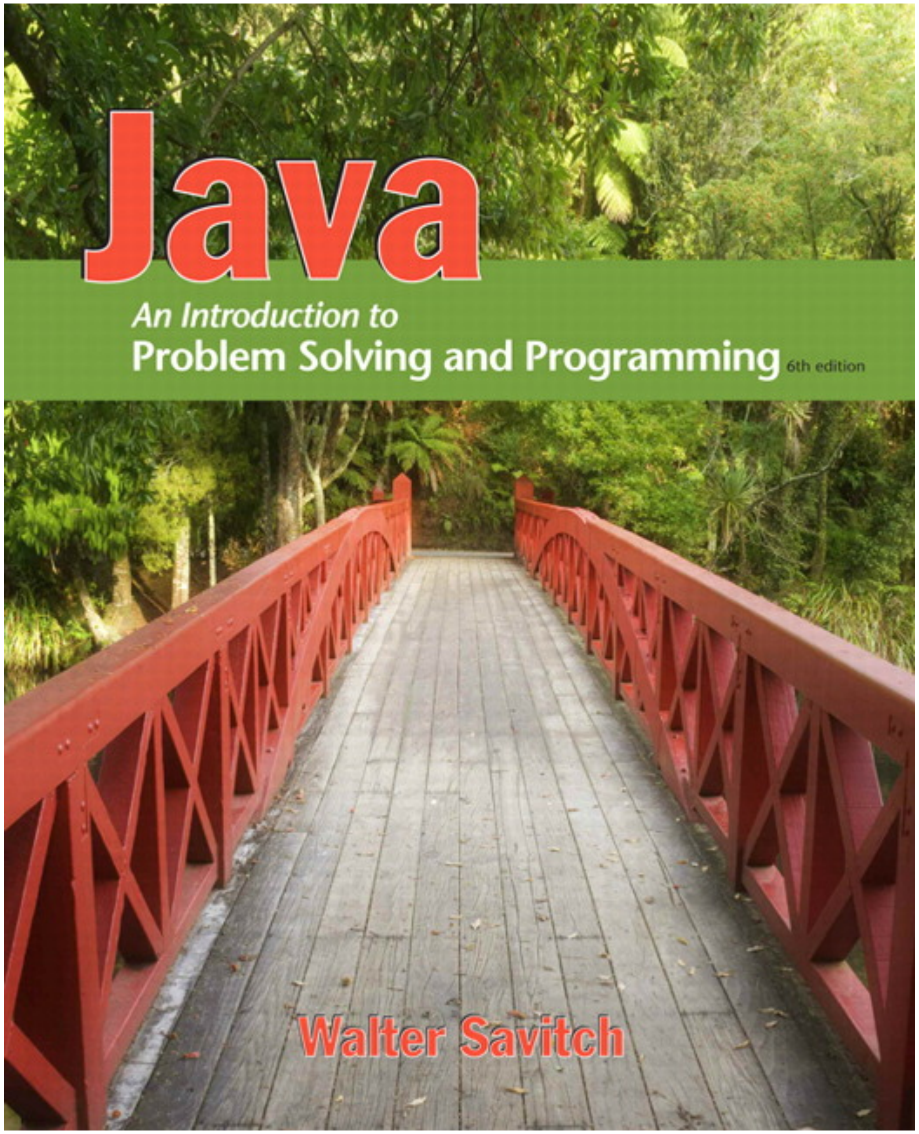
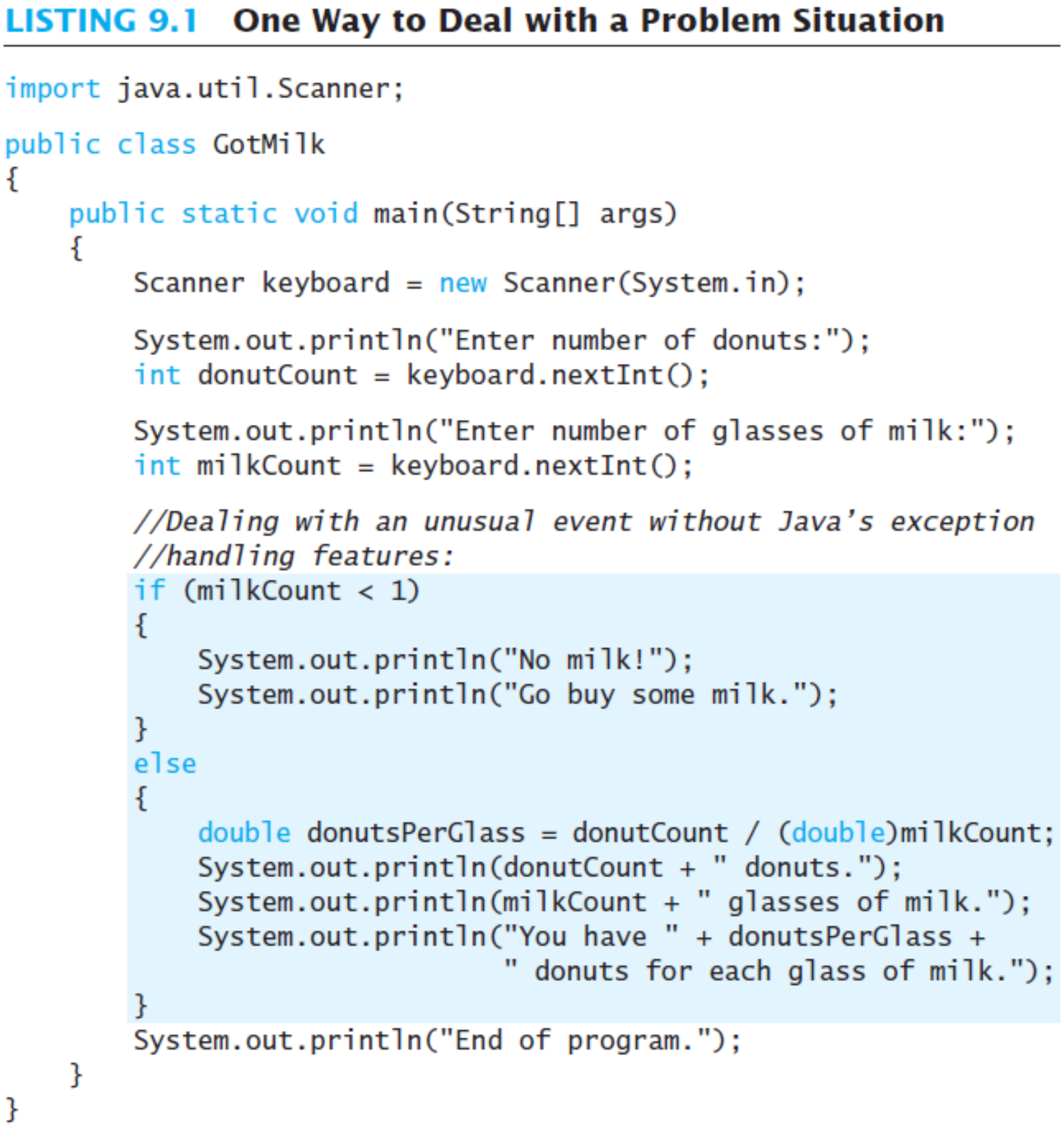


Exception Handling 9

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

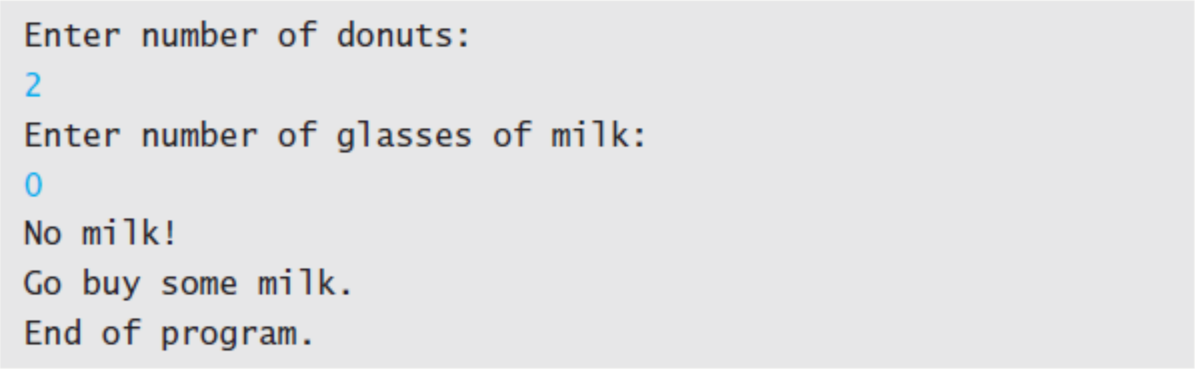


LISTING 9.1 One Way to Deal with a Problem Situation

```
import java.util.Scanner;

public class GotMilk
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);

        System.out.println("Enter number of donuts:");
        int donutCount = keyboard.nextInt();

        System.out.println("Enter number of glasses of milk:");
        int milkCount = keyboard.nextInt();

        //Dealing with an unusual event without Java's exception
        //handling features:
        if (milkCount < 1)
        {
            System.out.println("No milk!");
            System.out.println("Go buy some milk.");
        }
        else
        {
            double donutsPerGlass = donutCount / (double)milkCount;
            System.out.println(donutCount + " donuts.");
            System.out.println(milkCount + " glasses of milk.");
            System.out.println("You have " + donutsPerGlass +
                               " donuts for each glass of milk.");
        }
        System.out.println("End of program.");
    }
}
```

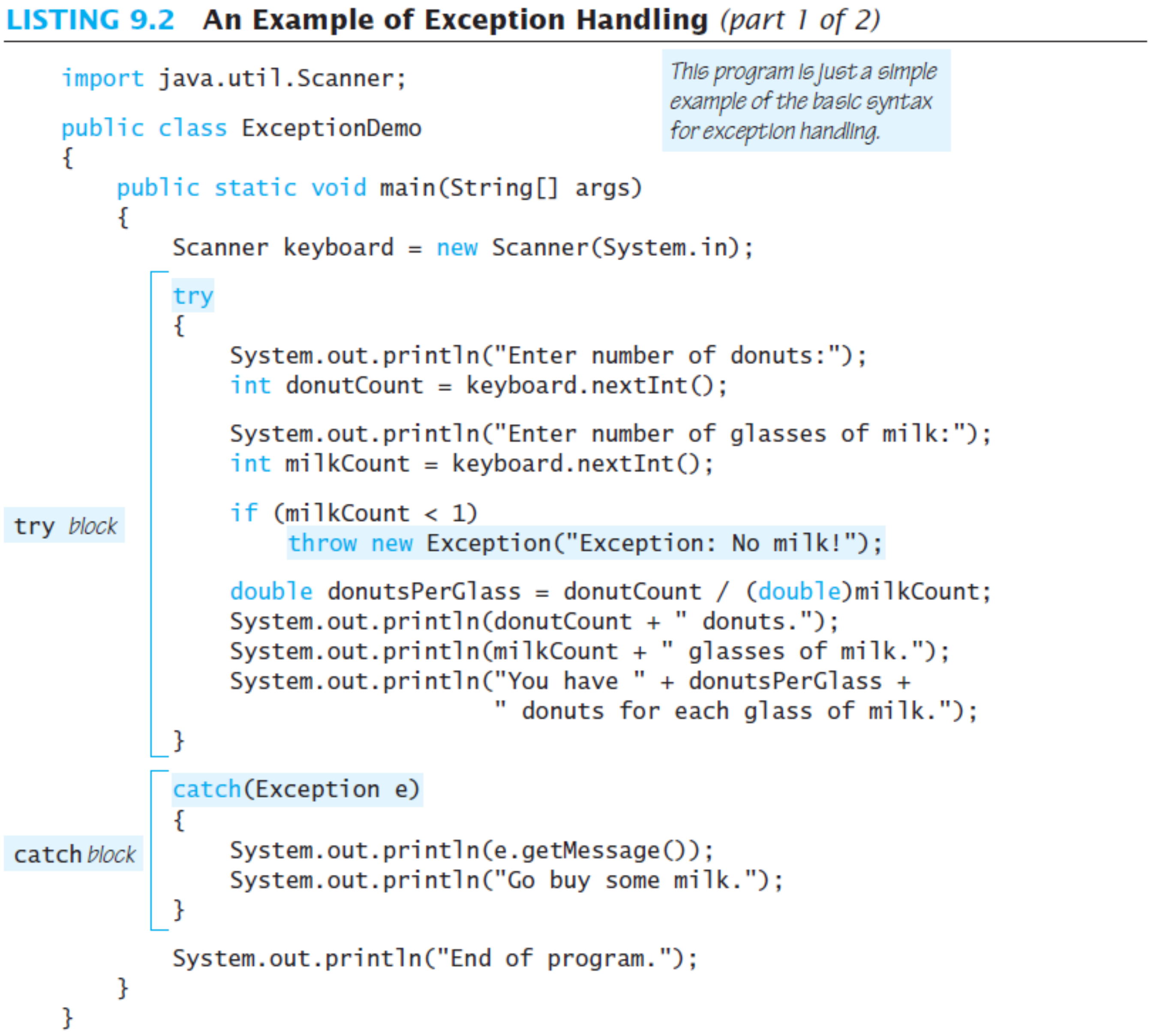
JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch
ISBN 0132162709 © 2012 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Sample Screen Output



```
Enter number of donuts:
2
Enter number of glasses of milk:
0
No milk!
Go buy some milk.
End of program.
```

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



LISTING 9.2 An Example of Exception Handling *(part 1 of 2)*

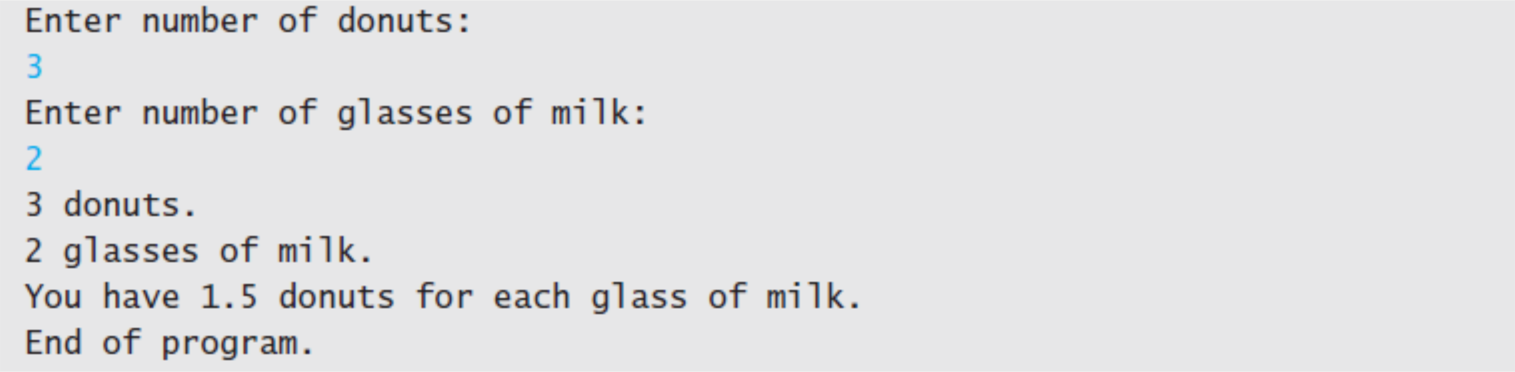
```
import java.util.Scanner;

public class ExceptionDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        try
        {
            System.out.println("Enter number of donuts:");
            int donutCount = keyboard.nextInt();

            System.out.println("Enter number of glasses of milk:");
            int milkCount = keyboard.nextInt();

            if (milkCount < 1)
                throw new Exception("Exception: No milk!");

            double donutsPerGlass = donutCount / (double)milkCount;
            System.out.println(donutCount + " donuts.");
            System.out.println(milkCount + " glasses of milk.");
            System.out.println("You have " + donutsPerGlass +
                               " donuts for each glass of milk.");
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
            System.out.println("Go buy some milk.");
        }
        System.out.println("End of program.");
    }
}
```

This program is just a simple example of the basic syntax for exception handling.

`try` *block*

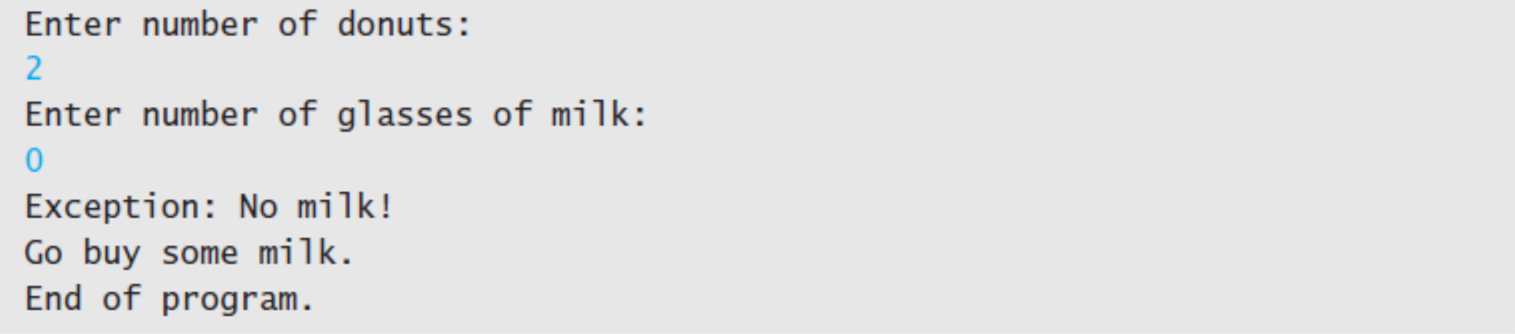
`catch` *block*

Sample Screen Output 1



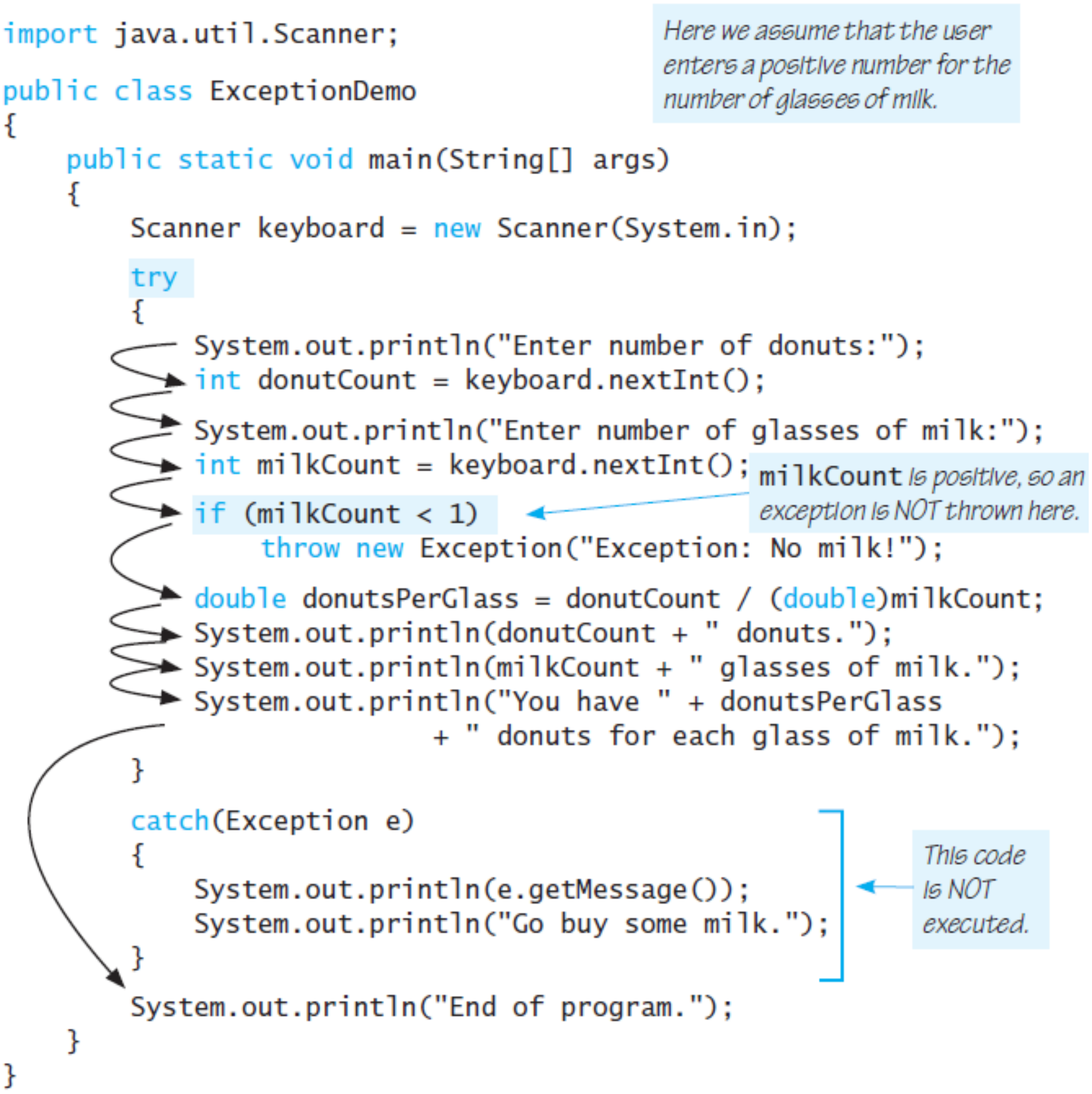
```
Enter number of donuts:
3
Enter number of glasses of milk:
2
3 donuts.
2 glasses of milk.
You have 1.5 donuts for each glass of milk.
End of program.
```

Sample Screen Output 1



```
Enter number of donuts:
2
Enter number of glasses of milk:
0
Exception: No milk!
Go buy some milk.
End of program.
```

LISTING 9.3 Flow of Control When No Exception Is Thrown



```
import java.util.Scanner;

public class ExceptionDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);

        try
        {
            System.out.println("Enter number of donuts:");
            int donutCount = keyboard.nextInt();

            System.out.println("Enter number of glasses of milk:");
            int milkCount = keyboard.nextInt();

            if (milkCount < 1)
                throw new Exception("Exception: No milk!");

            double donutsPerGlass = donutCount / (double)milkCount;
            System.out.println(donutCount + " donuts.");
            System.out.println(milkCount + " glasses of milk.");
            System.out.println("You have " + donutsPerGlass
                              + " donuts for each glass of milk.");
        }

        catch(Exception e)
        {
            System.out.println(e.getMessage());
            System.out.println("Go buy some milk.");
        }
        System.out.println("End of program.");
    }
}
```

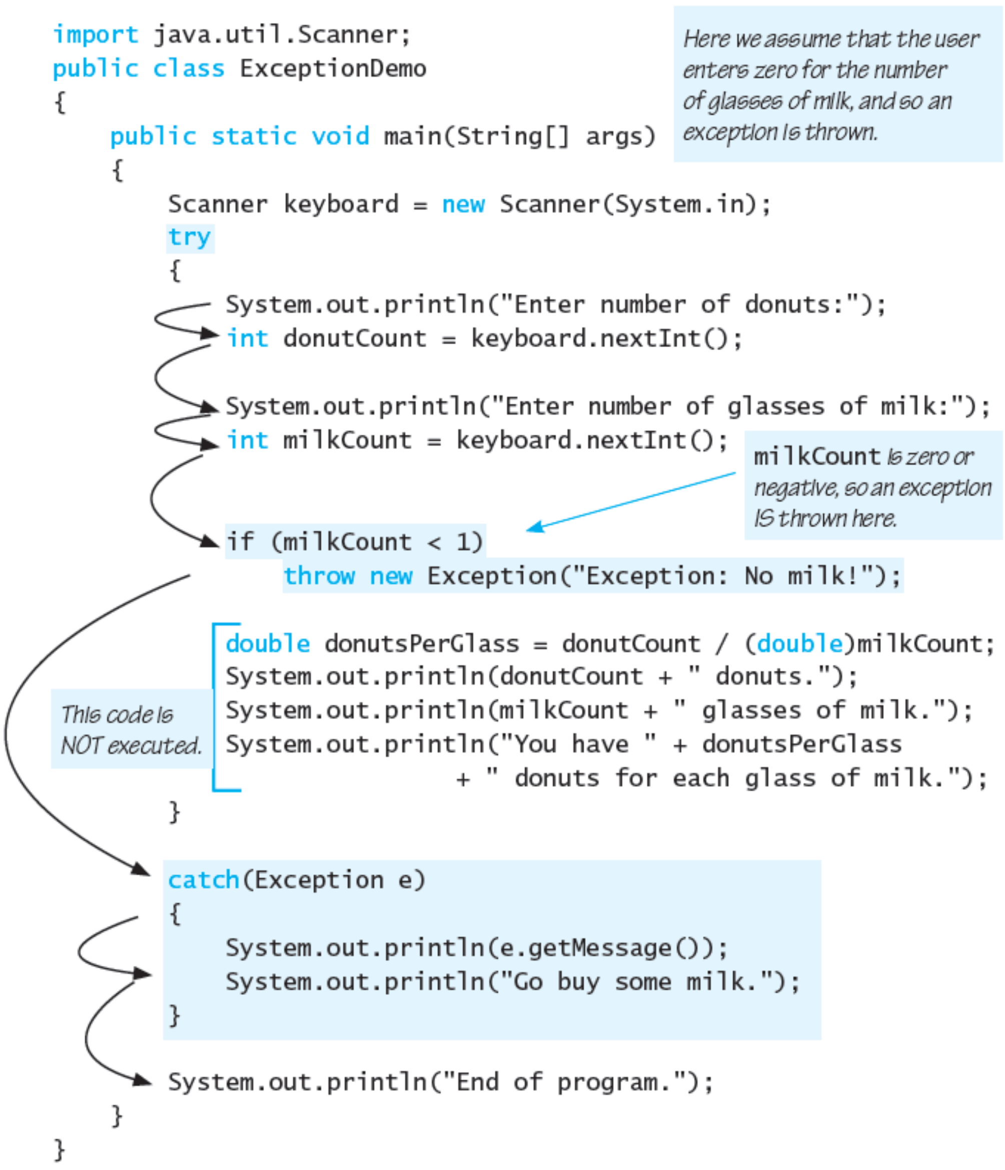
Here we assume that the user enters a positive number for the number of glasses of milk.

`milkCount` *is positive, so an exception is NOT thrown here.*

This code is NOT executed.

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 9.4 Flow of Control When an Exception Is Thrown



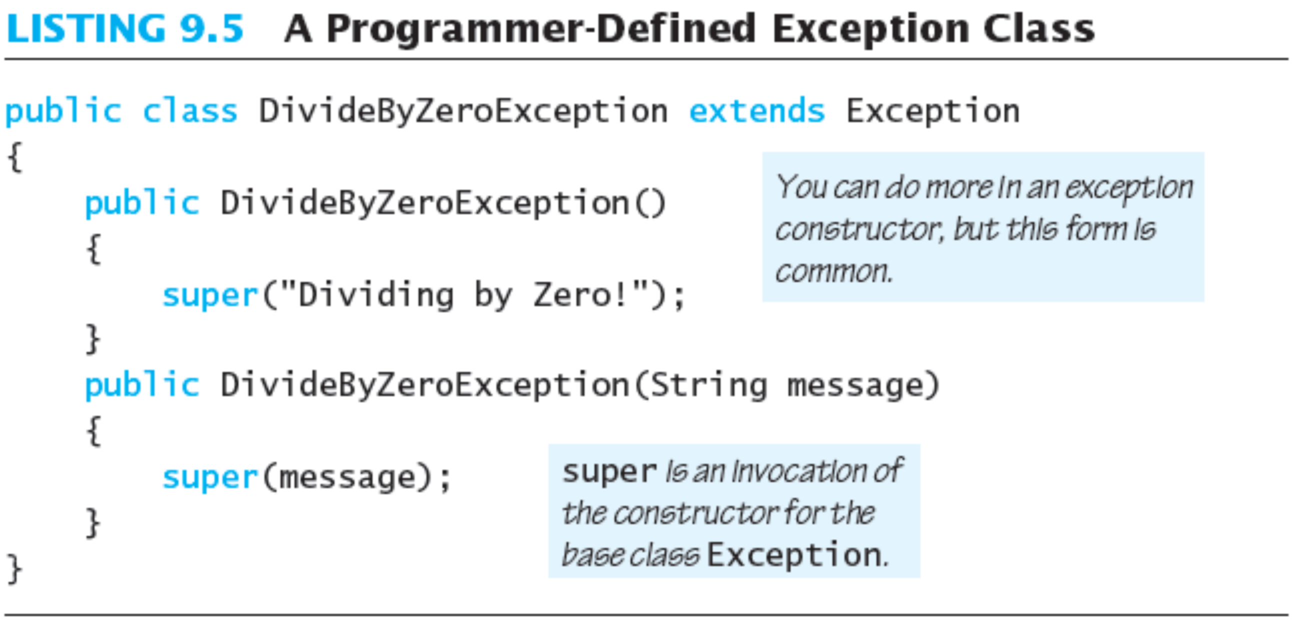
```
import java.util.Scanner;
public class ExceptionDemo
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        try
        {
            System.out.println("Enter number of donuts:");
            int donutCount = keyboard.nextInt();

            System.out.println("Enter number of glasses of milk:");
            int milkCount = keyboard.nextInt();

            if (milkCount < 1)
                throw new Exception("Exception: No milk!");

            double donutsPerGlass = donutCount / (double)milkCount;
            System.out.println(donutCount + " donuts.");
            System.out.println(milkCount + " glasses of milk.");
            System.out.println("You have " + donutsPerGlass
                                + " donuts for each glass of milk.");
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
            System.out.println("Go buy some milk.");
        }
        System.out.println("End of program.");
    }
}
```

Here we assume that the user enters zero for the number of glasses of milk, and so an exception is thrown.

`milkCount` *is zero or negative, so an exception IS thrown here.*

This code is NOT executed.

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

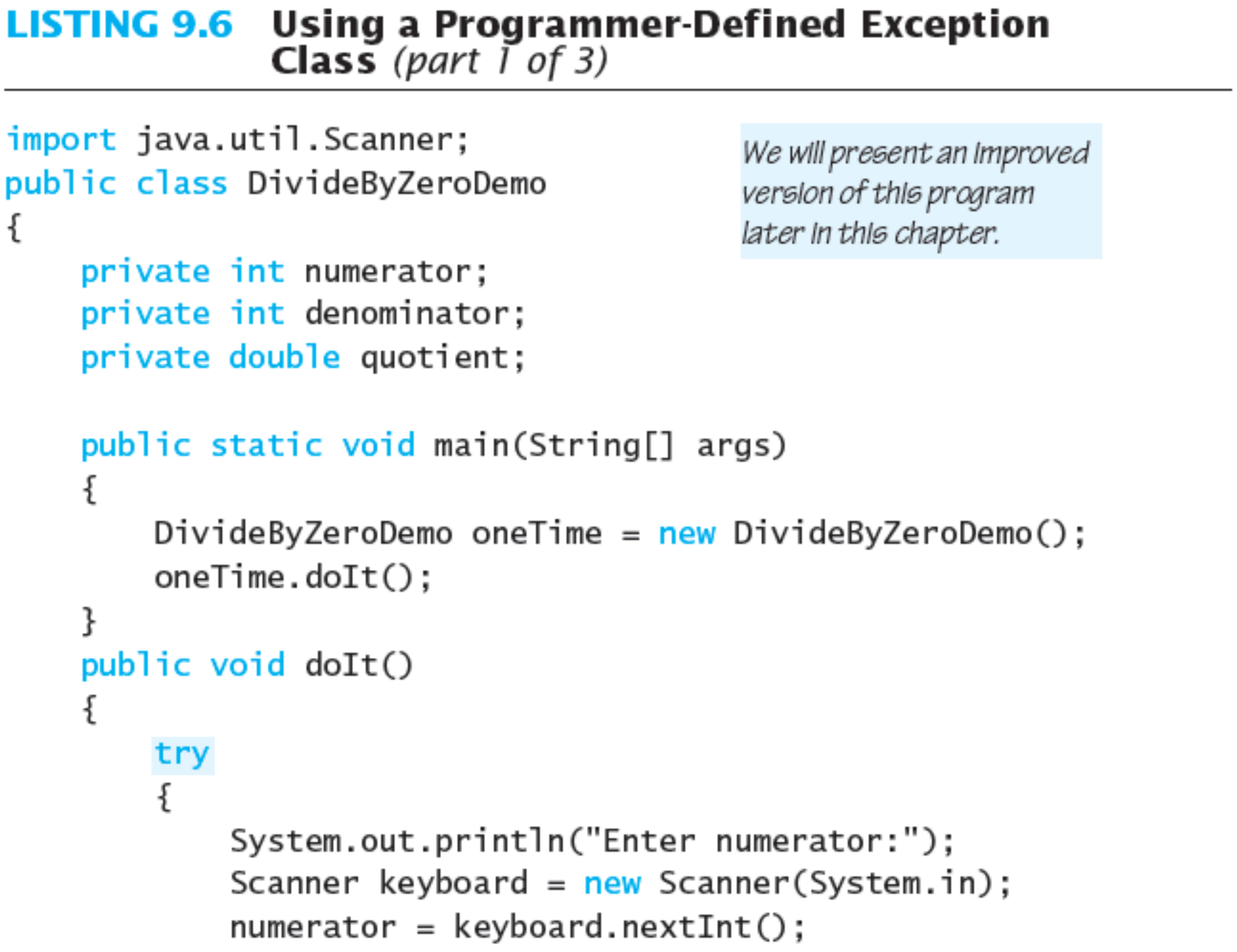


LISTING 9.5 A Programmer-Defined Exception Class

```
public class DivideByZeroException extends Exception
{
    public DivideByZeroException()
    {
        super("Dividing by Zero!");
    }
    public DivideByZeroException(String message)
    {
        super(message);
    }
}
```

You can do more in an exception constructor, but this form is common.

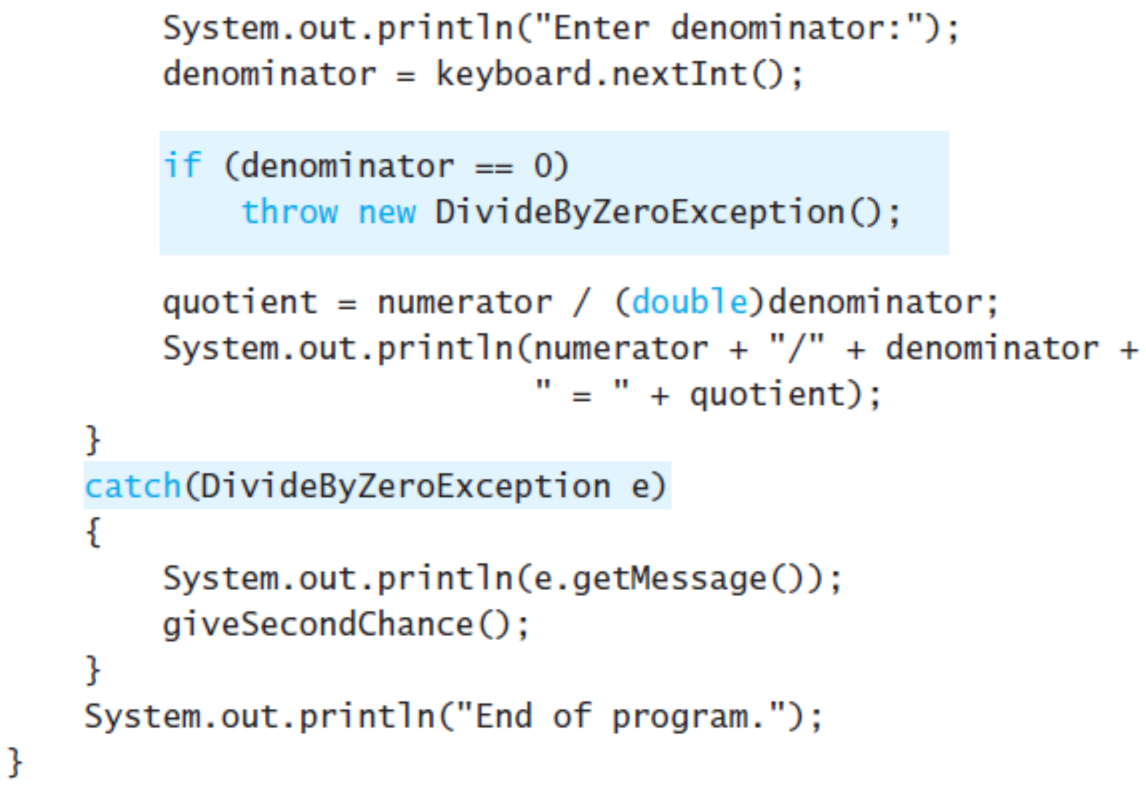
super is an invocation of the constructor for the base class Exception.



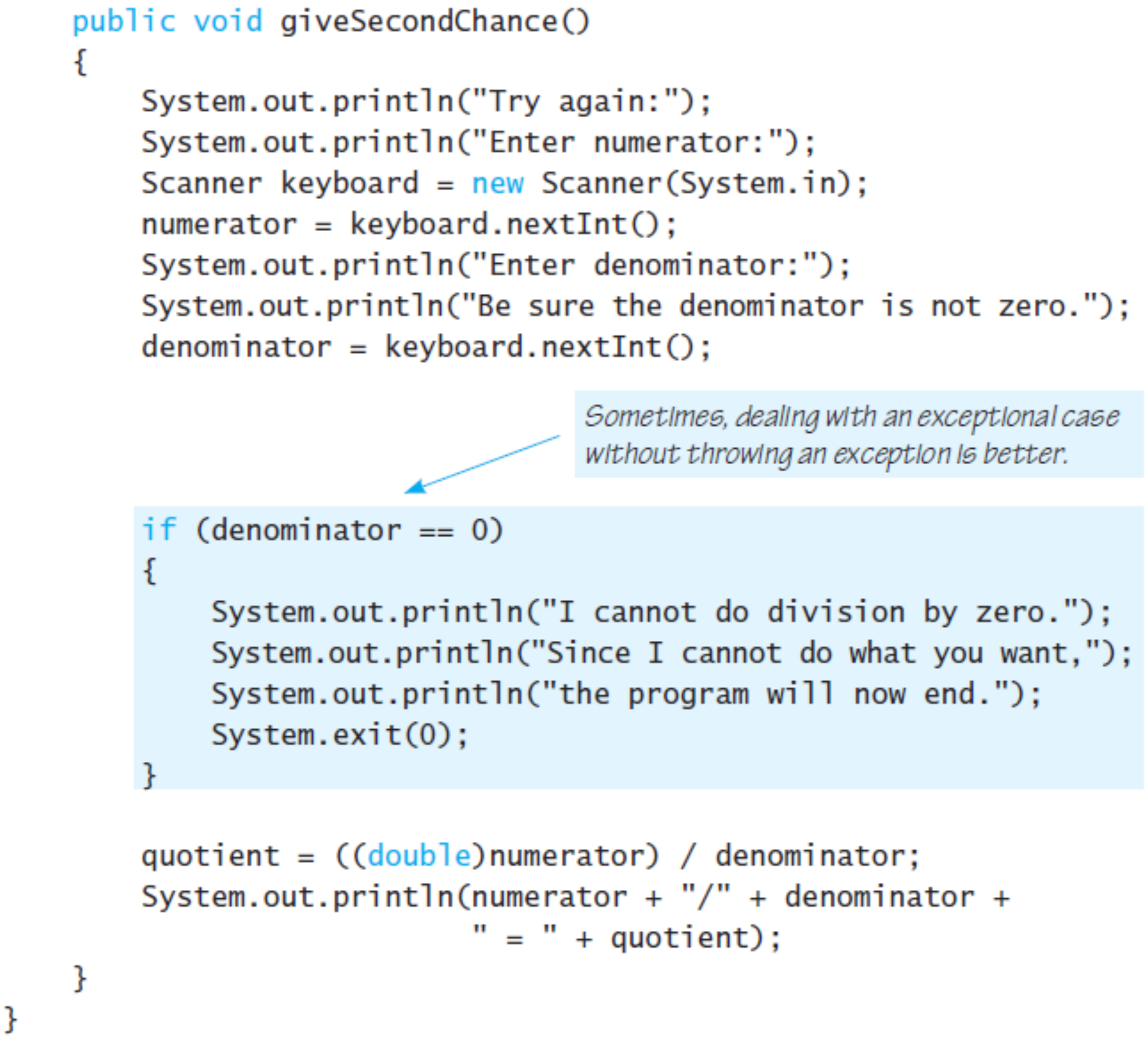
LISTING 9.6 **Using a Programmer-Defined Exception Class** *(part 1 of 3)*

```
import java.util.Scanner;
public class DivideByZeroDemo
{
    private int numerator;
    private int denominator;
    private double quotient;

    public static void main(String[] args)
    {
        DivideByZeroDemo oneTime = new DivideByZeroDemo();
        oneTime.doIt();
    }
    public void doIt()
    {
        try
        {
            System.out.println("Enter numerator:");
            Scanner keyboard = new Scanner(System.in);
            numerator = keyboard.nextInt();
```

We will present an improved version of this program later in this chapter.



```
        System.out.println("Enter denominator:");
        denominator = keyboard.nextInt();

        if (denominator == 0)
            throw new DivideByZeroException();

        quotient = numerator / (double)denominator;
        System.out.println(numerator + "/" + denominator +
                           " = " + quotient);
    }
    catch(DivideByZeroException e)
    {
        System.out.println(e.getMessage());
        giveSecondChance();
    }
    System.out.println("End of program.");
}
```



```java
    public void giveSecondChance()
    {
        System.out.println("Try again:");
        System.out.println("Enter numerator:");
        Scanner keyboard = new Scanner(System.in);
        numerator = keyboard.nextInt();
        System.out.println("Enter denominator:");
        System.out.println("Be sure the denominator is not zero.");
        denominator = keyboard.nextInt();

        if (denominator == 0)
        {
            System.out.println("I cannot do division by zero.");
            System.out.println("Since I cannot do what you want,");
            System.out.println("the program will now end.");
            System.exit(0);
        }

        quotient = ((double)numerator) / denominator;
        System.out.println(numerator + "/" + denominator +
                           " = " + quotient);
    }
}
```

Sometimes, dealing with an exceptional case without throwing an exception is better.

Sample Screen Output 1

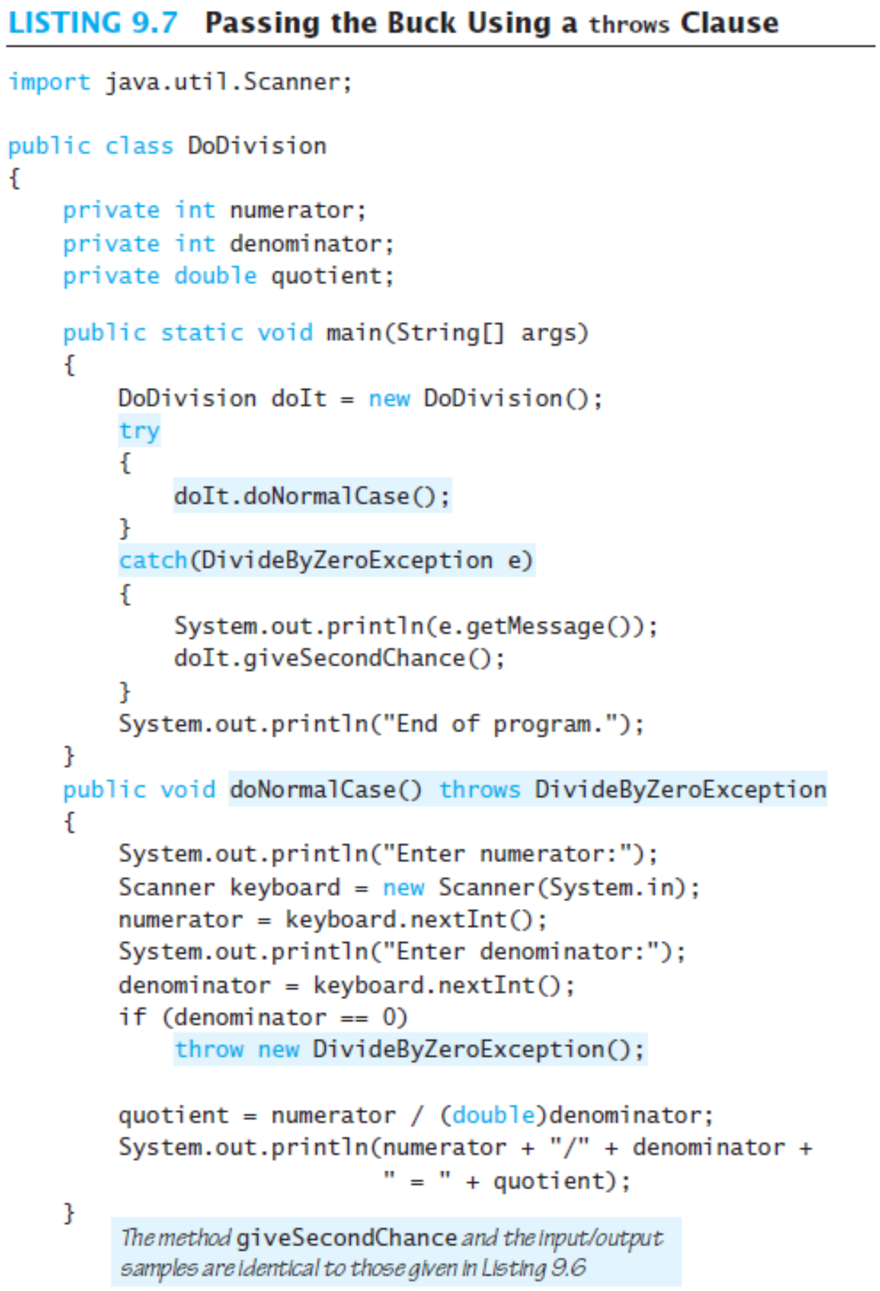
```
Enter numerator:
5
Enter denominator:
10
5/10 = 0.5
End of program.
```

Sample Screen Output 2

```
Enter numerator:
5
Enter denominator:
0
Dividing by Zero!
Try again.
Enter numerator:
5
Enter denominator:
Be sure the denominator is not zero.
10
5/10 = 0.5
End of program.
```

Sample Screen Output 3

```
Enter numerator:
5
Enter denominator:
0
Dividing by Zero!
Try again.
Enter numerator:
5
Enter denominator:
Be sure the denominator is not zero.
0
I cannot do division by zero.
Since I cannot do what you want,
the program will now end.
```

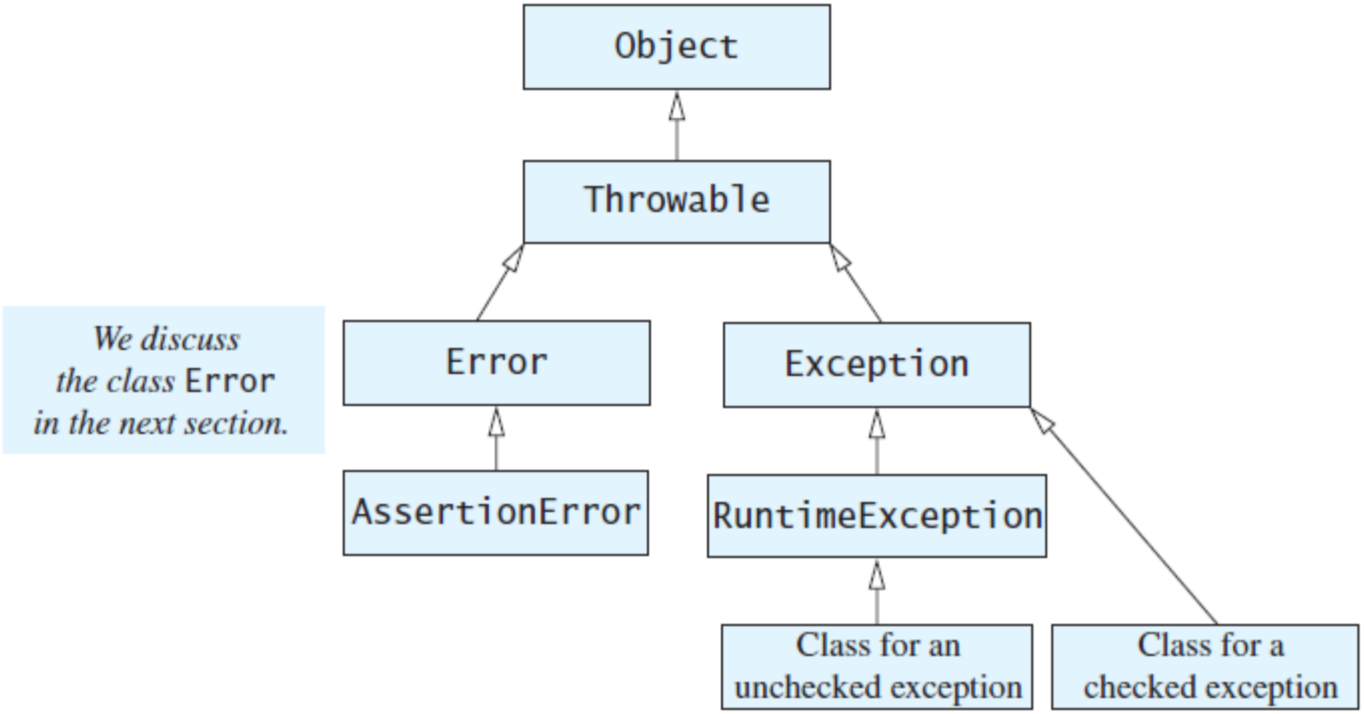


LISTING 9.7 Passing the Buck Using a `throws` Clause

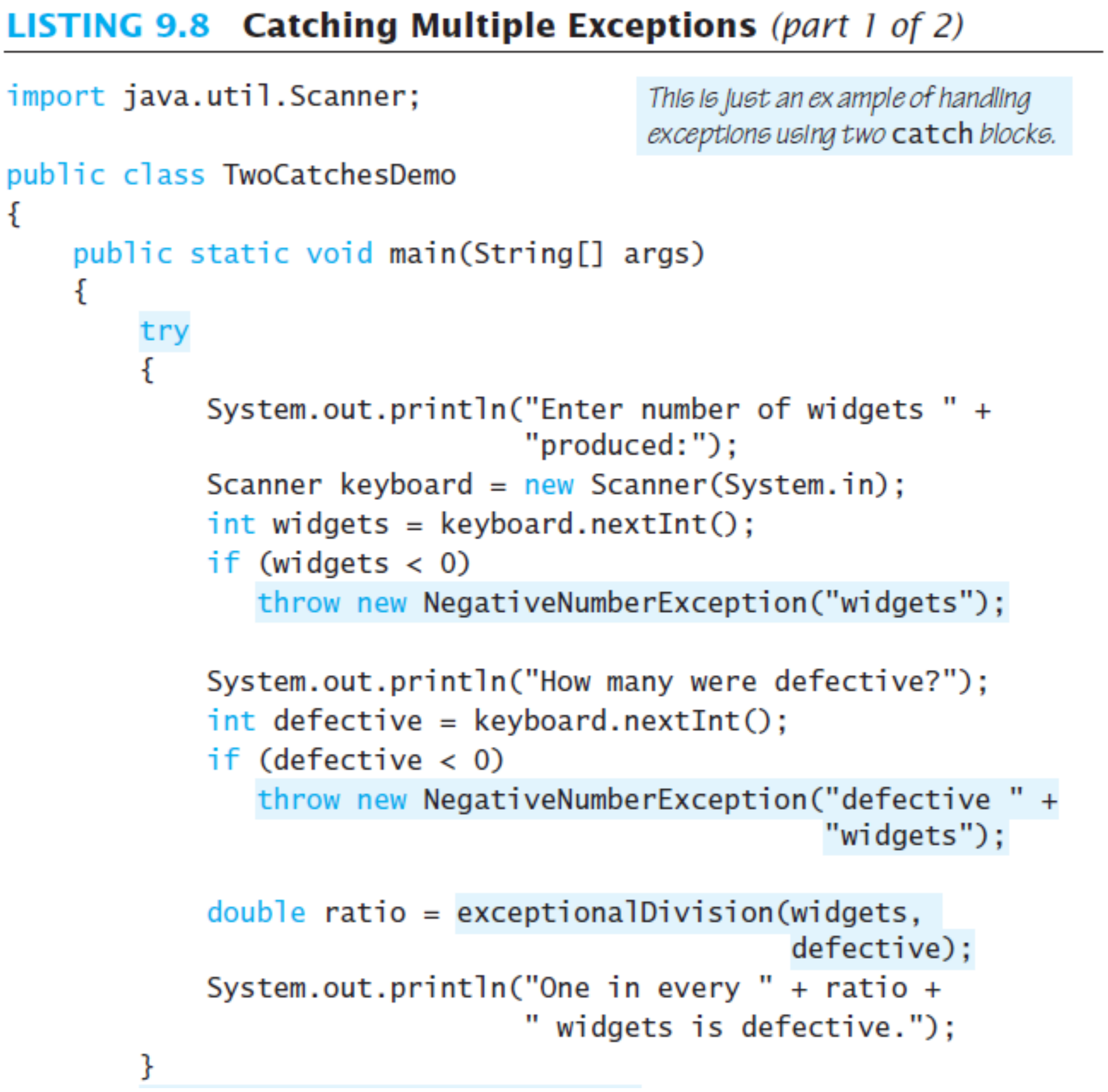
```
import java.util.Scanner;

public class DoDivision
{
    private int numerator;
    private int denominator;
    private double quotient;

    public static void main(String[] args)
    {
        DoDivision doIt = new DoDivision();
        try
        {
            doIt.doNormalCase();
        }
        catch(DivideByZeroException e)
        {
            System.out.println(e.getMessage());
            doIt.giveSecondChance();
        }
        System.out.println("End of program.");
    }
    public void doNormalCase() throws DivideByZeroException
    {
        System.out.println("Enter numerator:");
        Scanner keyboard = new Scanner(System.in);
        numerator = keyboard.nextInt();
        System.out.println("Enter denominator:");
        denominator = keyboard.nextInt();
        if (denominator == 0)
            throw new DivideByZeroException();

        quotient = numerator / (double)denominator;
        System.out.println(numerator + "/" + denominator +
                           " = " + quotient);
    }
```

The method **giveSecondChance** *and the input/output samples are identical to those given in Listing 9.6*

FIGURE 9.1 Hierarchy of the Predefined Exception Classes


Object
Throwable
We discuss the class Error in the next section.
Error
Exception
AssertionError
RuntimeException
Class for an unchecked exception
Class for a checked exception

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

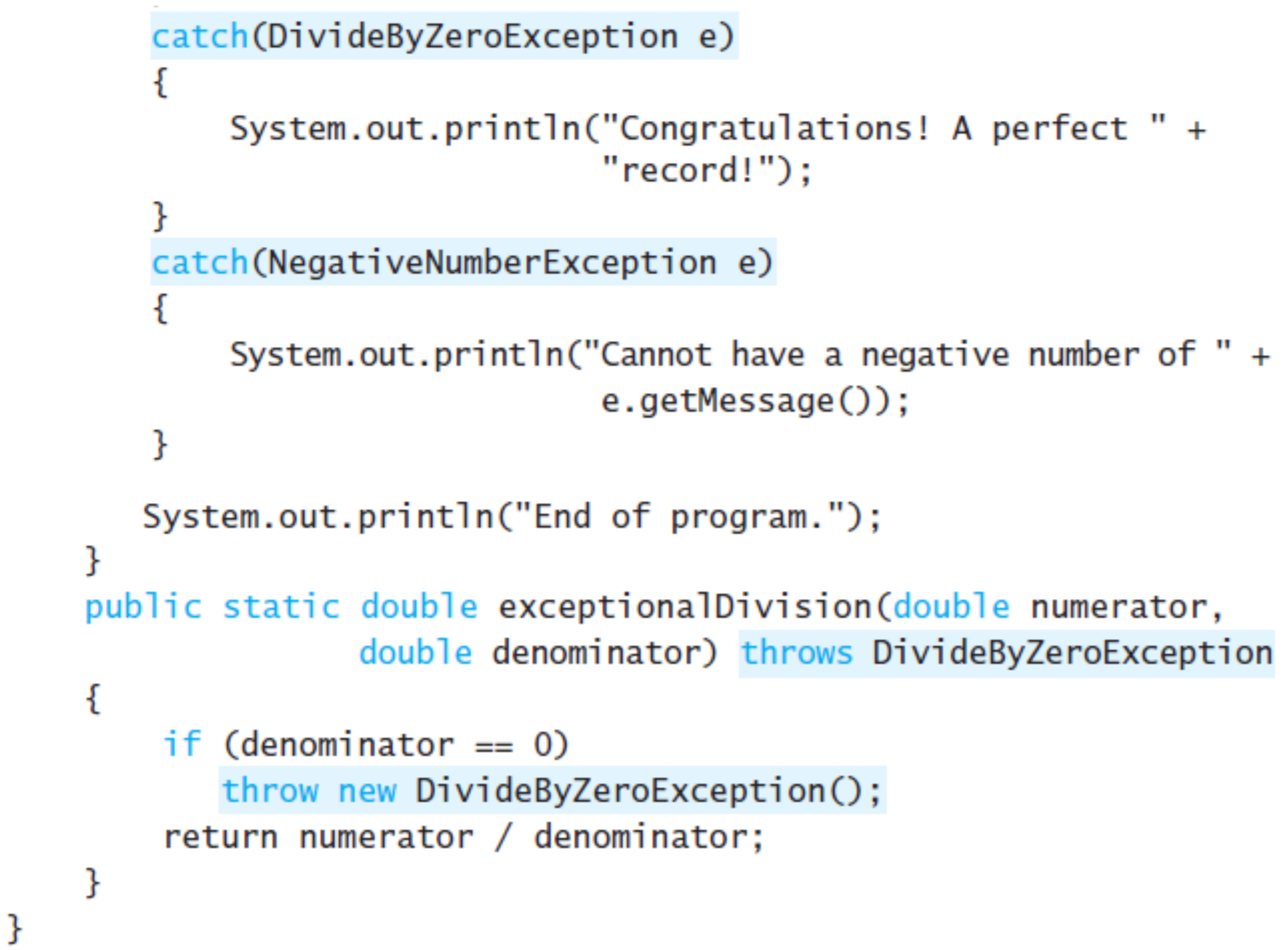


LISTING 9.8 Catching Multiple Exceptions *(part 1 of 2)*

This is just an example of handling exceptions using two **catch** *blocks.*

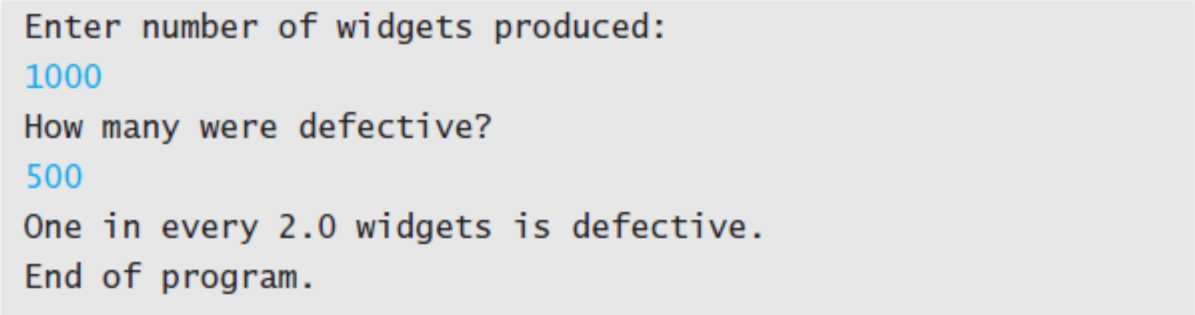
```
import java.util.Scanner;

public class TwoCatchesDemo
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("Enter number of widgets " +
                               "produced:");
            Scanner keyboard = new Scanner(System.in);
            int widgets = keyboard.nextInt();
            if (widgets < 0)
               throw new NegativeNumberException("widgets");

            System.out.println("How many were defective?");
            int defective = keyboard.nextInt();
            if (defective < 0)
               throw new NegativeNumberException("defective " +
                                                  "widgets");

            double ratio = exceptionalDivision(widgets,
                                                    defective);
            System.out.println("One in every " + ratio +
                               " widgets is defective.");
        }
```

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



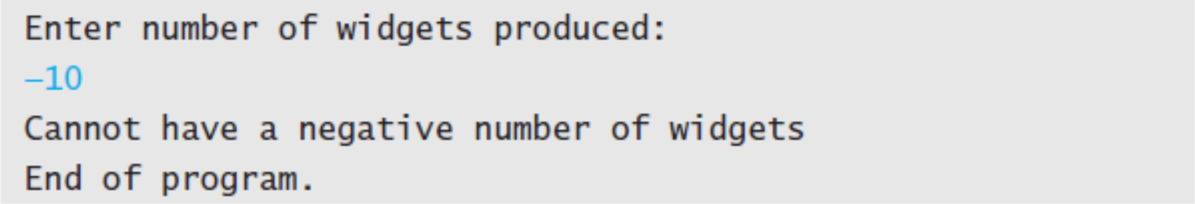
```
        catch(DivideByZeroException e)
        {
            System.out.println("Congratulations! A perfect " +
                               "record!");
        }
        catch(NegativeNumberException e)
        {
            System.out.println("Cannot have a negative number of " +
                               e.getMessage());
        }

        System.out.println("End of program.");
    }
    public static double exceptionalDivision(double numerator,
                  double denominator) throws DivideByZeroException
    {
        if (denominator == 0)
           throw new DivideByZeroException();
        return numerator / denominator;
    }
}
```

Sample Screen Output 1



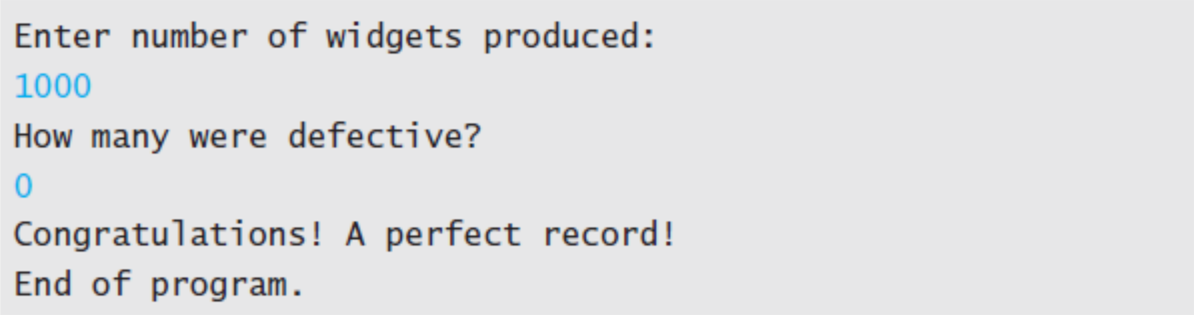
```
Enter number of widgets produced:
1000
How many were defective?
500
One in every 2.0 widgets is defective.
End of program.
```

Sample Screen Output 2

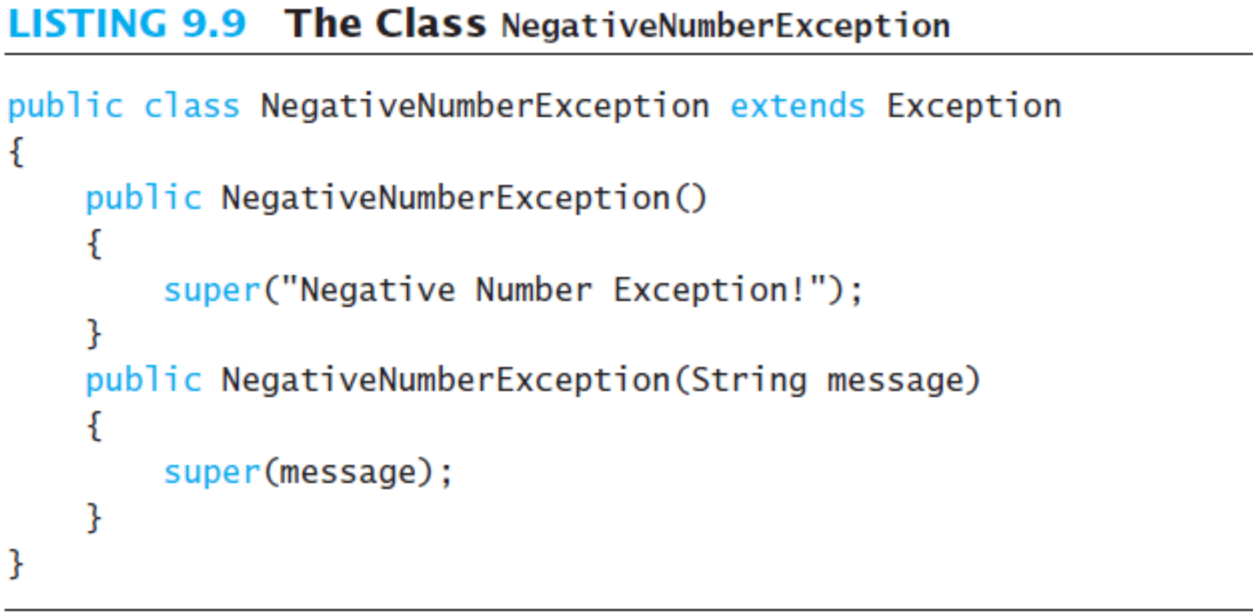


```
Enter number of widgets produced:
–10
Cannot have a negative number of widgets
End of program.
```

Sample Screen Output 3

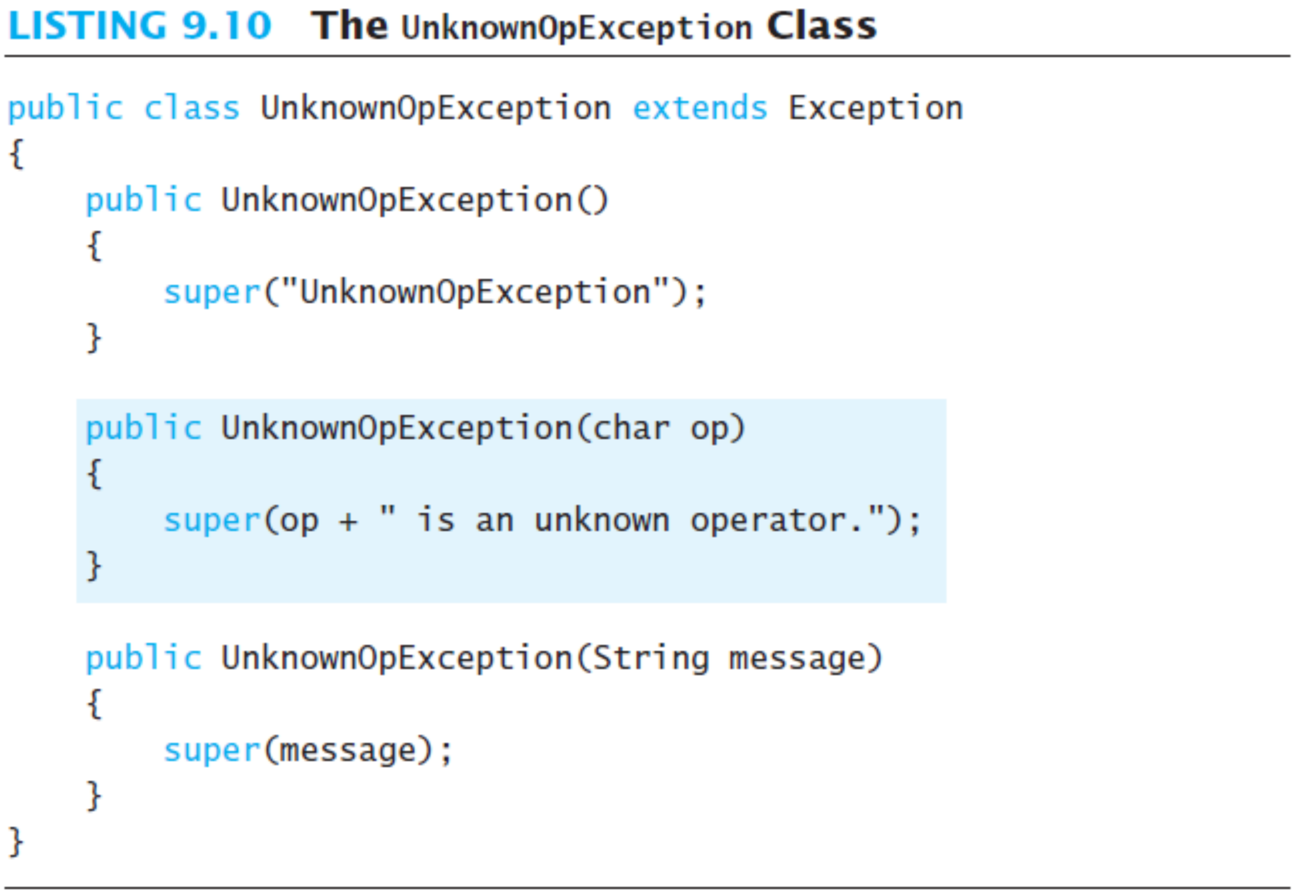


```
Enter number of widgets produced:
1000
How many were defective?
0
Congratulations! A perfect record!
End of program.
```



LISTING 9.9 The Class NegativeNumberException

```
public class NegativeNumberException extends Exception
{
    public NegativeNumberException()
    {
        super("Negative Number Exception!");
    }
    public NegativeNumberException(String message)
    {
        super(message);
    }
}
```



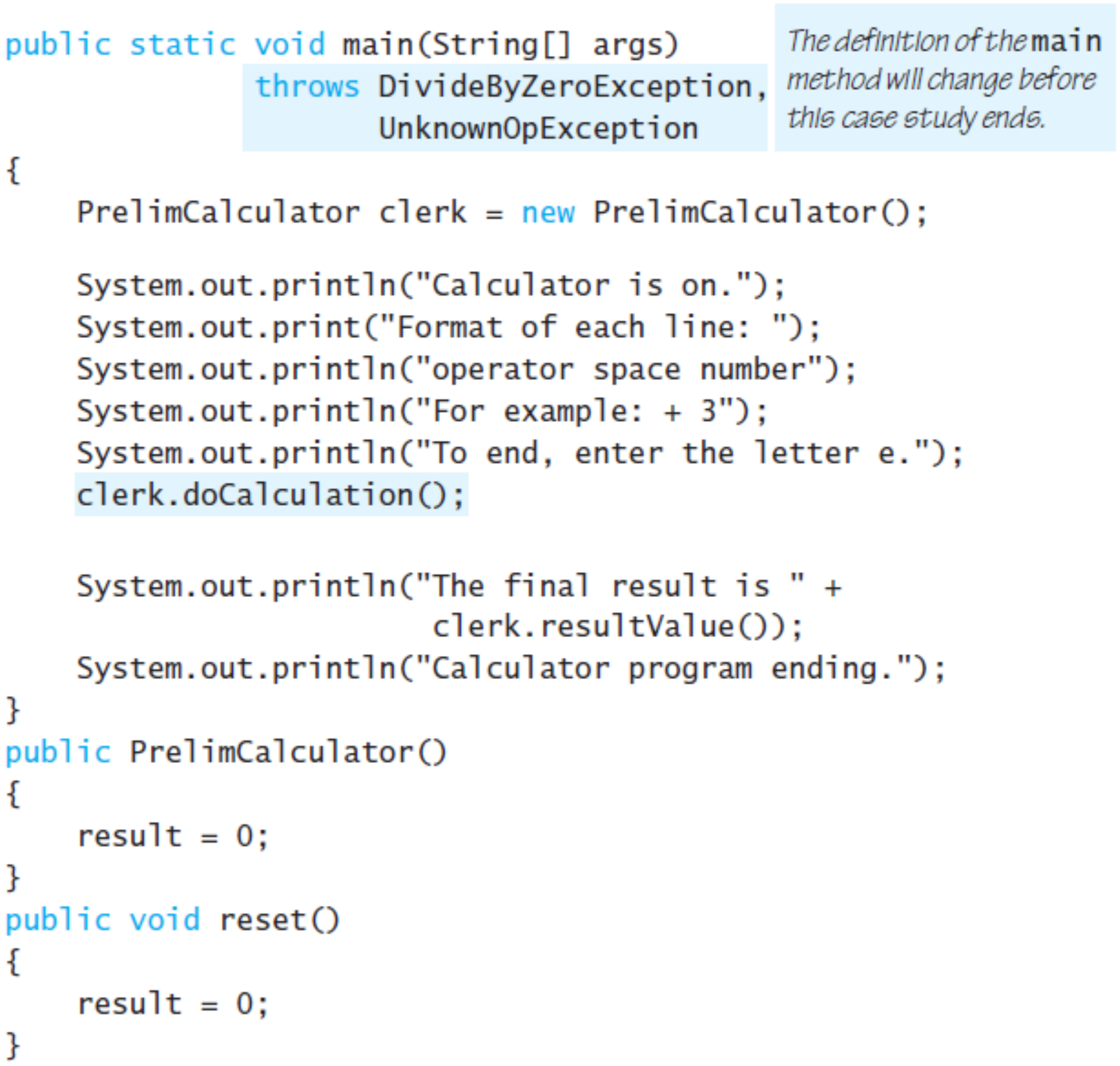
LISTING 9.10 The UnknownOpException Class

```
public class UnknownOpException extends Exception
{
    public UnknownOpException()
    {
        super("UnknownOpException");
    }

    public UnknownOpException(char op)
    {
        super(op + " is an unknown operator.");
    }

    public UnknownOpException(String message)
    {
        super(message);
    }
}
```

LISTING 9.11 The Unexceptional Cases *(part 1 of 3)*

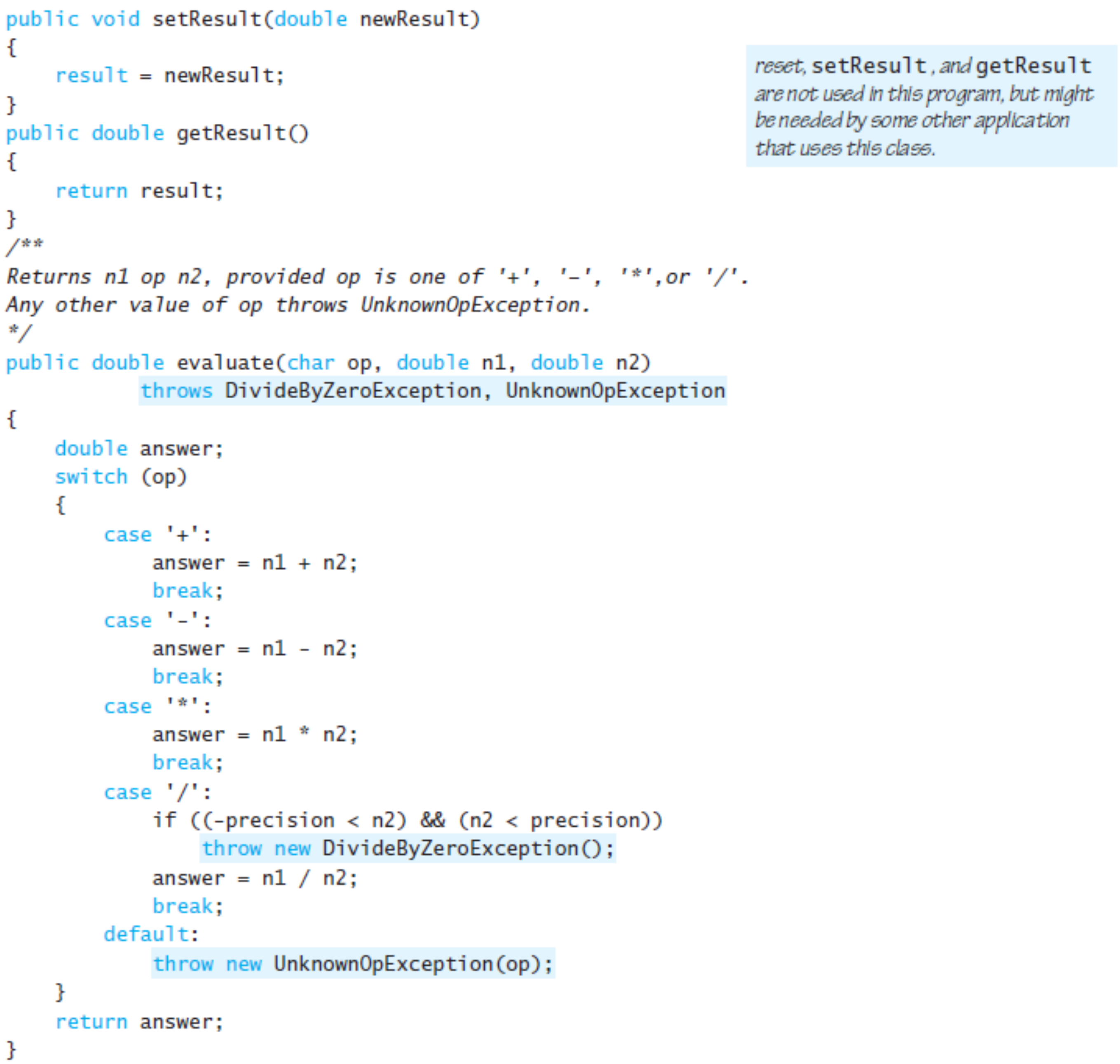
This version of the program does not handle exceptions and thus is not yet complete. However, it does run and can be used for debugging.

```java
import java.util.Scanner;
/**
PRELIMINARY VERSION without exception handling.
Simple line-oriented calculator program. The class
can also be used to create other calculator programs.
*/
public class PrelimCalculator
{
    private double result;
    private double precision = 0.0001;
    //Numbers this close to zero are treated as if equal to zero.
```



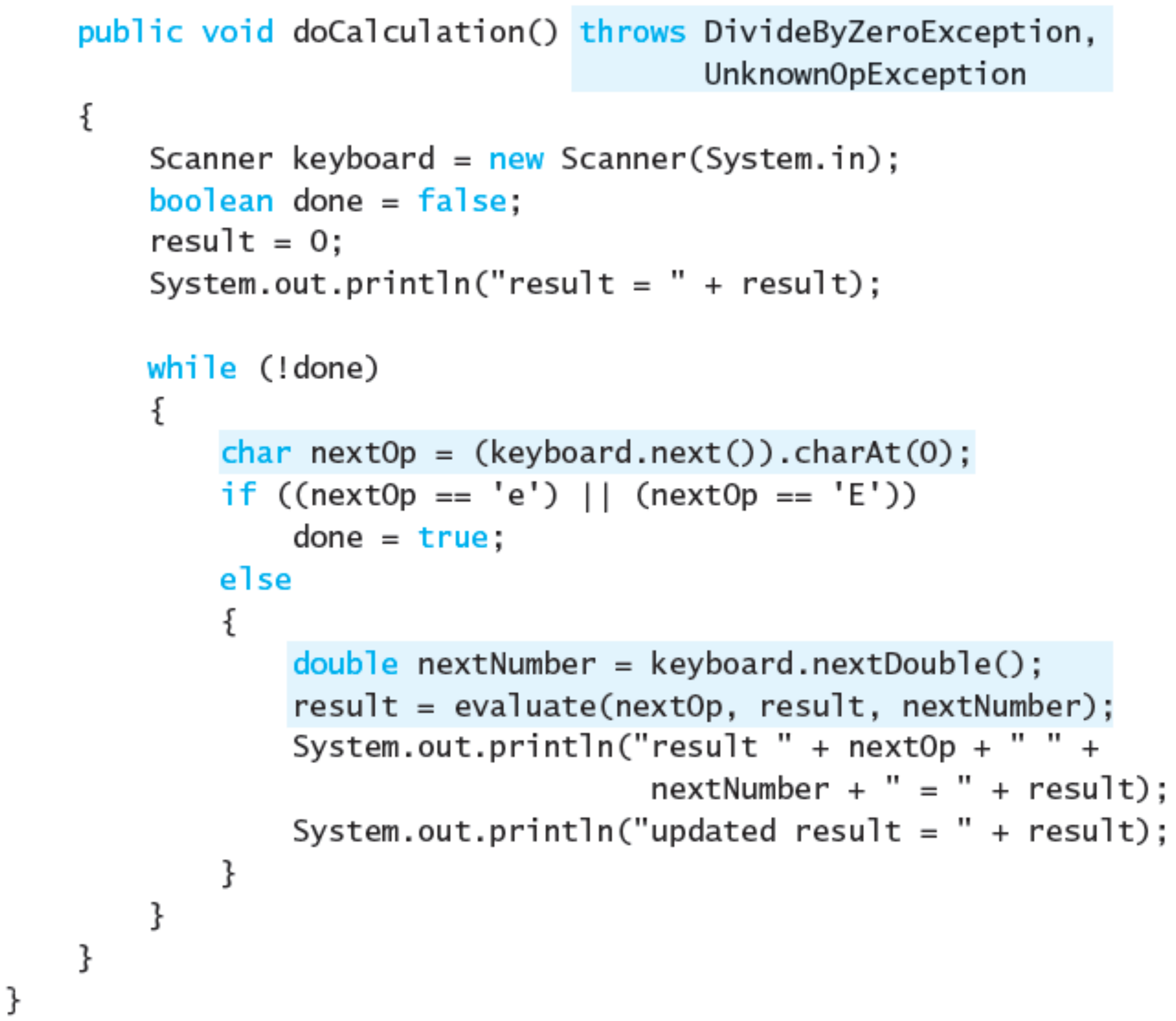
```
public static void main(String[] args)
               throws DivideByZeroException,
                      UnknownOpException
{
    PrelimCalculator clerk = new PrelimCalculator();

    System.out.println("Calculator is on.");
    System.out.print("Format of each line: ");
    System.out.println("operator space number");
    System.out.println("For example: + 3");
    System.out.println("To end, enter the letter e.");
    clerk.doCalculation();

    System.out.println("The final result is " +
                        clerk.resultValue());
    System.out.println("Calculator program ending.");
}
public PrelimCalculator()
{
    result = 0;
}
public void reset()
{
    result = 0;
}
```

The definition of the `main` *method will change before this case study ends.*



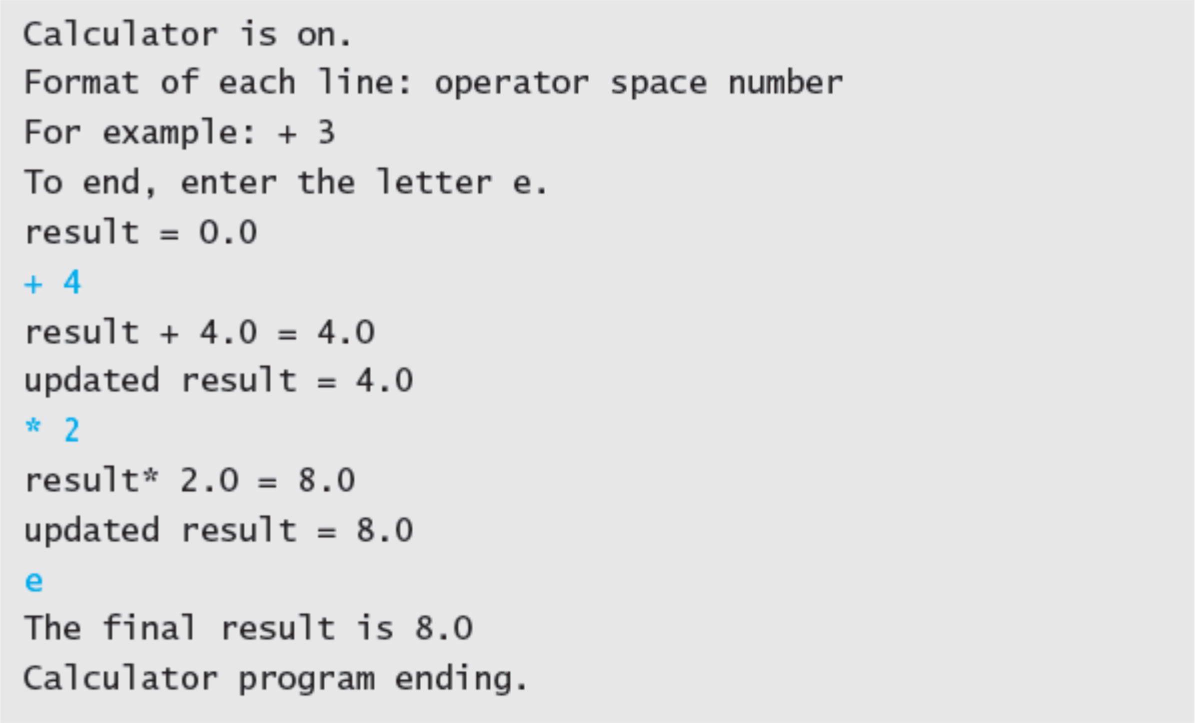
```
public void setResult(double newResult)
{
    result = newResult;
}
public double getResult()
{
    return result;
}
/**
Returns n1 op n2, provided op is one of '+', '-', '*',or '/'.
Any other value of op throws UnknownOpException.
*/
public double evaluate(char op, double n1, double n2)
           throws DivideByZeroException, UnknownOpException
{
    double answer;
    switch (op)
    {
        case '+':
            answer = n1 + n2;
            break;
        case '-':
            answer = n1 - n2;
            break;
        case '*':
            answer = n1 * n2;
            break;
        case '/':
            if ((-precision < n2) && (n2 < precision))
                throw new DivideByZeroException();
            answer = n1 / n2;
            break;
        default:
            throw new UnknownOpException(op);
    }
    return answer;
}
```

reset, `setResult`, *and* `getResult` *are not used in this program, but might be needed by some other application that uses this class.*

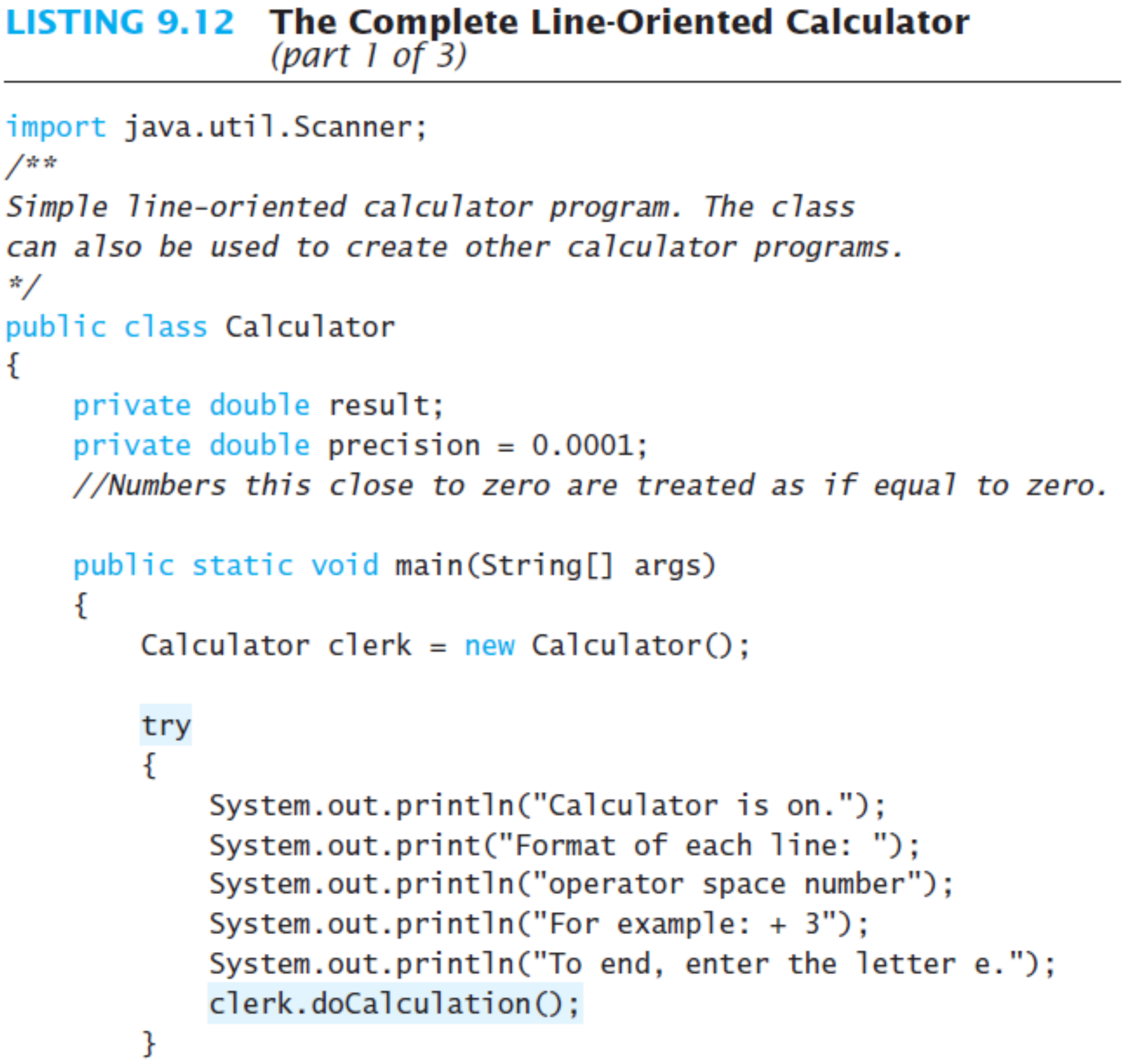


```
    public void doCalculation() throws DivideByZeroException,
                                       UnknownOpException
    {
        Scanner keyboard = new Scanner(System.in);
        boolean done = false;
        result = 0;
        System.out.println("result = " + result);

        while (!done)
        {
            char nextOp = (keyboard.next()).charAt(0);
            if ((nextOp == 'e') || (nextOp == 'E'))
                done = true;
            else
            {
                double nextNumber = keyboard.nextDouble();
                result = evaluate(nextOp, result, nextNumber);
                System.out.println("result " + nextOp + " " +
                                    nextNumber + " = " + result);
                System.out.println("updated result = " + result);
            }
        }
    }
}
```

Sample Screen Output

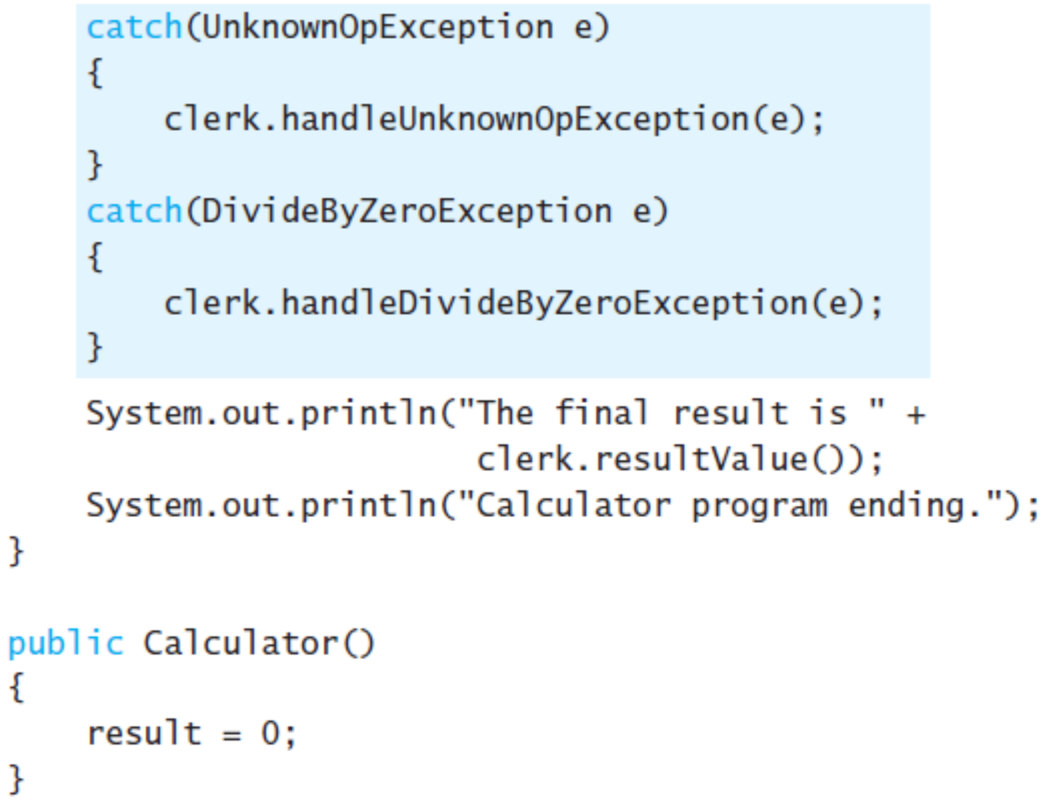


```
Calculator is on.
Format of each line: operator space number
For example: + 3
To end, enter the letter e.
result = 0.0
+ 4
result + 4.0 = 4.0
updated result = 4.0
* 2
result* 2.0 = 8.0
updated result = 8.0
e
The final result is 8.0
Calculator program ending.
```

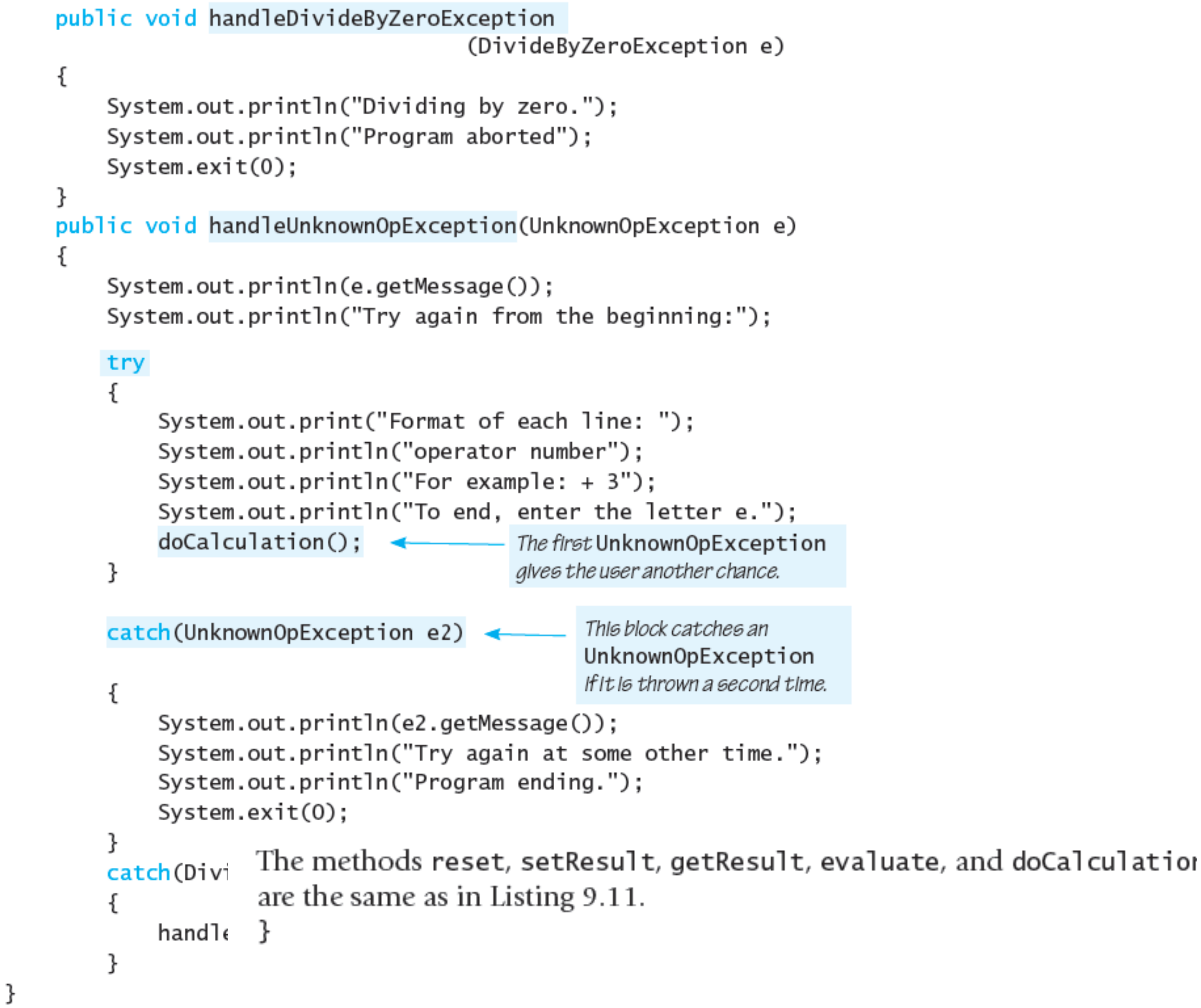


LISTING 9.12 The Complete Line-Oriented Calculator *(part 1 of 3)*

```java
import java.util.Scanner;
/**
Simple line-oriented calculator program. The class
can also be used to create other calculator programs.
*/
public class Calculator
{
    private double result;
    private double precision = 0.0001;
    //Numbers this close to zero are treated as if equal to zero.

    public static void main(String[] args)
    {
        Calculator clerk = new Calculator();

        try
        {
            System.out.println("Calculator is on.");
            System.out.print("Format of each line: ");
            System.out.println("operator space number");
            System.out.println("For example: + 3");
            System.out.println("To end, enter the letter e.");
            clerk.doCalculation();
        }
```



```java
    catch(UnknownOpException e)
    {
        clerk.handleUnknownOpException(e);
    }
    catch(DivideByZeroException e)
    {
        clerk.handleDivideByZeroException(e);
    }
    System.out.println("The final result is " +
                        clerk.resultValue());
    System.out.println("Calculator program ending.");
}

public Calculator()
{
    result = 0;
}
```



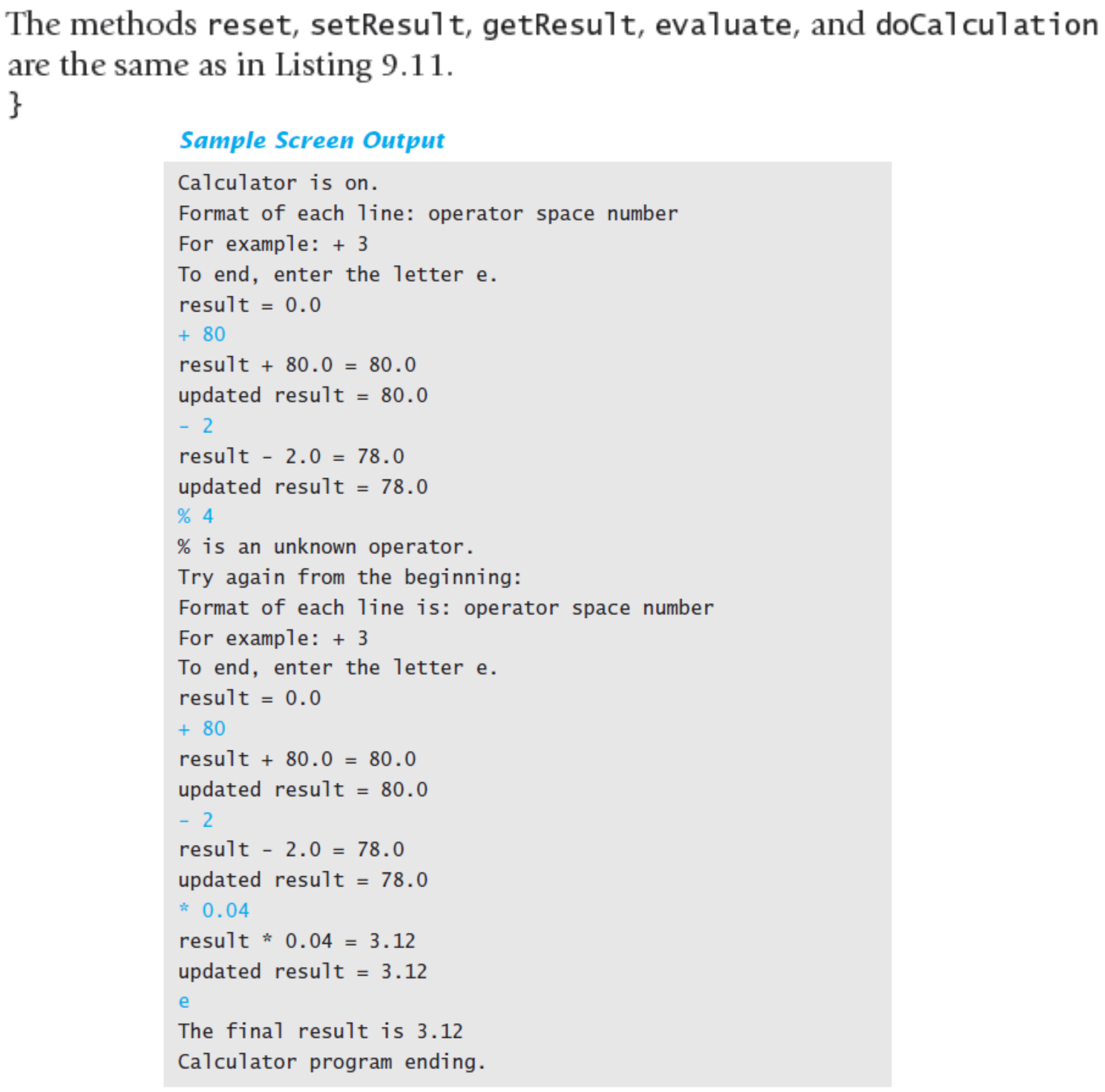
```
    public void handleDivideByZeroException
                                        (DivideByZeroException e)
    {
        System.out.println("Dividing by zero.");
        System.out.println("Program aborted");
        System.exit(0);
    }
    public void handleUnknownOpException(UnknownOpException e)
    {
        System.out.println(e.getMessage());
        System.out.println("Try again from the beginning:");

        try
        {
            System.out.print("Format of each line: ");
            System.out.println("operator number");
            System.out.println("For example: + 3");
            System.out.println("To end, enter the letter e.");
            doCalculation();
        }

        catch(UnknownOpException e2)

        {
            System.out.println(e2.getMessage());
            System.out.println("Try again at some other time.");
            System.out.println("Program ending.");
            System.exit(0);
        }
        catch(Divi
        {
            handl
        }
}
```

The first UnknownOpException *gives the user another chance.*

This block catches an UnknownOpException *if it is thrown a second time.*

The methods reset, setResult, getResult, evaluate, and doCalculatio
are the same as in Listing 9.11.
}

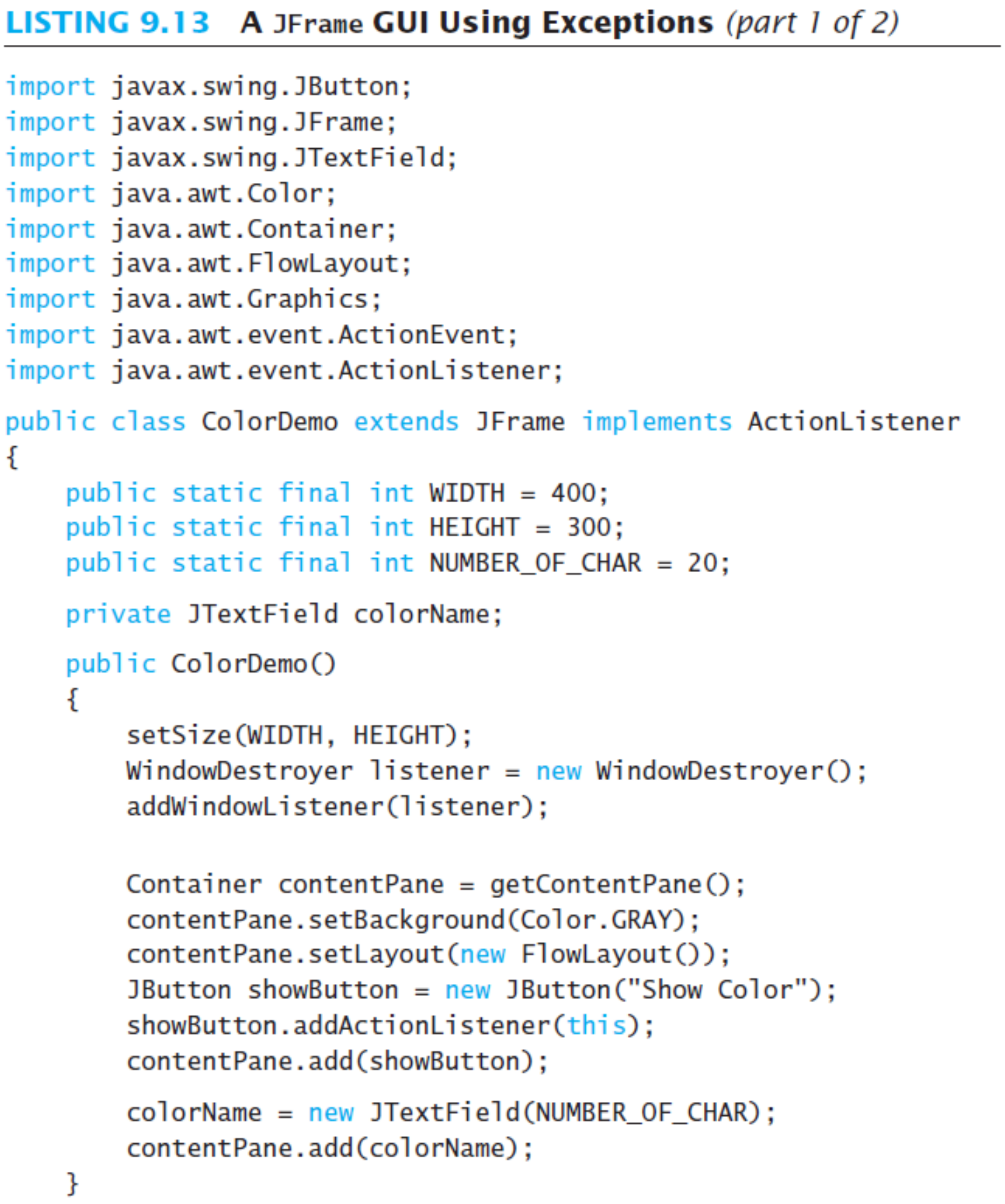
JAVA: An Introduction to Problem Solving & Programming, 6th Ed. By Walter Savitch



The methods reset, setResult, getResult, evaluate, and doCalculation are the same as in Listing 9.11.
}

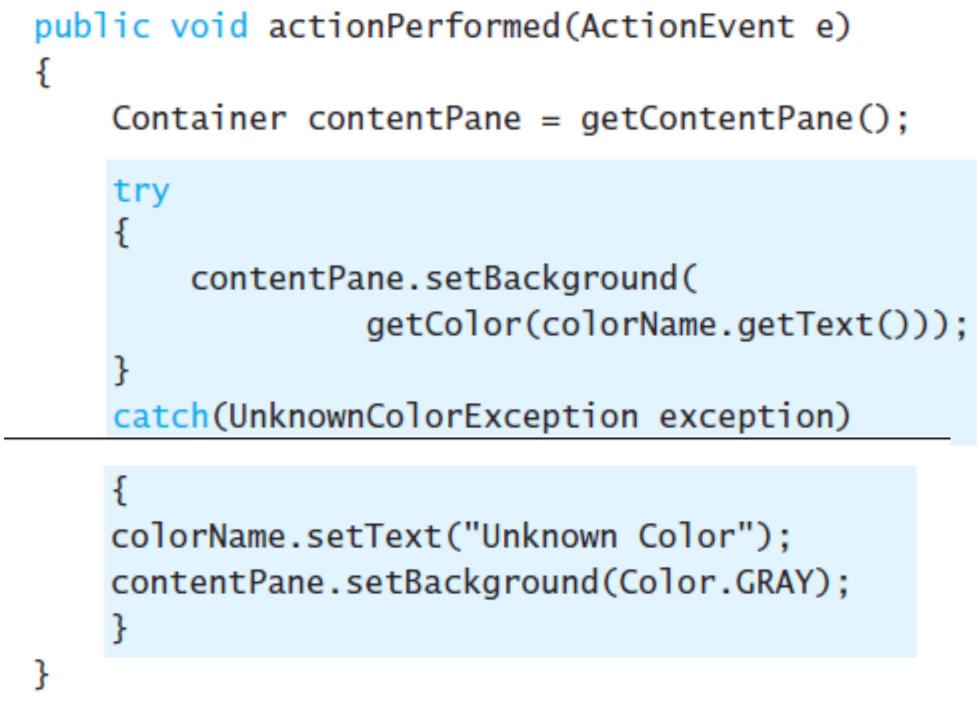
Sample Screen Output

```
Calculator is on.
Format of each line: operator space number
For example: + 3
To end, enter the letter e.
result = 0.0
+ 80
result + 80.0 = 80.0
updated result = 80.0
- 2
result - 2.0 = 78.0
updated result = 78.0
% 4
% is an unknown operator.
Try again from the beginning:
Format of each line is: operator space number
For example: + 3
To end, enter the letter e.
result = 0.0
+ 80
result + 80.0 = 80.0
updated result = 80.0
- 2
result - 2.0 = 78.0
updated result = 78.0
* 0.04
result * 0.04 = 3.12
updated result = 3.12
e
The final result is 3.12
Calculator program ending.
```

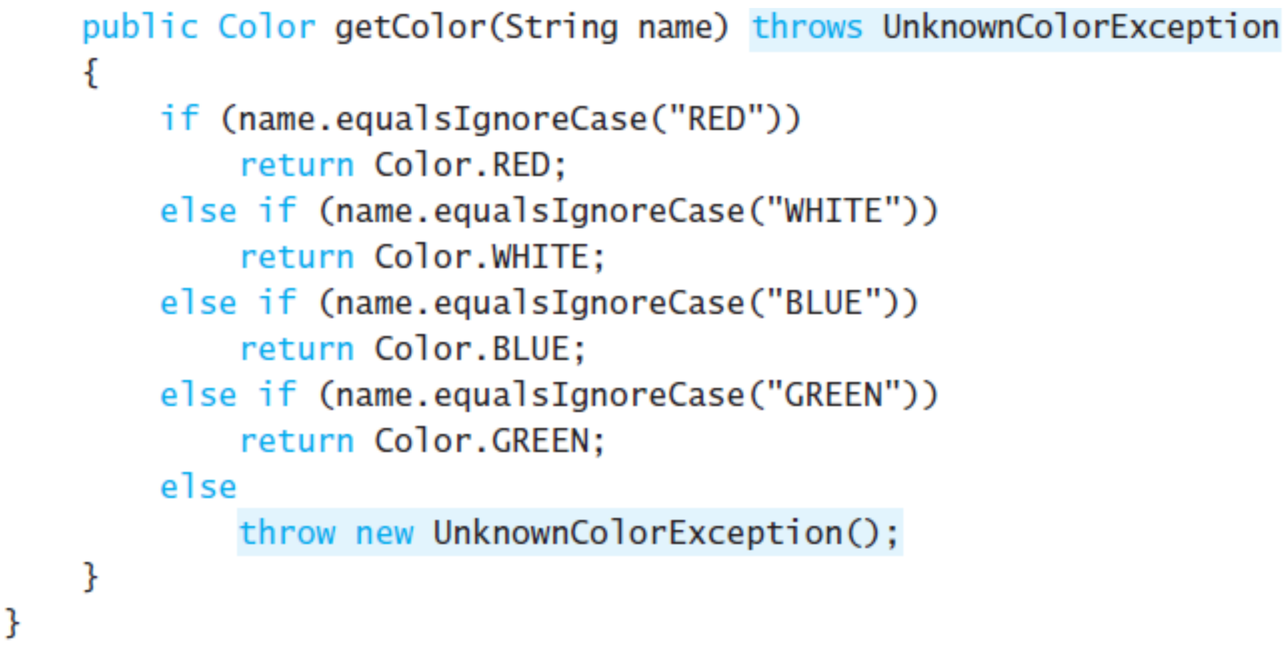


LISTING 9.13 A JFrame GUI Using Exceptions *(part 1 of 2)*

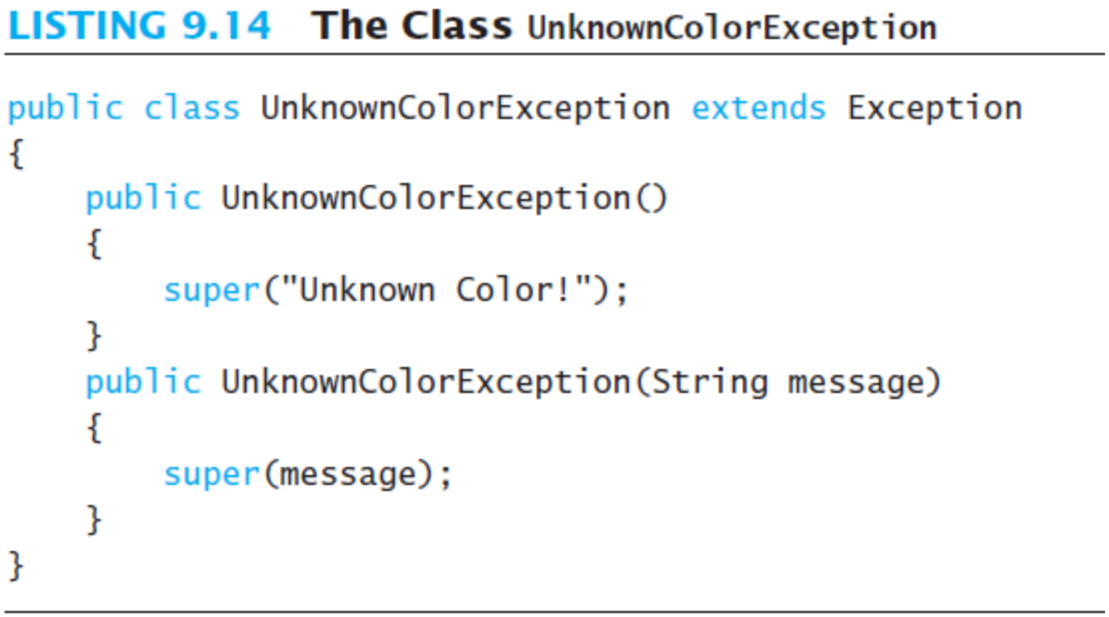
```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ColorDemo extends JFrame implements ActionListener
{
    public static final int WIDTH = 400;
    public static final int HEIGHT = 300;
    public static final int NUMBER_OF_CHAR = 20;

    private JTextField colorName;

    public ColorDemo()
    {
        setSize(WIDTH, HEIGHT);
        WindowDestroyer listener = new WindowDestroyer();
        addWindowListener(listener);

        Container contentPane = getContentPane();
        contentPane.setBackground(Color.GRAY);
        contentPane.setLayout(new FlowLayout());
        JButton showButton = new JButton("Show Color");
        showButton.addActionListener(this);
        contentPane.add(showButton);

        colorName = new JTextField(NUMBER_OF_CHAR);
        contentPane.add(colorName);
    }
```



```
public void actionPerformed(ActionEvent e)
{
    Container contentPane = getContentPane();
    try
    {
        contentPane.setBackground(
                 getColor(colorName.getText()));
    }
    catch(UnknownColorException exception)
    {
    colorName.setText("Unknown Color");
    contentPane.setBackground(Color.GRAY);
    }
}
```

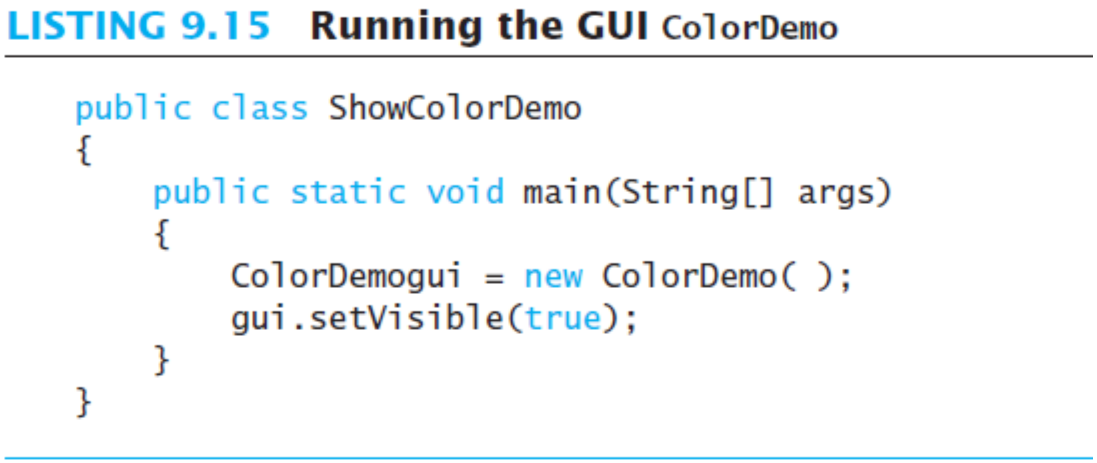


```
    public Color getColor(String name) throws UnknownColorException
    {
        if (name.equalsIgnoreCase("RED"))
            return Color.RED;
        else if (name.equalsIgnoreCase("WHITE"))
            return Color.WHITE;
        else if (name.equalsIgnoreCase("BLUE"))
            return Color.BLUE;
        else if (name.equalsIgnoreCase("GREEN"))
            return Color.GREEN;
        else
            throw new UnknownColorException();
    }
}
```



LISTING 9.14 The Class `UnknownColorException`

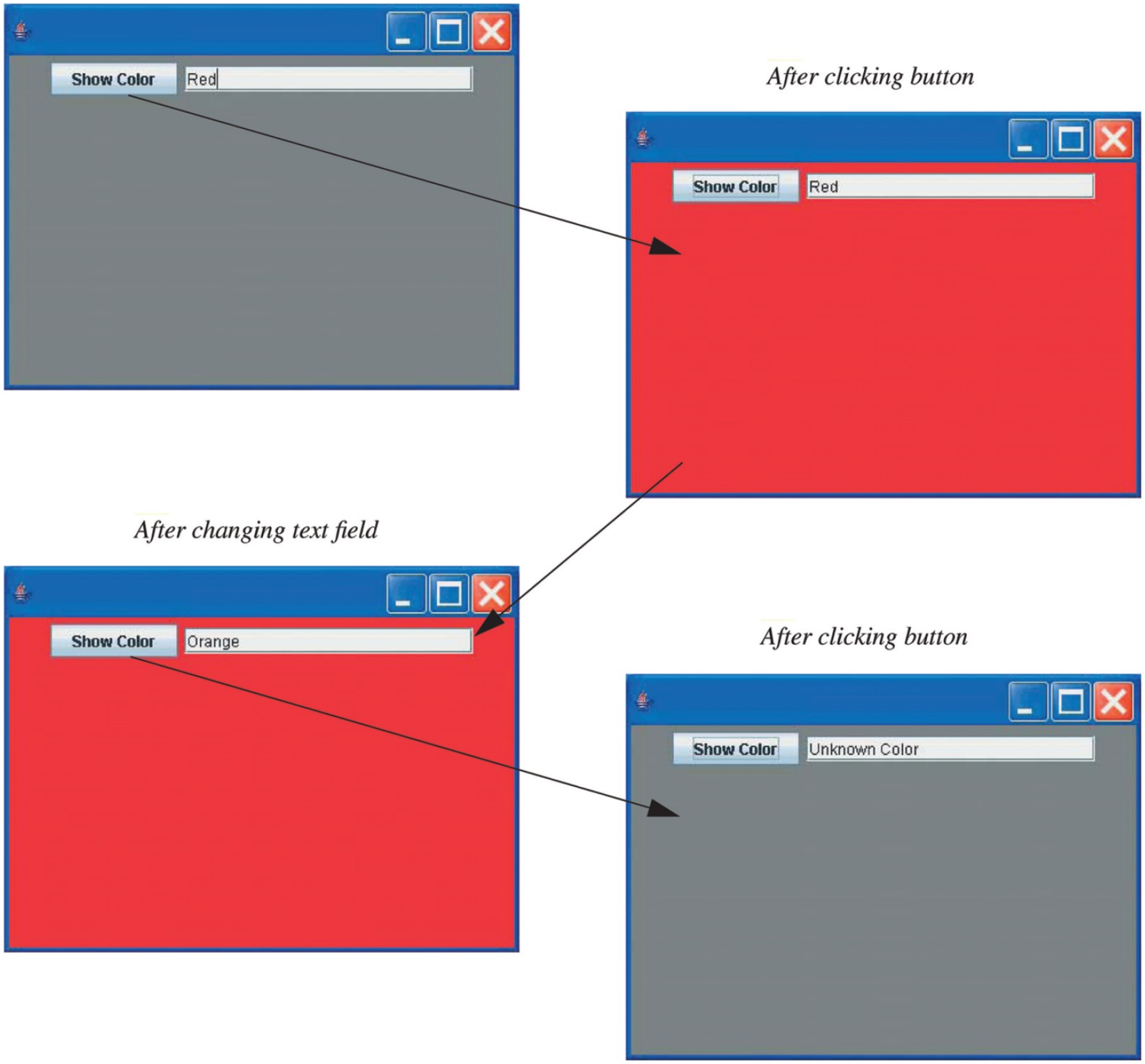
```
public class UnknownColorException extends Exception
{
    public UnknownColorException()
    {
        super("Unknown Color!");
    }
    public UnknownColorException(String message)
    {
        super(message);
    }
}
```



LISTING 9.15 Running the GUI ColorDemo

```
public class ShowColorDemo
{
    public static void main(String[] args)
    {
        ColorDemogui = new ColorDemo( );
        gui.setVisible(true);
    }
}
```

Sample Screen Output



Red

After clicking button

Red

After changing text field

Orange

After clicking button

Unknown Color

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