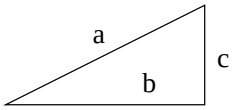
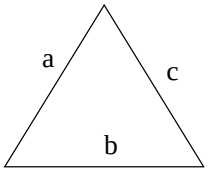
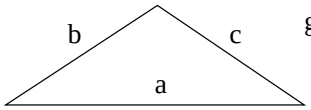
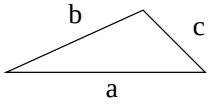
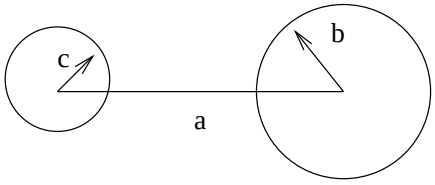
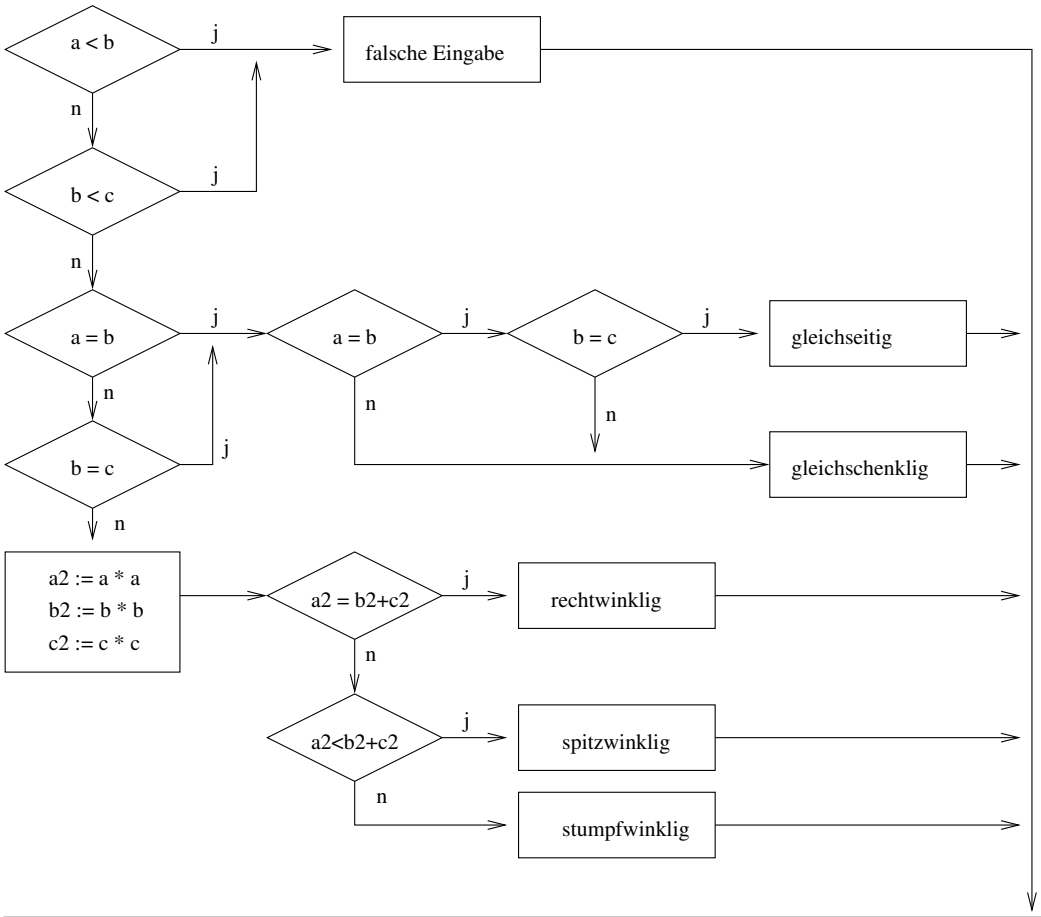
Testdaten aus Blackbox-Methode

a	b	c	Soll−Ergebnisse		
5	4	3			rechtwinklig **0 0 0 0 1**
5	5	5		gleichseitig	spitzwinklig **0 1 1 0 0**
5	4	4		gleichschenklig	stumpfwinklig **1 0 0 1 0**
5	4	2			stumpfwinklig **0 0 0 1 0**
5	2	1			kein Dreieck
0	0	0			kein Dreieck
−5	−6	−7			kein Dreieck
5	6	7			**wrong input**

Testdaten aus Whitebox-Methode



Eingaben		Testfall 1	2	3	4	5	6
a	5	x	x	x	x	x	x
b	6	x					
	5		x	x			x
	4				x	x	
c	5	x	x				
	3			x	x		x
	2					x	
Soll-Ergebnisse		falsche Eingabe	gleich-seitig	gleich-schenklig	recht-winklig	spitz-winklig	stump-winklig

Testprogramm

C++

```
#include <iostream>
#include "Dreieck.h"

int main( int argc, char **argv ){
  double a,b,c;
  std::cin >> a >> b >> c;

  try {
    Dreieck d( a,b,c );
    std::cout << d.ist_gleichschenklig()
              << d.ist_gleichseitig()
              << d.ist_rechtwinklig()
              << d.ist_stumpfwinklig()
              << d.ist_spitzwinklig()
              << std::endl;
  }
  catch ( ... ){
    std::cout << "wrong input" << std::endl;
  }
  return 0;
}
```

Java

```java
package geometry;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.regex.Pattern;

public class TriangleApp {
  public static void main( String args[] ){
    try {
      Pattern p = Pattern.compile(" ");
      BufferedReader in = new BufferedReader(
                new InputStreamReader( System.in ) );

      String line = null;
      while( (line = in.readLine()) != null ){
        String [] result = p.split( line );
        if ( result.length != 3 ){
            throw new Exception( "wrong number of input values" );
        }
        double a = Double.parseDouble( result[0] );
        double b = Double.parseDouble( result[1] );
        double c = Double.parseDouble( result[2] );
        Triangle t = new TriangleImpl( a, b, c );
                System.out.println( t );
      }
    }
    catch( Exception e ){
            System.out.println( e.getMessage() );
    }
  }
}
```

Perl

```
#!/usr/local/bin/perl
#
# Dreieck Test Programm
#
#
my %test_cases = (                  # Definition der Testflle:
    '5 4 3', '00100',     #  pro Zeile: Eingabe Soll-Ergebnis
    '5 5 5', '11001',               #
    '5 4 4', '10010',               # Ergebnis: (jeweils 0 oder 1)
    '5 4 2', '00010',               #  1. gleichschenklig
    '5 2 1', 'wrong input,          #  2. gleichseitig
    '0 0 0', 'wrong input'          #  3. rechtwinklig
);                                  #  4. stumpfwinklig
                                    #  5. spitzwinklig
                                    # oder: "wrong input"

foreach $input ( keys( %test_cases ) ){

  open( TEMP, ">input" );
  print TEMP "$input\n";
  close TEMP;

  $pid=open(DREIECK, "DreieckTest <input |" )
      || die "Can't start DreieckTest: $!\n";

  while( <DREIECK> ){
    if( /$test_cases{$input}/ ){
      print "Test: $input passed\n";
    }
    else {
      print "Test: $input failed\n";
    }
  }
  close DREIECK;
}
```

Python

```python
#!/usr/bin/python
#
import subprocess

testcases=[
    ['5 4 3', '00100'],
    ['5 5 5', '11001'],
    ['5 4 4', '10010'],
    ['5 4 2', '00010'],
    ['5 2 1', 'wrong input'],
    ['0 0 0', 'wrong input'],
    ['1 2 3', 'wrong input']]

failed=0
runs=0
for t in testcases:
    p = subprocess.Popen(["java", "-cp", "bin",
                          "geometry.TriangleApp"], bufsize=1,
                         stdin=subprocess.PIPE,
                         stdout=subprocess.PIPE,
                         close_fds=True)
    p.stdin.write(t)
    p.stdin.close()
    runs = runs + 1
    for l in p.stdout:
        if t[1] == l.strip():
            print "Test %s passed" % t[0]
        else:
            failed = failed + 1
            print "Test %s (expected %s) failed " % (t[0], t[1])

print "Total %d (failed %d)" % (runs, failed)
```

Cucumber

```
Feature: Classify Triangles
  As a student I want to use classify triangles.

  Scenario: Determination of a right angled triangle
    Given a triangle
    When I enter the sides 5.0, 4.0, 3.0
    Then the resulting type should be "00001"
```

```
public class TriangleClassificationStepDefs {

  Triangle triangle;

  @Given("a triangle")
  public void init_feature() {}

  @When("I enter the sides {double}, {double}, {double}")
  public void add_the_sides(double a, double b, double c)
      throws Exception {
    triangle = new TriangleImpl(a, b, c);
  }

  @Then("the resulting type should be {string}")
  public void the_type_should_be(String expectedType) {
    assertEquals("Triangle Type",
                 expectedType, triangle.toString());
  }
}
```