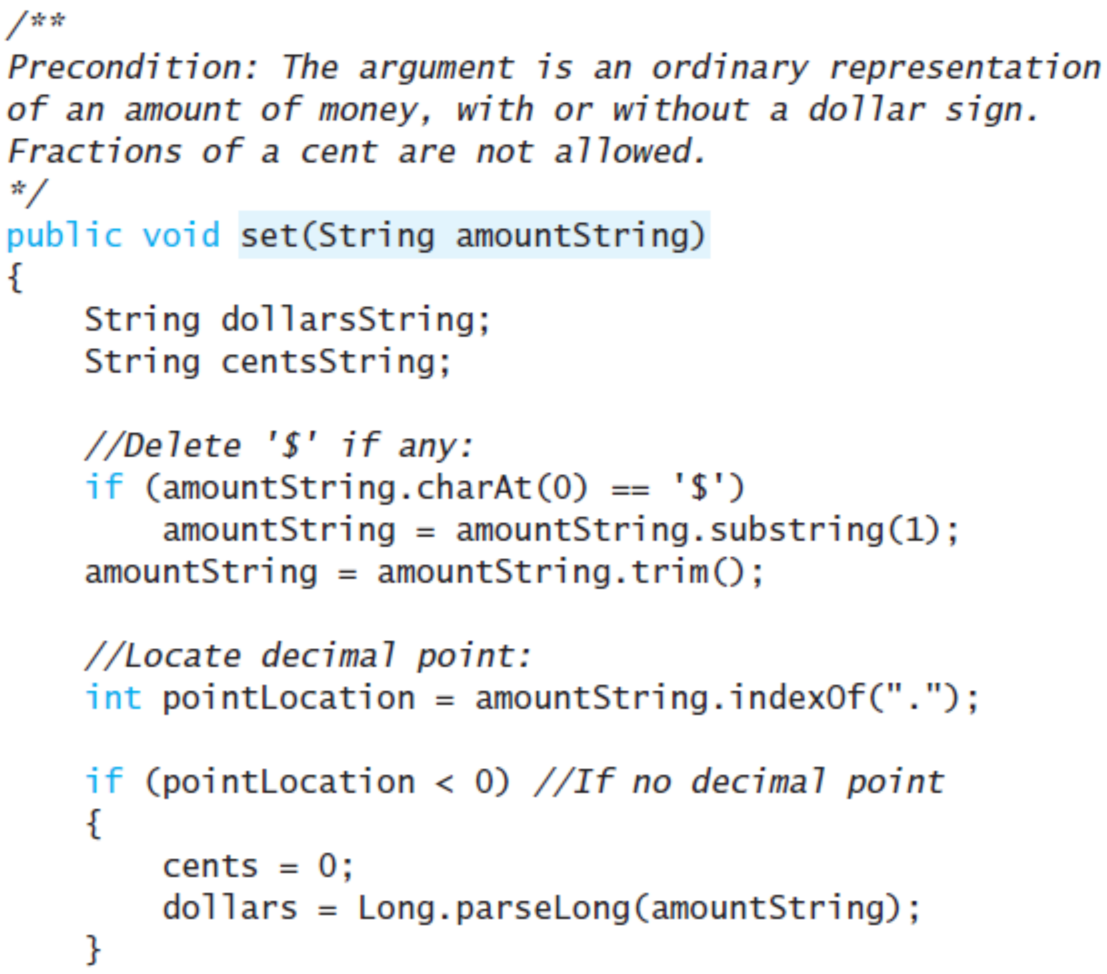
LISTING 6.16 The Money Class *(part 1 of 3)*

```
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;
        }
    }
```

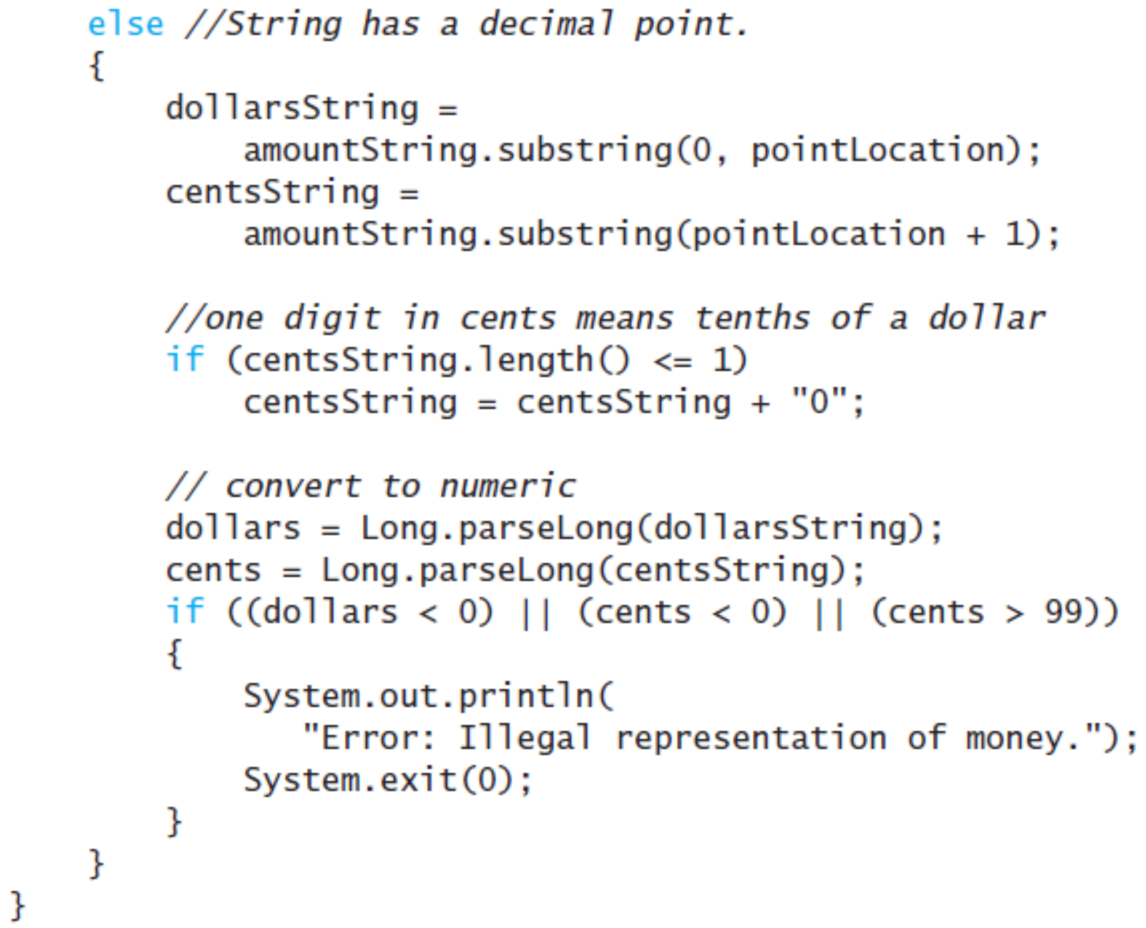
JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



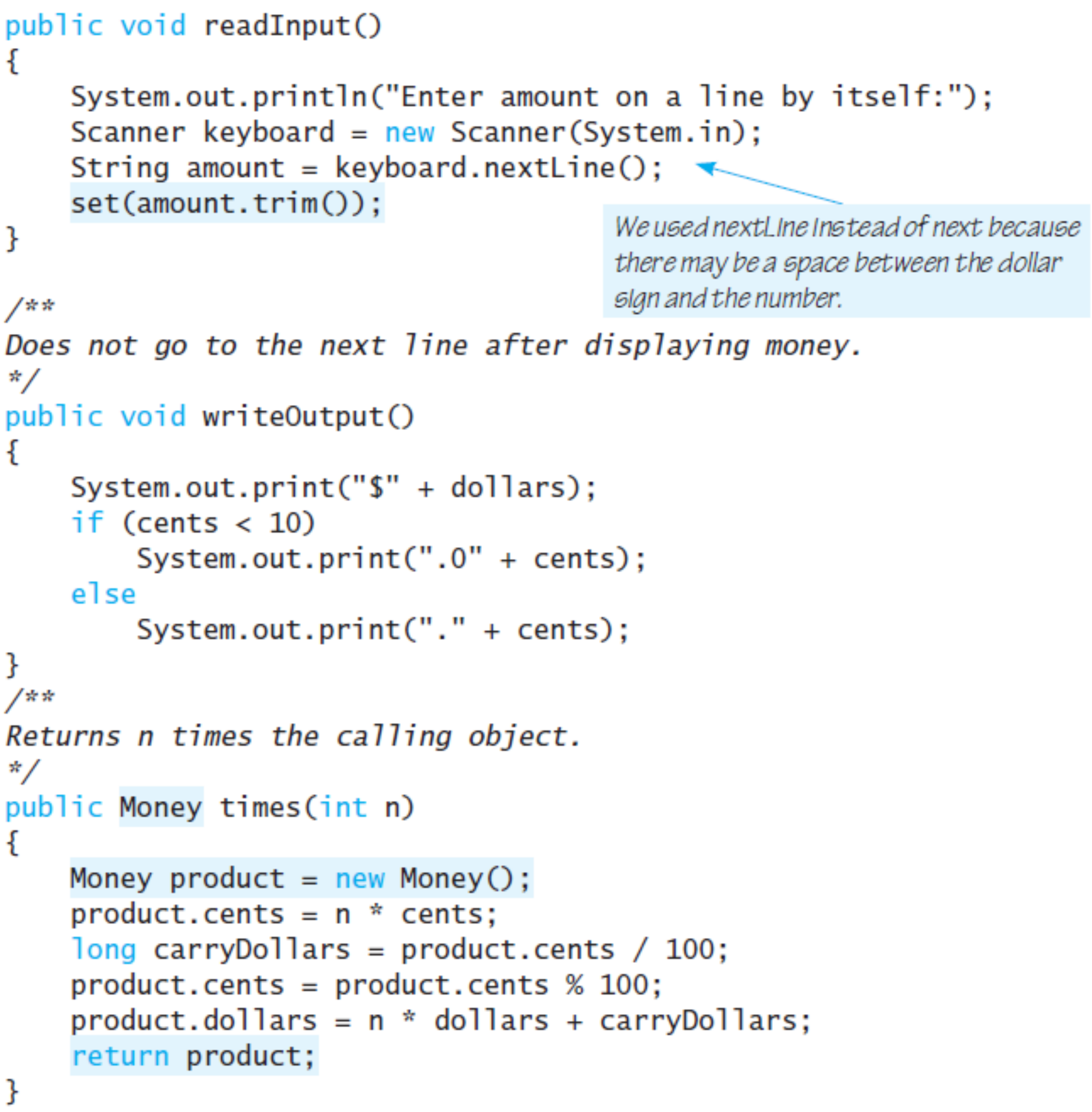
```
public void set(double newAmount)
{
    if (newAmount < 0)
    {
        System.out.println(
           "Error: Negative amounts of money are not allowed.");
        System.exit(0);
    }
    else
    {
        long allCents = Math.round(newAmount * 100);
        dollars = allCents / 100;
        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);
    }
```



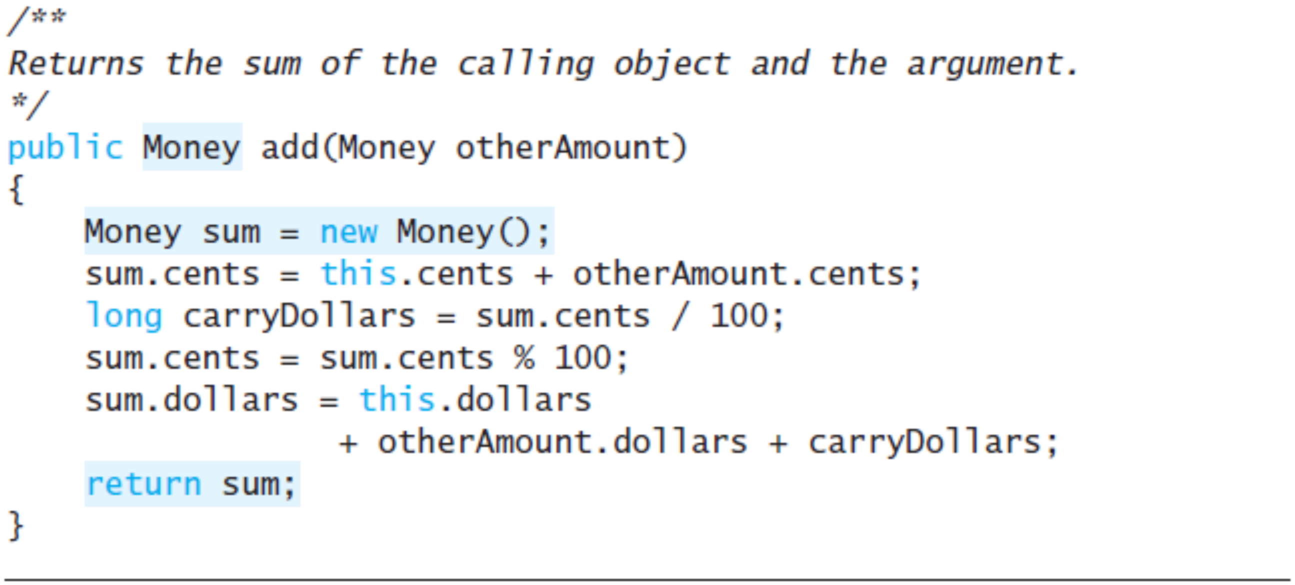
```
    else //String has a decimal point.
    {
        dollarsString =
            amountString.substring(0, pointLocation);
        centsString =
            amountString.substring(pointLocation + 1);

        //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);
        }
    }
}
```



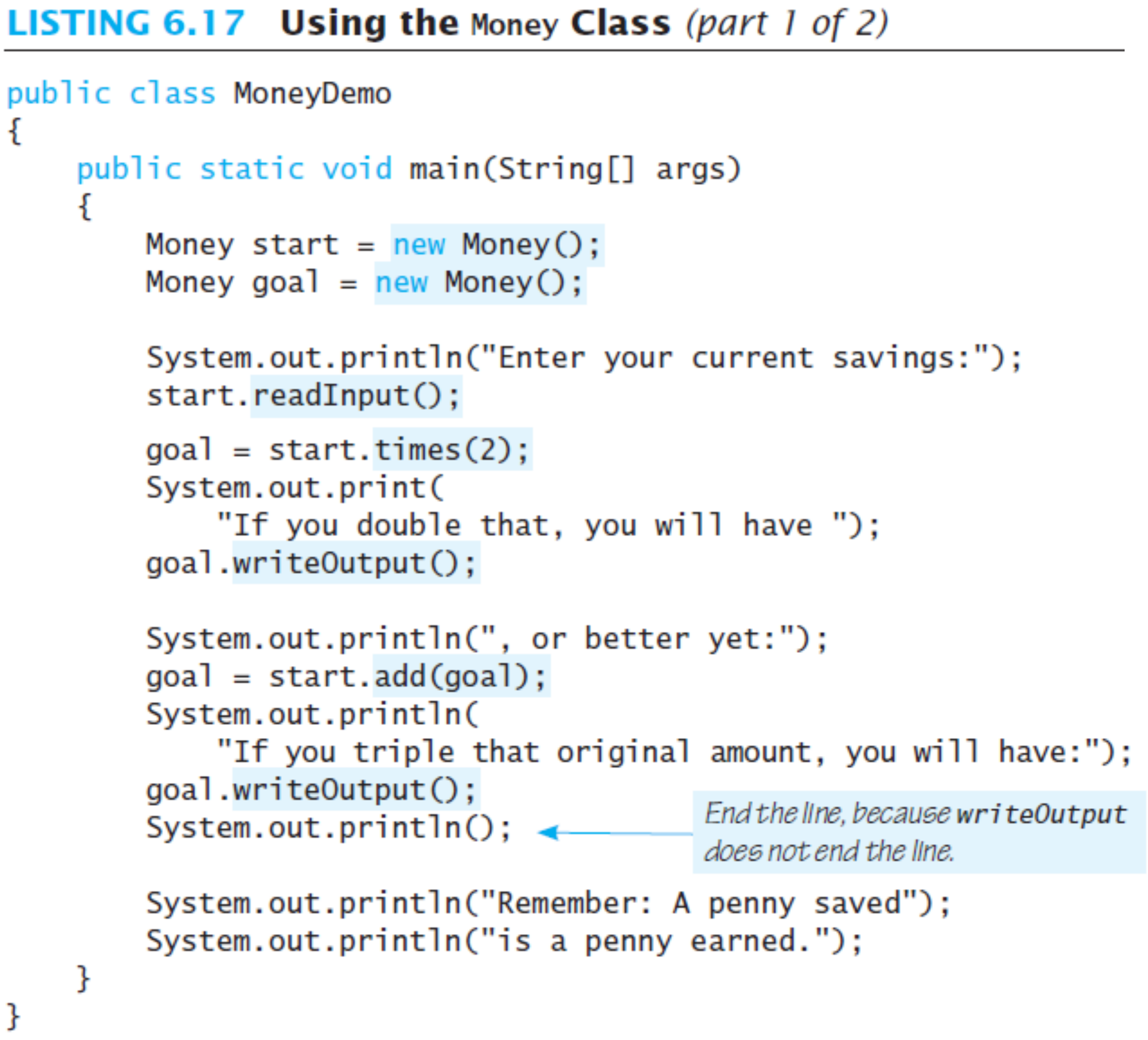
```
public void readInput()
{
    System.out.println("Enter amount on a line by itself:");
    Scanner keyboard = new Scanner(System.in);
    String amount = keyboard.nextLine();
    set(amount.trim());
}

/**
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;
}
```

We used nextLine instead of next because there may be a space between the dollar sign and the number.

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



```
/**
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;
}
```



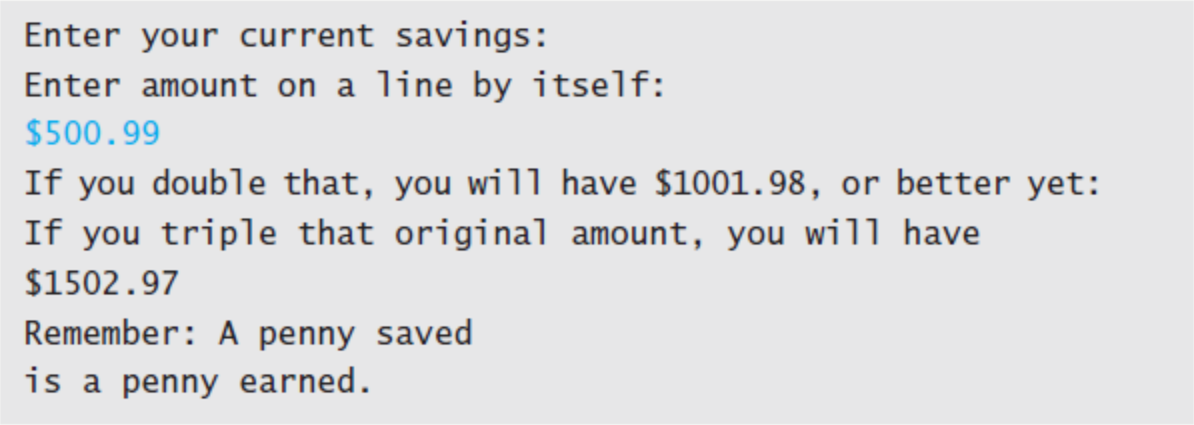
LISTING 6.17 Using the Money Class *(part 1 of 2)*

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

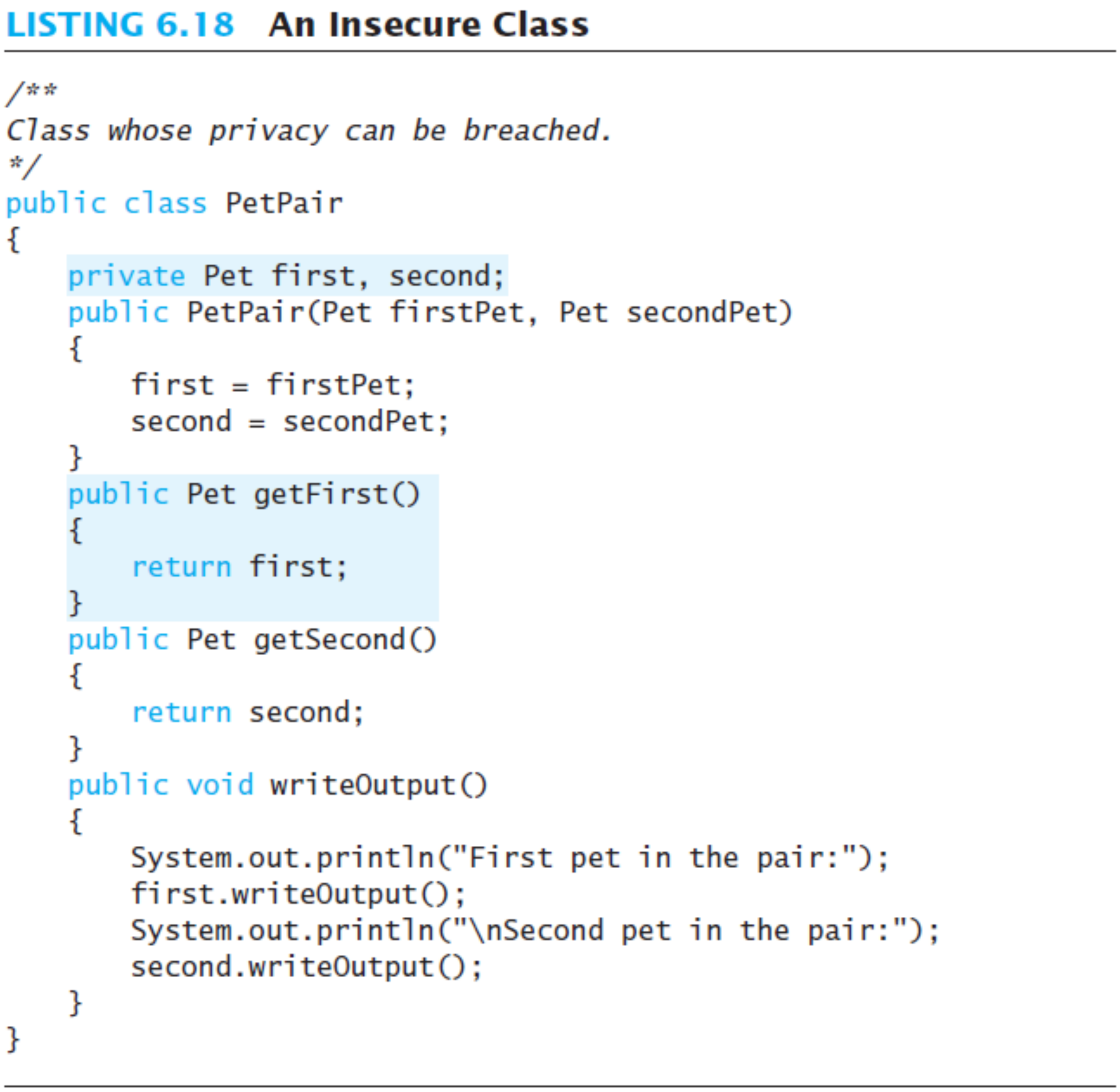
End the line, because `writeOutput` *does not end the line.*

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Sample Screen Output



```
Enter your current savings:
Enter amount on a line by itself:
$500.99
If you double that, you will have $1001.98, or better yet:
If you triple that original amount, you will have
$1502.97
Remember: A penny saved
is a penny earned.
```

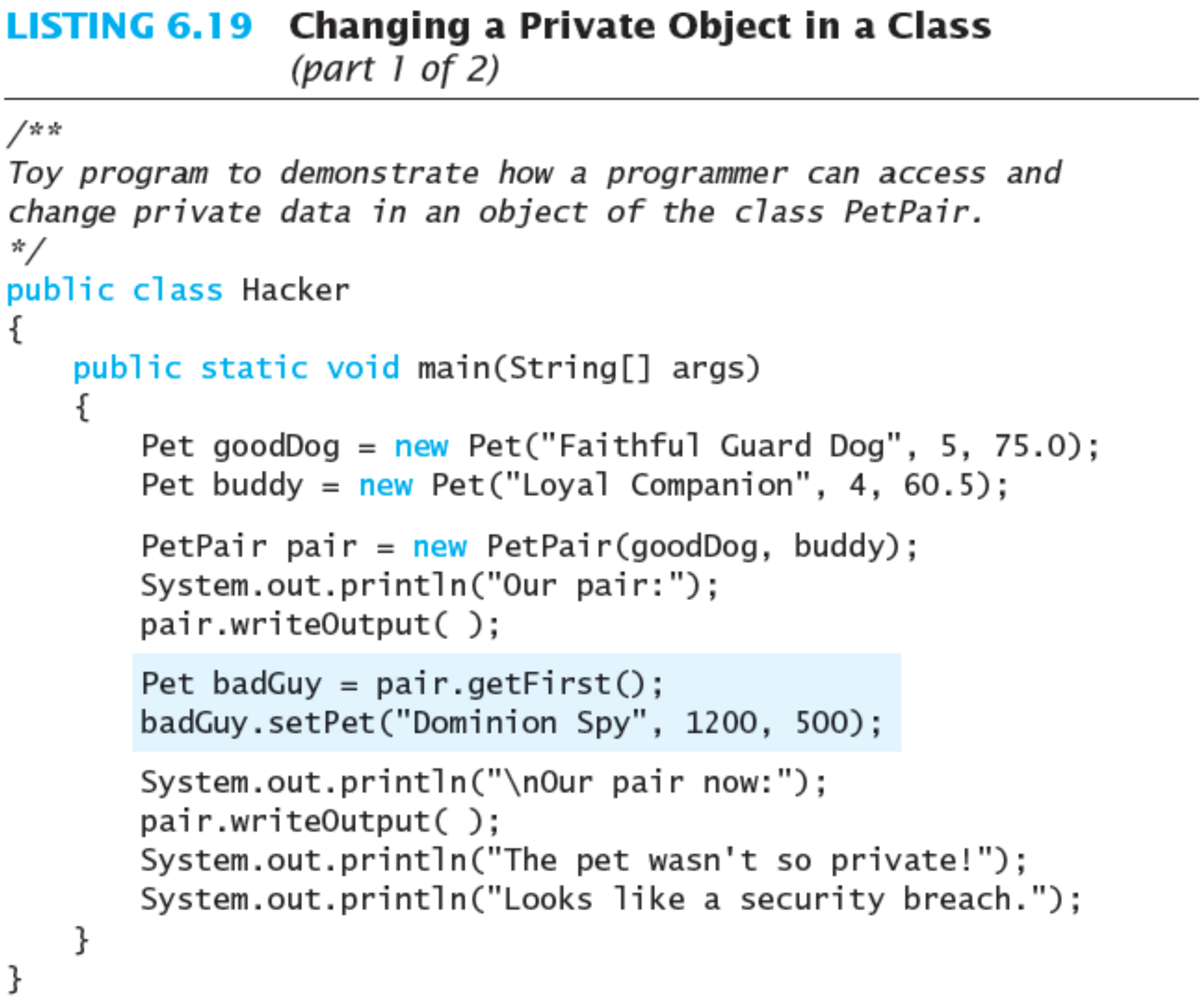


LISTING 6.18 An Insecure Class

```
/**
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();
    }
}
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch

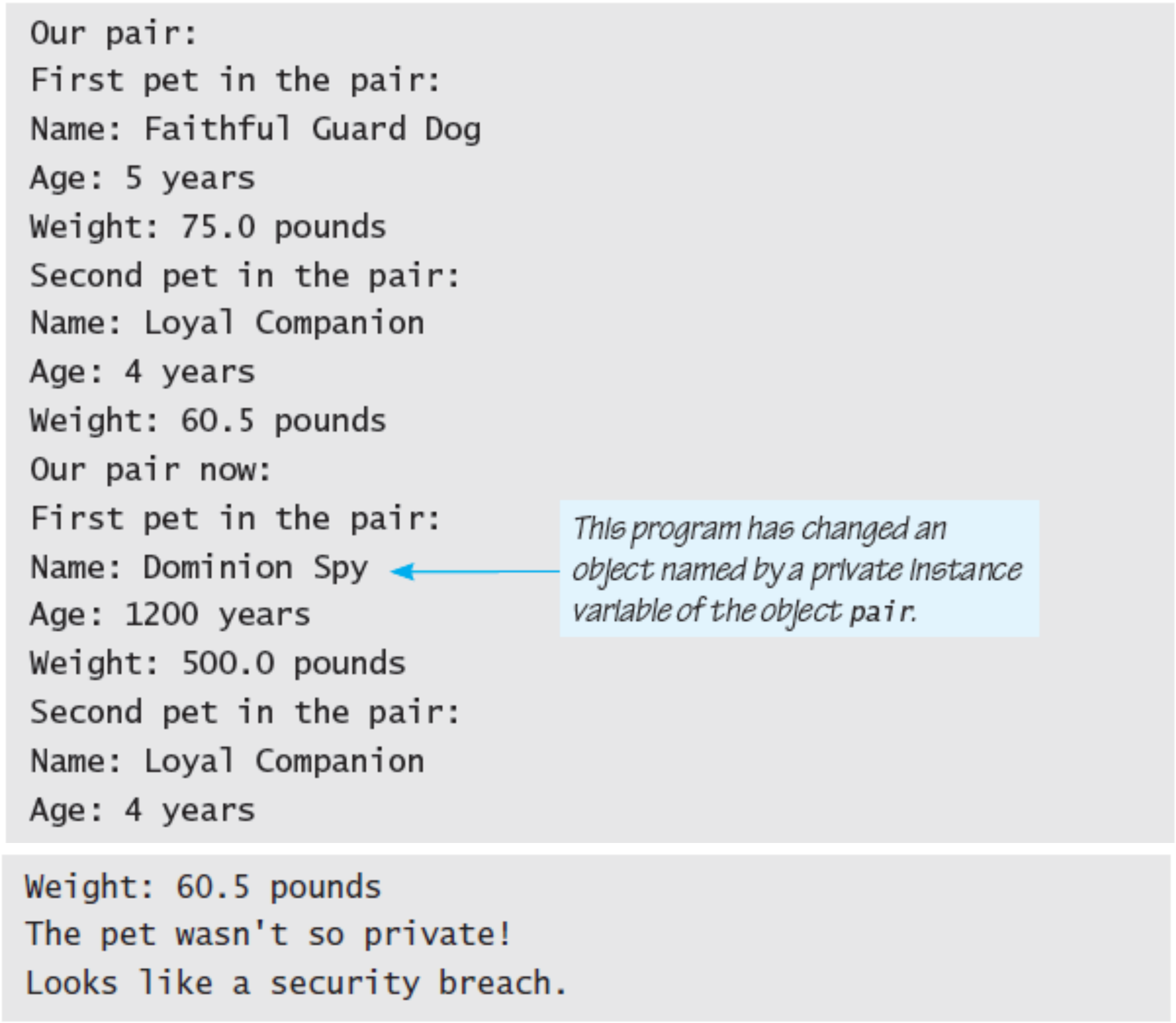
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



LISTING 6.19 Changing a Private Object in a Class *(part 1 of 2)*

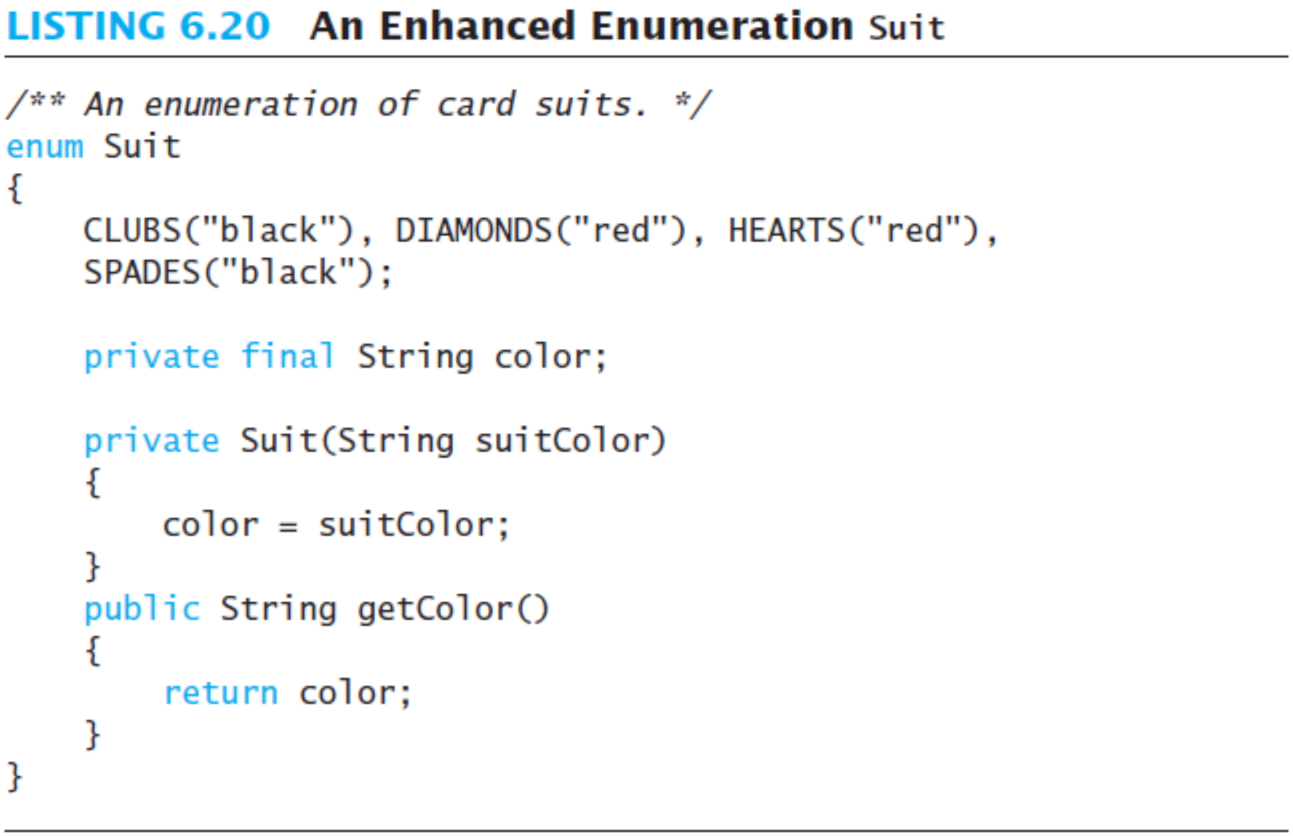
```
/**
Toy program to demonstrate how a programmer can access and
change private data in an object of the class PetPair.
*/
public class Hacker
{
    public static void main(String[] args)
    {
        Pet goodDog = new Pet("Faithful Guard Dog", 5, 75.0);
        Pet buddy = new Pet("Loyal Companion", 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.");
    }
}
```

Screen Output



```
Our pair:
First pet in the pair:
Name: Faithful Guard Dog
Age: 5 years
Weight: 75.0 pounds
Second pet in the pair:
Name: Loyal Companion
Age: 4 years
Weight: 60.5 pounds
Our pair now:
First pet in the pair:
Name: Dominion Spy
Age: 1200 years
Weight: 500.0 pounds
Second pet in the pair:
Name: Loyal Companion
Age: 4 years
Weight: 60.5 pounds
The pet wasn't so private!
Looks like a security breach.
```

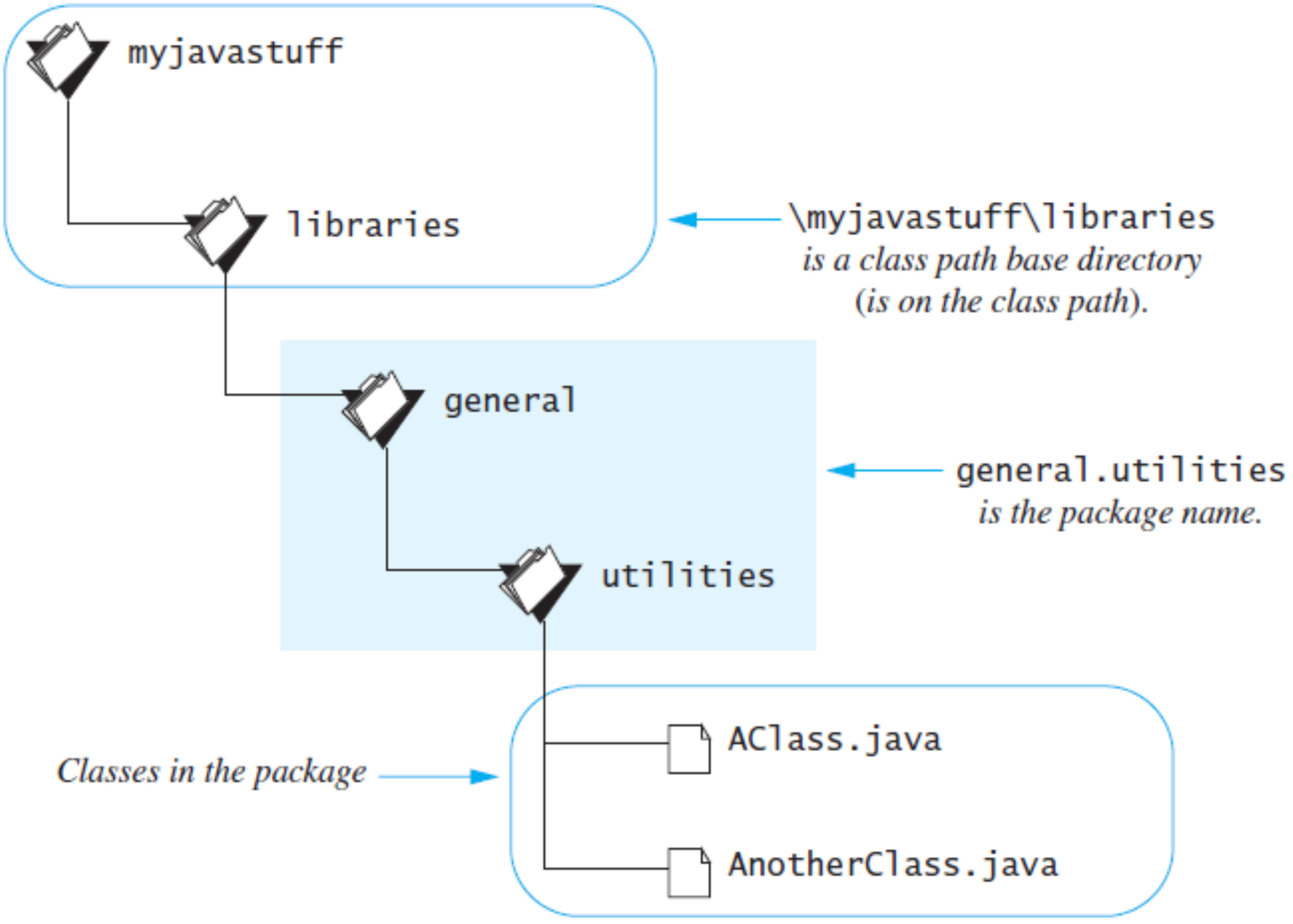
This program has changed an object named by a private instance variable of the object `pair`.



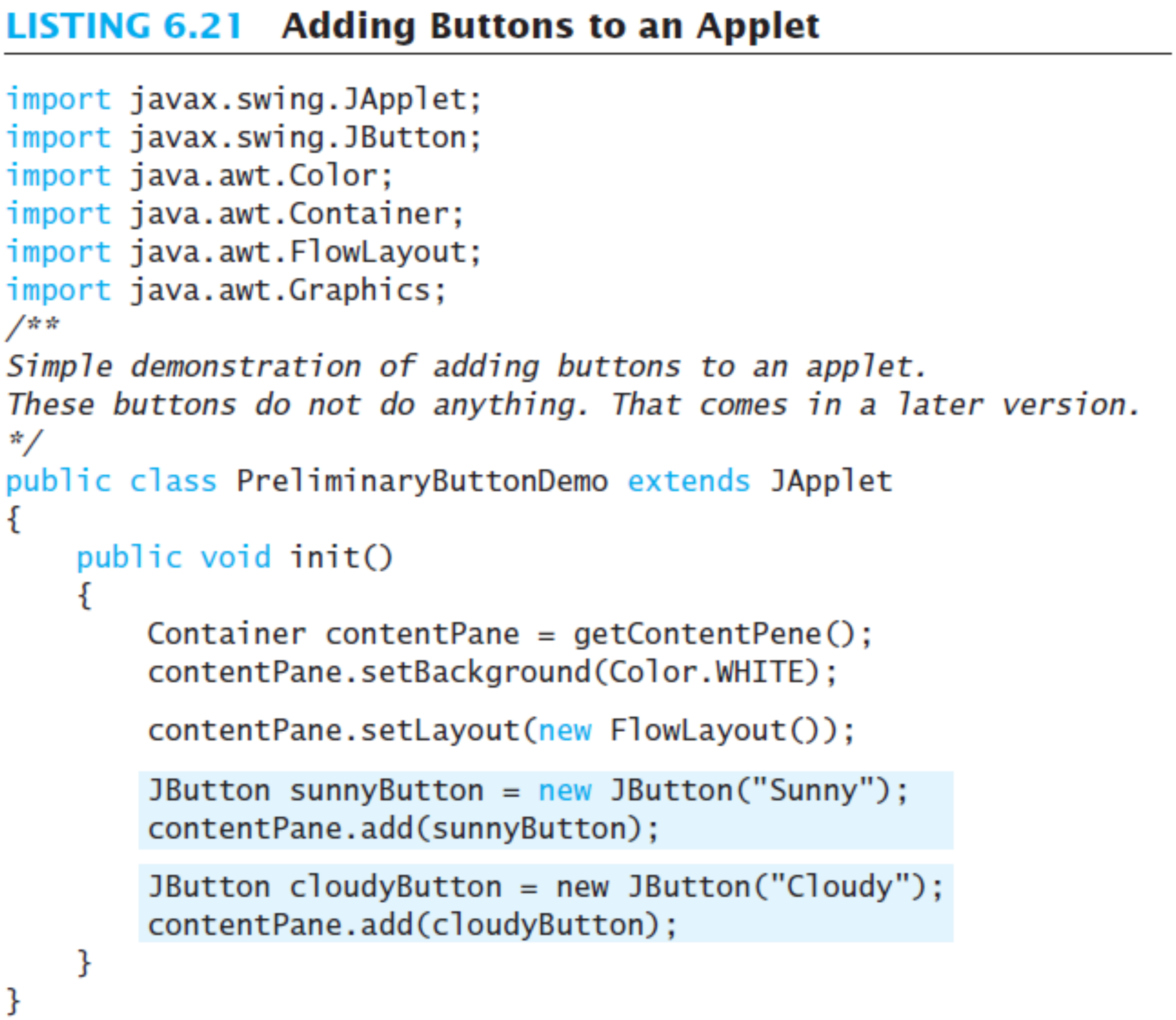
LISTING 6.20 An Enhanced Enumeration Suit

```
/** 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;
    }
}
```

FIGURE 6.5 A Package Name



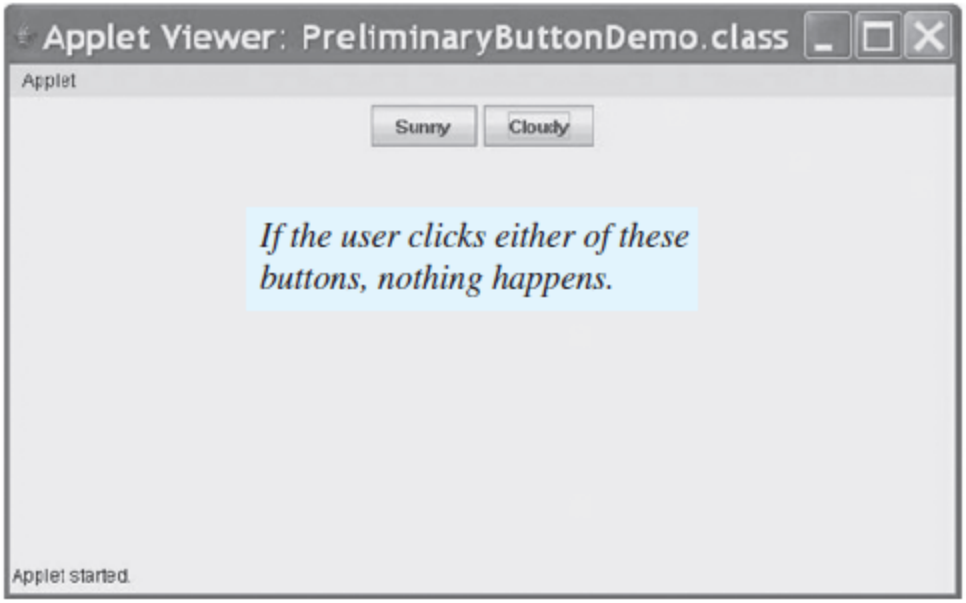
JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



LISTING 6.21 Adding Buttons to an Applet

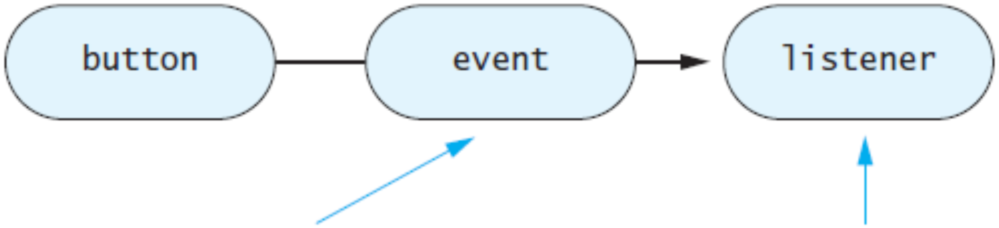
```
import javax.swing.JApplet;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Graphics;
/**
Simple demonstration of adding buttons to an applet.
These buttons do not do anything. That comes in a later version.
*/
public class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

        contentPane.setLayout(new FlowLayout());

        JButton sunnyButton = new JButton("Sunny");
        contentPane.add(sunnyButton);

        JButton cloudyButton = new JButton("Cloudy");
        contentPane.add(cloudyButton);
    }
}
```

Applet Output



If the user clicks either of these buttons, nothing happens.

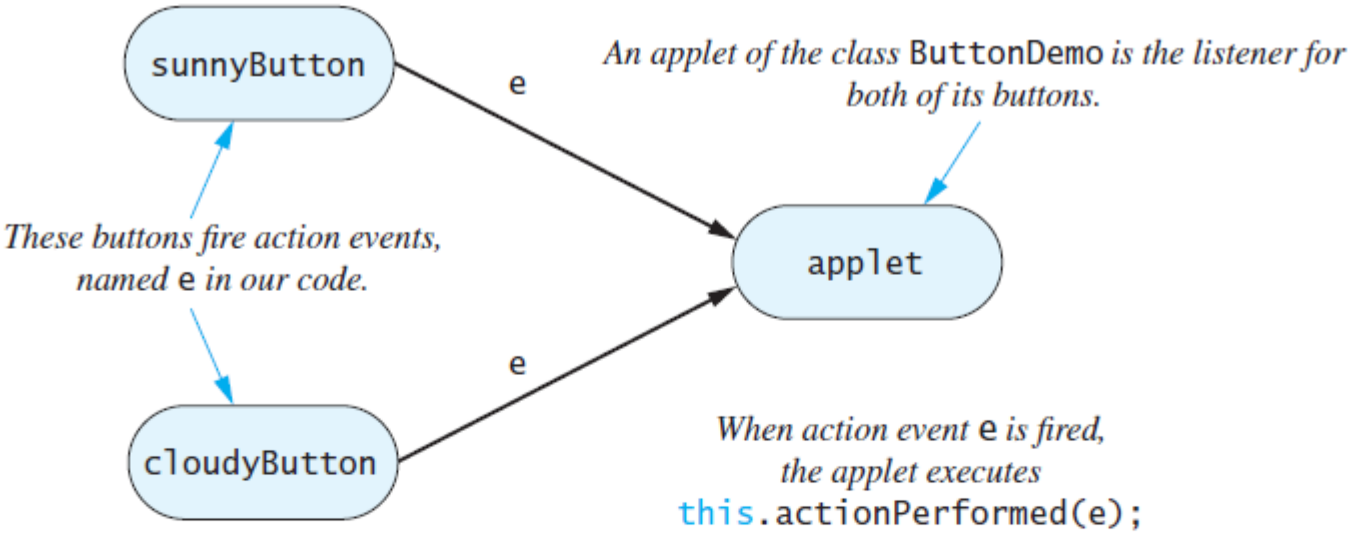
FIGURE 6.6 Event Firing and an Event Listener



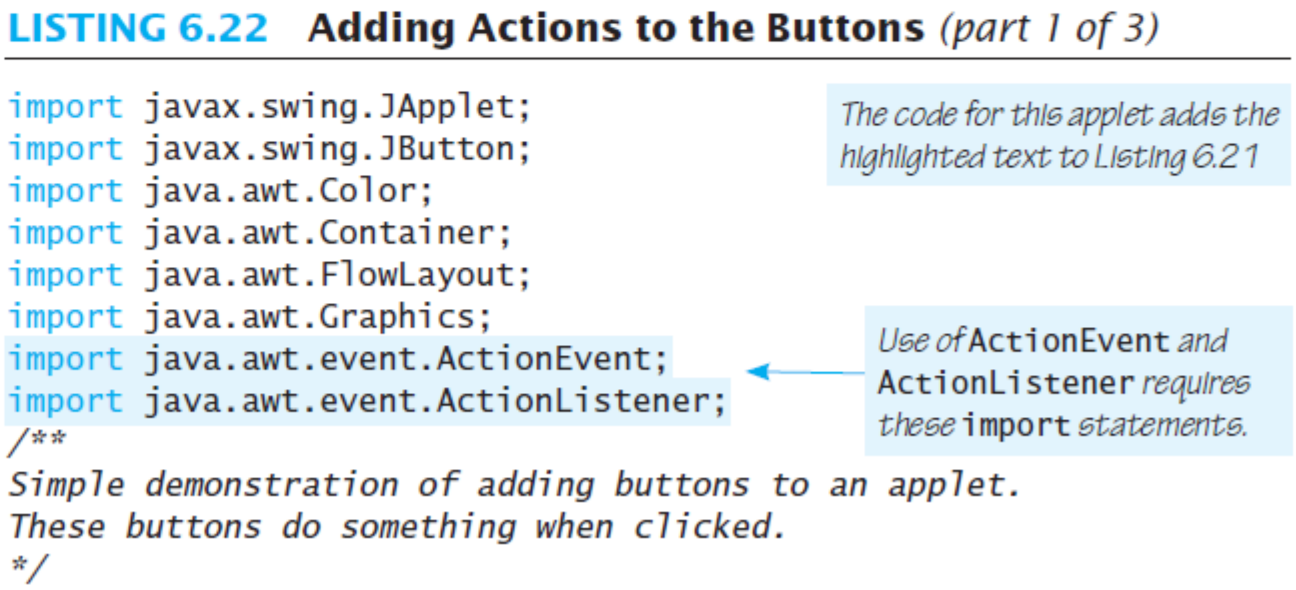
This event object is the result of a button click. The event object goes from the button to the listener.

This listener object performs some action, such as making text visible in the applet, when it receives the event object.

FIGURE 6.7 Buttons and an Action Listener


sunnyButton
e
An applet of the class ButtonDemo is the listener for both of its buttons.
These buttons fire action events, named e in our code.
applet
e
cloudyButton
When action event e is fired, the applet executes this.actionPerformed(e);

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

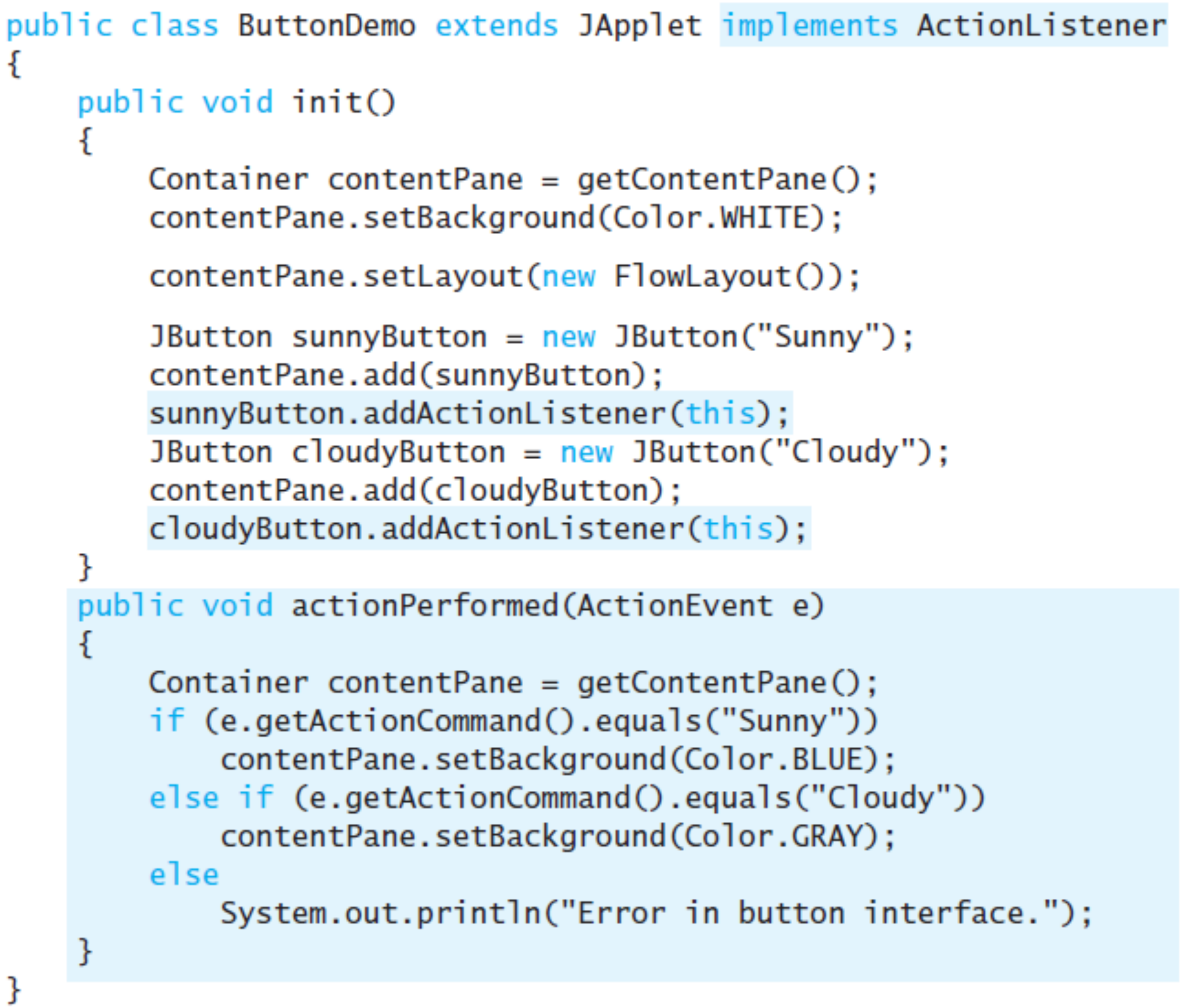


LISTING 6.22 Adding Actions to the Buttons *(part 1 of 3)*

```
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.
*/
```

The code for this applet adds the highlighted text to Listing 6.21

Use of `ActionEvent` *and* `ActionListener` *requires these* `import` *statements.*

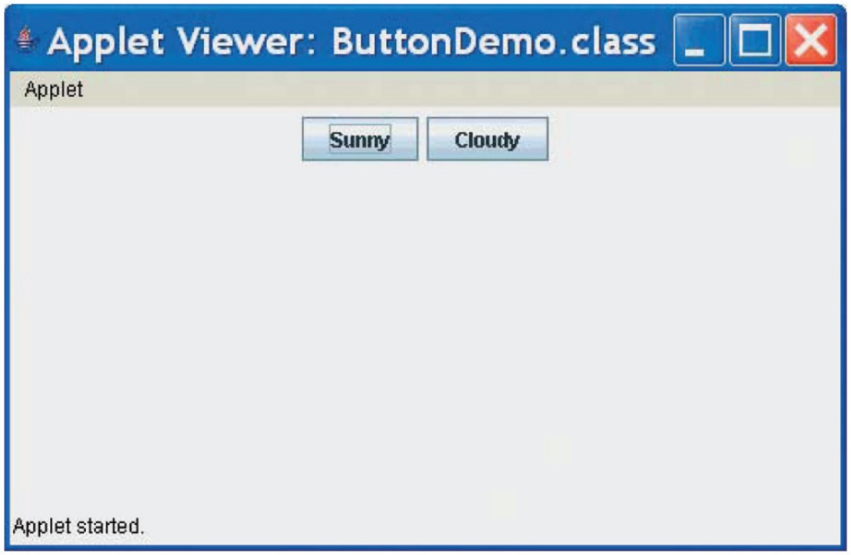


```
public class ButtonDemo extends JApplet implements ActionListener
{
    public void init()
    {
        Container contentPane = getContentPane();
        contentPane.setBackground(Color.WHITE);

        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.");
    }
}
```

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

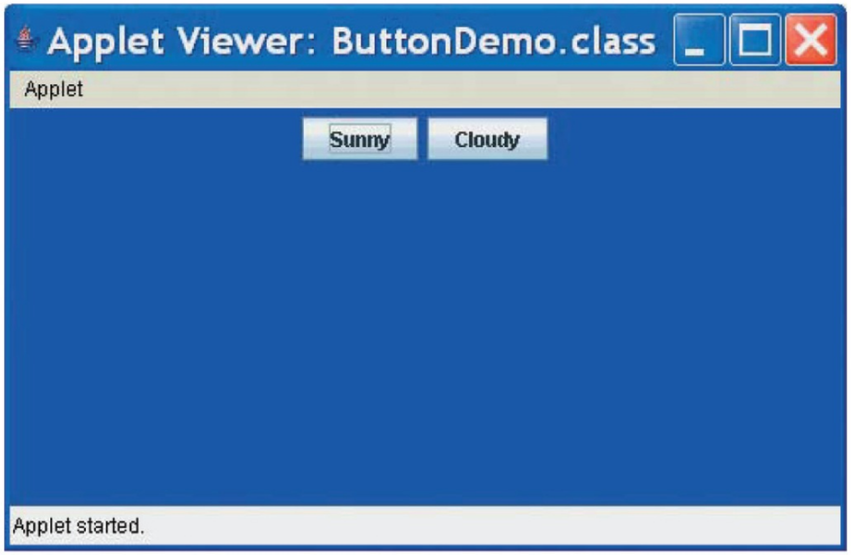
LISTING 6.22 Adding Actions to the Buttons *(Continued)*

Applet Output Initially



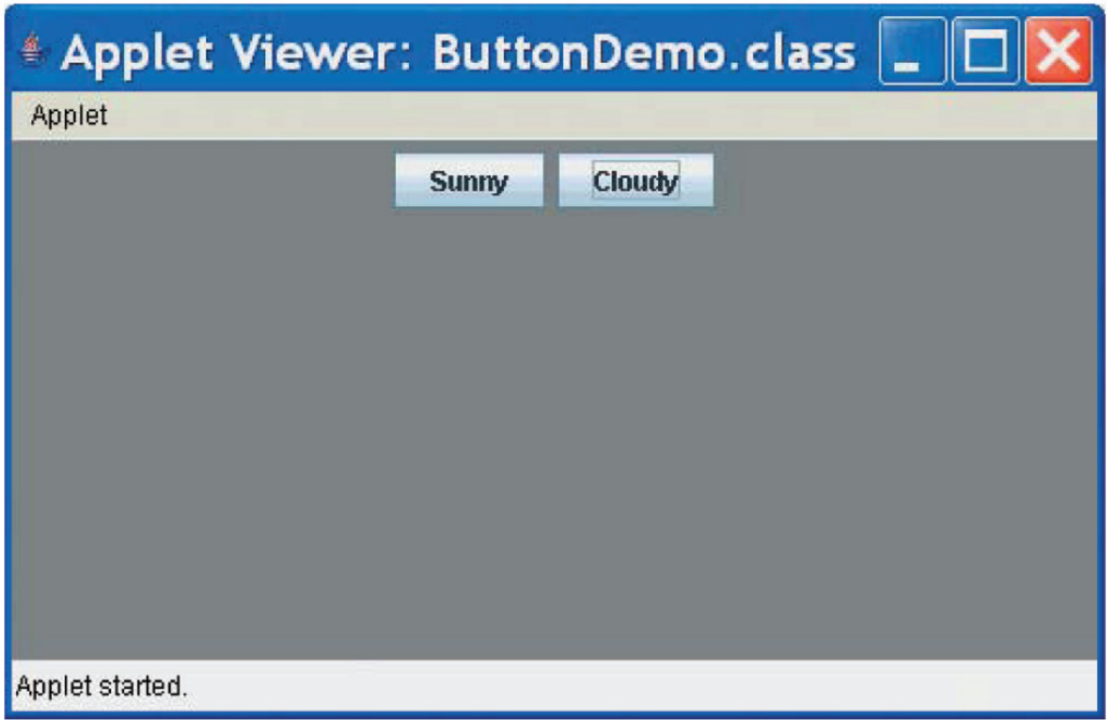
Applet started.

Applet Output After Clicking Sunny

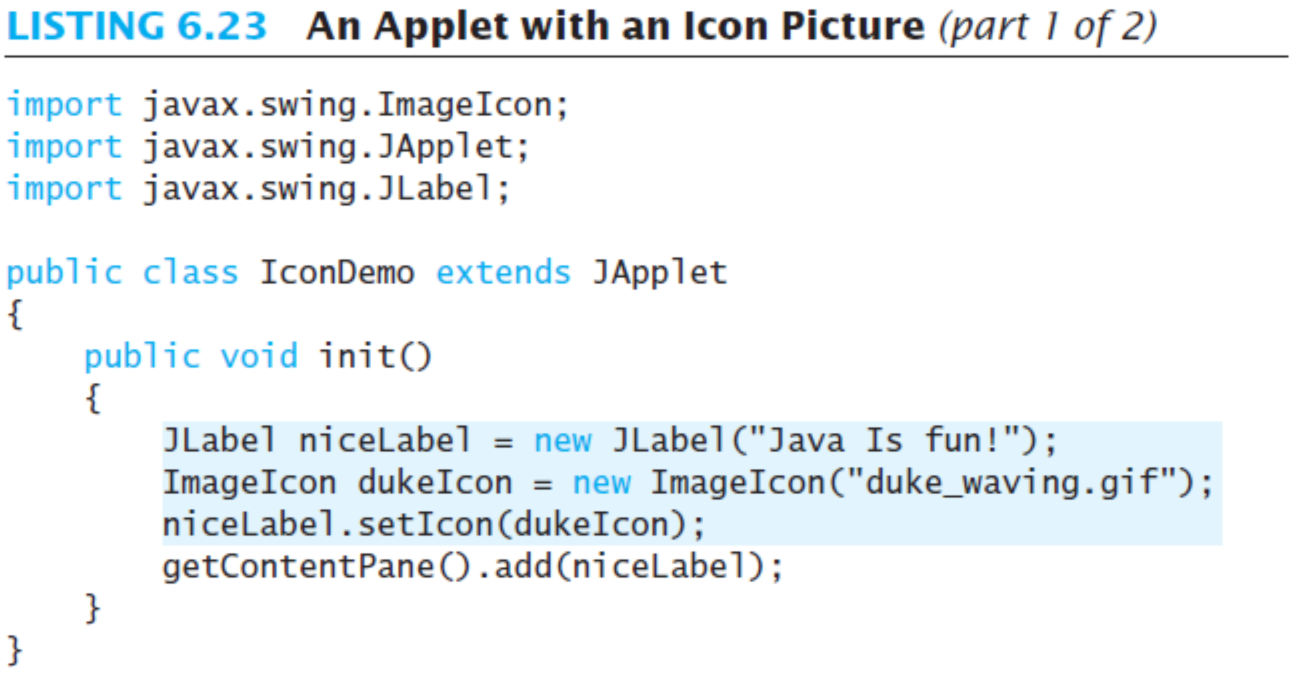


Applet started.

Applet Output After Clicking Cloudy



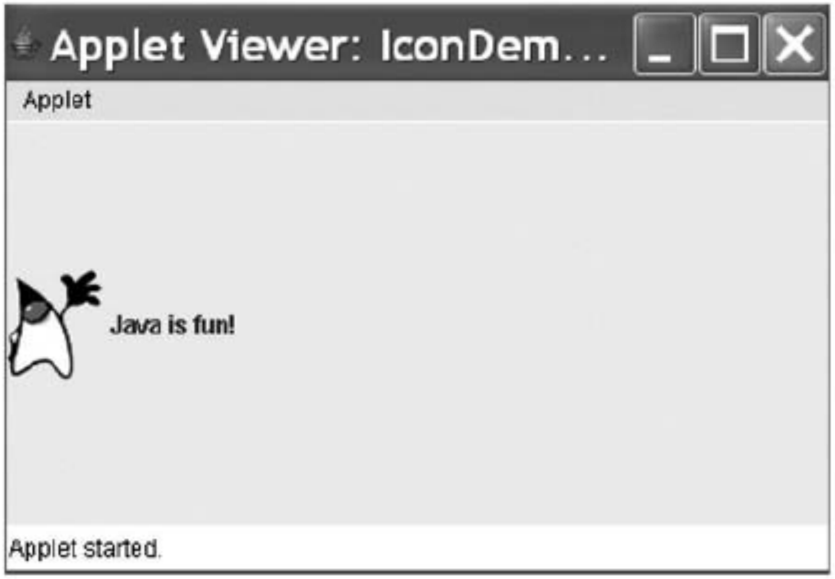
Applet started.



LISTING 6.23 **An Applet with an Icon Picture** *(part 1 of 2)*

```
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);
    }
}
```

Applet Output[4]

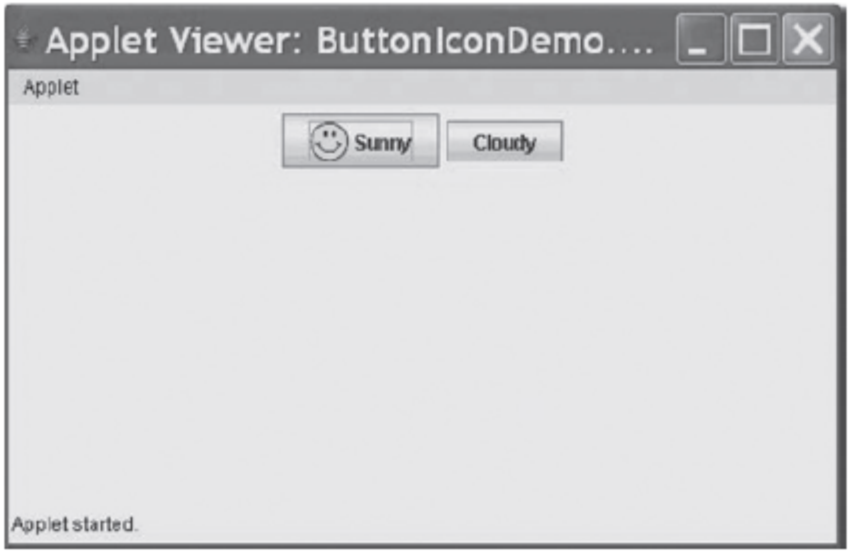


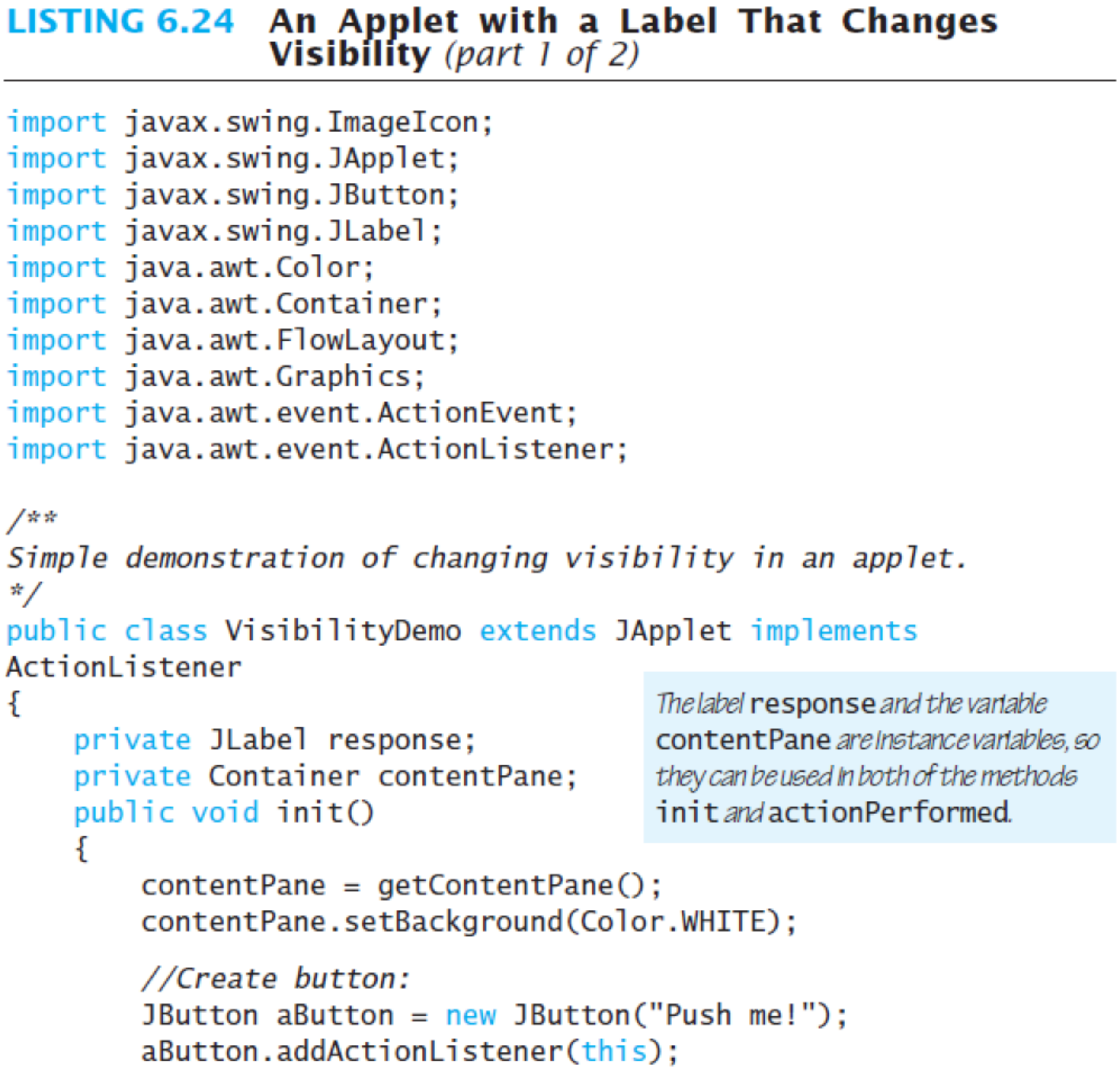
Java is fun!

Applet started.

FIGURE 6.8 A Button Containing an Icon

The code for this applet is in the file `ButtonIconDemo.java` *in the source code on the Web.*



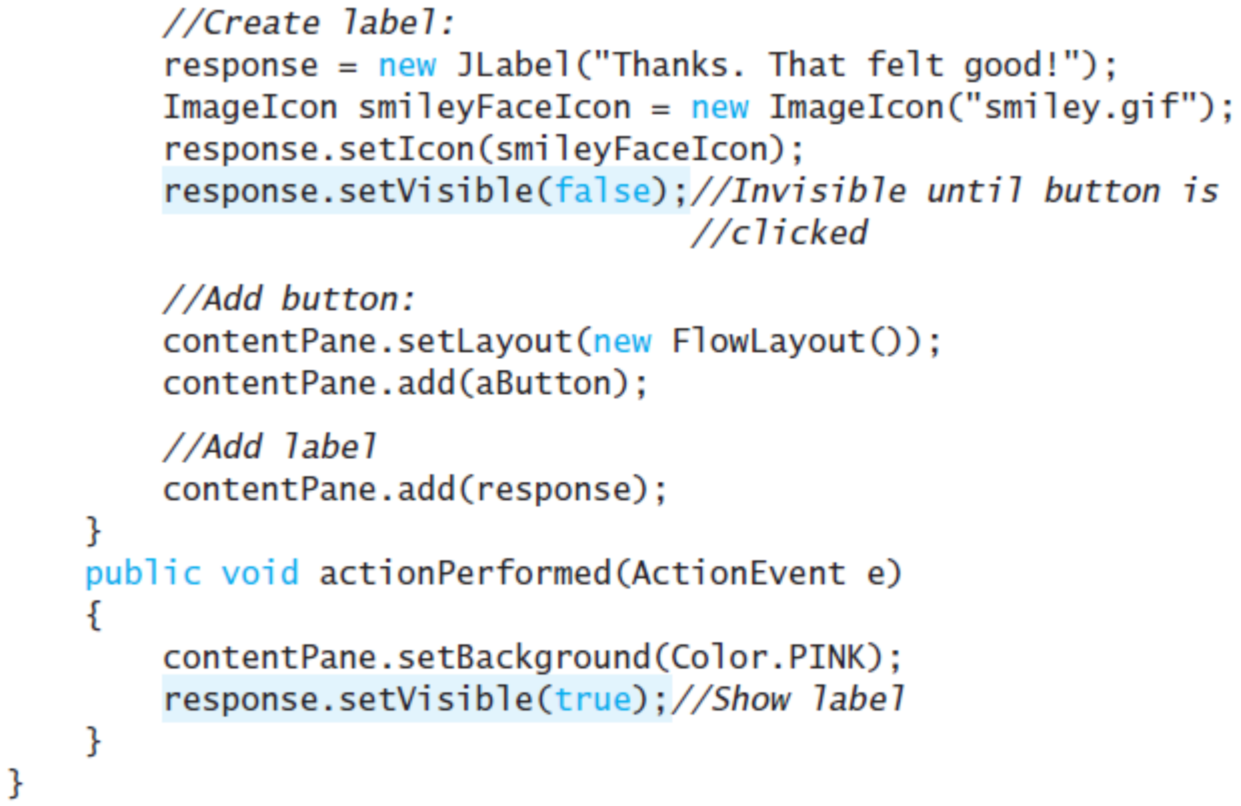


LISTING 6.24 An Applet with a Label That Changes Visibility *(part 1 of 2)*

```java
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);
```

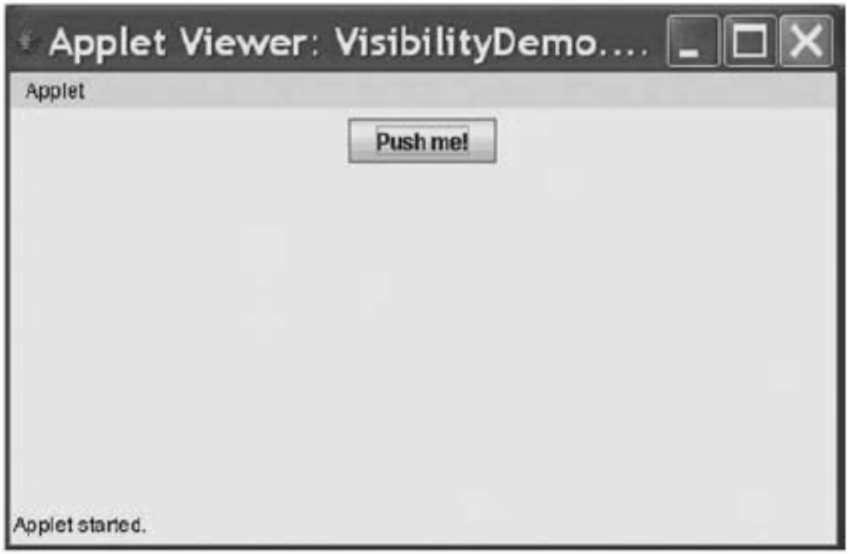
The label `response` *and the variable* `contentPane` *are instance variables, so they can be used in both of the methods* `init` *and* `actionPerformed`*.*

JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

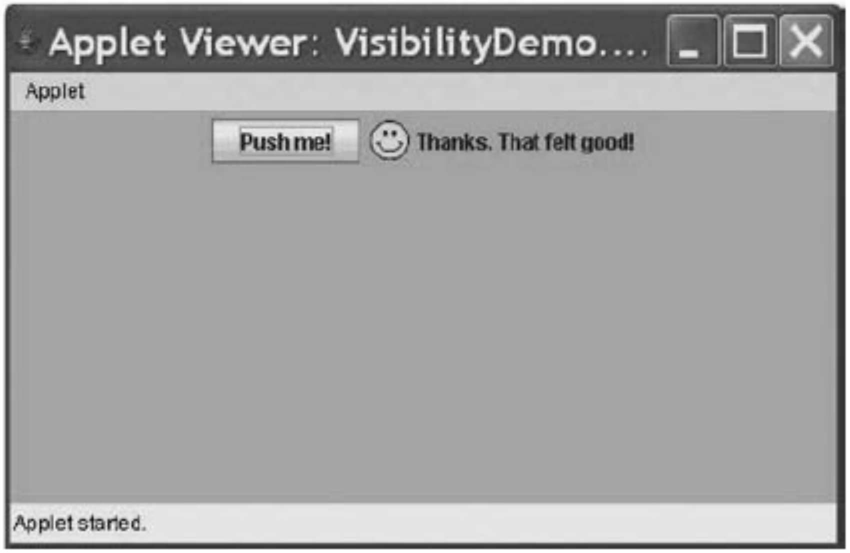


```
        //Create label:
        response = new JLabel("Thanks. That felt good!");
        ImageIcon smileyFaceIcon = new ImageIcon("smiley.gif");
        response.setIcon(smileyFaceIcon);
        response.setVisible(false);//Invisible until button is
                                   //clicked

        //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
    }
}
```

Applet Output Initially



Applet Output After Clicking Button



JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved