```
 1 // Jeff Offutt--Java version Feb 2003
 2 // Classify triangles
 3 import java.io.*;
 4
 5 class trityp
 6 {
 7   private static String[] triTypes = { "", // Ignore 0.
 8         "scalene", "isosceles", "equilateral",
          "not a valid triangle"};
 9   private static String instructions = "This is the ancient
     TriTyp program.\nEnter three integers that represent the
     lengths of the sides of a triangle.\nThe triangle will be
     categorized as either scalene, isosceles, equilateral\n
     or invalid.\n";
10
11 public static void main (String[] argv)
12 {  // Driver program for trityp
13   int A, B, C;
14   int T;
15
16   System.out.println (instructions);
17   System.out.println ("Enter side 1: ");
18   A = getN();
19   System.out.println ("Enter side 2: ");
20   B = getN();
21   System.out.println ("Enter side 3: ");
22   C = getN();
23   T = Triang (A, B, C);
24
25   System.out.println ("Result is: " + triTypes[T]);
26 }
27
28 // ================================
29 // The main triangle classification method
30 private static int Triang (int Side1, int Side2, int Side3)
31 {
32   int triOut;
33
34   // triOut is output from the routine:
35   //    Triang = 1 if triangle is scalene
36   //    Triang = 2 if triangle is isosceles
37   //    Triang = 3 if triangle is equilateral
38   //    Triang = 4 if not a triangle
39
40   // After a quick confirmation that it's a valid
41   // triangle, detect any sides of equal length
42   if (Side1 <= 0 || Side2 <= 0 || Side3 <= 0)
43   {
44     triOut = 4;
45     return (triOut);
46   }
47
48   triOut = 0;
49   if (Side1 == Side2)
50     triOut = triOut + 1;
51   if (Side1 == Side3)
52     triOut = triOut + 2;
53   if (Side2 == Side3)
54     triOut = triOut + 3;
55   if (triOut == 0)
56   {  // Confirm it's a valid triangle before declaring
57     // it to be scalene
58
```

Figure 3.2. TriTyp – Part A.

```
59      if (Side1+Side2 <= Side3 || Side2+Side3 <= Side1 ||
60          Side1+Side3 <= Side2)
61        triOut = 4;
62      else
63        triOut = 1;
64      return (triOut);
65    }
66
67    // Confirm it's a valid triangle before declaring
68    // it to be isosceles or equilateral
69
70    if (triOut > 3)
71      triOut = 3;
72    else if (triOut == 1 && Side1+Side2 > Side3)
73      triOut = 2;
74    else if (triOut == 2 && Side1+Side3 > Side2)
75      triOut = 2;
76    else if (triOut == 3 && Side2+Side3 > Side1)
77      triOut = 2;
78    else
79      triOut = 4;
80    return (triOut);
81  } // end Triang
82
83  // ============================================
84  // Read (or choose) an integer
85  private static int getN ()
86  {
87    int inputInt = 1;
88    BufferedReader in = new BufferedReader (new InputStreamReader (System.in));
89    String inStr;
90
91    try
92    {
93      inStr    = in.readLine ();
94      inputInt = Integer.parseInt(inStr);
95    }
96    catch (IOException e)
97    {
98      System.out.println ("Could not read input, choosing 1.");
99    }
100   catch (NumberFormatException e)
101   {
102     System.out.println ("Entry must be a number, choosing 1.");
103   }
104
105   return (inputInt);
106 } // end getN
107
108 } // end trityp class
```

Figure 3.3. TriTyp – Part B.

the testing literature for many years. As an example, it has several advantages: its purpose is relatively easy to understand, it is small enough to fit in a classroom exercise, and it has a very complicated logic structure that can illustrate most of the concepts. This version of TriTyp is written in Java and was compiled and tested with Sun's JDK Java 1.4.1. Line numbers have been added to allow us to refer to specific decision statements in the text.

Predicates are taken from decision points in the program, including if statements. case/switch statements, for loops, while loops, and do-until loops. This is illustrated with the Triang() method in the TriTyp program. Triang() has the following predicates (line numbers are shown on the left, and the else statements at lines 62 and 78 do not have their own predicates):