```java
// Class Trt implements the logic for classifying triangles.

public class Trt {

    // Method to classify triangles based on their type.
    public static int classifyTriangle(int sideA, int sideB, int sideC) {
        if (sideA + sideB <= sideC || sideA + sideC <= sideB || sideB + sideC <= sideA) {
            return -1; // Not a valid triangle.
        }
        int maxSide = Math.max(sideA, Math.max(sideB, sideC));
        if (sideA == maxSide && sideB == maxSide && sideC == maxSide) {
            return 0; // Equilateral triangle.
        }
        if (sideA == maxSide || sideB == maxSide || sideC == maxSide) {
            return 1; // Isosceles triangle.
        }
        return 2; // Scalene triangle.
    }
}
```

**Figure 3.2: Trt**

- **example**: The `classifyTriangle` method takes three parameters representing the sides of a triangle and returns an integer indicating the type of triangle: 0 for equilateral, 1 for isosceles, 2 for scalene.
- **assertions**: The logic checks if the given sides form a valid triangle and classifies it accordingly.
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 as lines 62 and 70 do not have their own predicates):