Test cases of triangle test

assume that the input values are `a,b,c` respectively and `INT_MAX=0x7fffffff`

1. Argument type tests

Logical test case

Case #	input values	expected result
L1	any one of the value in `a,b,c` is **not an integer**	program throws self-defined `ArgumentTypeException`

Corresponding concrete test cases

Case #	input values	expected result
C1.1	`a=1,b=1.2,c=2`	program throws self-defined `ArgumentTypeException` as `b` is a float number
C1.2	`a=1,b='a',c=2`	program throws self-defined `ArgumentTypeException` as `b` is a character
C1.3	`a=true,b=new int[]{1},c=1`	program throws self-defined `ArgumentTypeException` as `a` is bool and `b` is an array

Corresponding test code in Java:

```
// Test 1: Illegal argument type
    @Test
    public void testType()
    {
        int[] testArray = new int[]{1};
        Triangle testFloat = new Triangle(1,1.2,2);
        Triangle testChar = new Triangle(1,'a',2);
        Triangle testOthers = new Triangle(true,testArray,1);

        // float number error
        assertThrows(ArgumentTypeException.class, testFloat::judgeTriangle);

        // character error
        assertThrows(ArgumentTypeException.class, testChar::judgeTriangle);

        // other error(including boolean and array)
        assertThrows(ArgumentTypeException.class, testOthers::judgeTriangle);

    }
```

2. Boundary tests

Logical test case

Case #	input values	expected result
L2	any one of the value in `a,b,c` is **less or equal to** `0` or **greater than** `INT_MAX`	program throws self-defined `ArgumentRangeException`

Corresponding concrete test cases

Case #	input values	expected result
C2.1	`a=0,b=2,c=1`	program throws self-defined `ArgumentRangeException` as `a==0`
C2.2	`a=-1,b=4,c=5`	program throws self-defined `ArgumentRangeException` as `a<0`
C2.3	`a=INT_MAX+1,b=INT_MAX+1,c=1`	program throws self-defined `ArgumentRangeException` as `a>INT_MAX` and `b>INT_MAX`
C2.4	`a=INT_MAX,b=INT_MAX,c=1`	program returns `"Acute scalene triangle"`

Corresponding test code in Java:

```
// Test 2: Boundary
// OUT_RANGE=INT_MAX+1
    @Test
    public void testRange()
    {
        Triangle testZero = new Triangle(0,2,1);
        Triangle testOutOfMin = new Triangle(-1,4,5);
        Triangle testOutOfMax = new Triangle(OUT_RANGE,OUT_RANGE,1);
        Integer IN_RANGE = Integer.MAX_VALUE;
        Triangle testInRange = new Triangle(IN_RANGE, IN_RANGE,1);

        // out of range error
        assertThrows(ArgumentRangeException.class, testOutOfMax::judgeTriangle);
        assertThrows(ArgumentRangeException.class, testZero::judgeTriangle);
        assertThrows(ArgumentRangeException.class,testOutOfMin::judgeTriangle);

        // in range, normal output
        assertEquals("Acute isosceles triangle",testInRange.judgeTriangle());
    }
```

3. Legality tests (whether the 3 sides can form a triangle)

Logical test case

Case #	input values	expected result
L3	`a+b<=c` or `b+c<=a` or `c+a<=b`	program throws self-defined `CannotFormTriangleException`

Corresponding concrete test case

Case #	input values	expected result
C3.1	`a=1,b=2,c=4`	program throws self-defined `CannotFormTriangleException` as `a+b<c`

Corresponding test code in Java:

```
// Test 3: Legality of sides to form a triangle
@Test
public void testLegal()
{
    Triangle testIllegal = new Triangle(1,2,4);

    // illegal sides to form a triangle
    assertThrows(CannotFormTriangleException.class,testIllegal::judgeTriangle);
}
```

4. Shape judgement tests

1. Scalene triangles

In **all of the cases below** input values shall satisfy: `a!=b` and `b!=c` and `c!=a`

I have divided it into three subsets: Acute,Right and Obtuse.

Logical test cases

Case #	input values	expected result
L4.1.1	`a*a+b*b>c*c` and `b*b+c*c>a*a` and `c*c+a*a>b*b`	program returns `"Acute scalene triangle"`
L4.1.2	`a*a+b*b==c*c` or `b*b+c*c==a*a` or `c*c+a*a==b*b`	program returns `"Right scalene triangle"`
L4.1.3	Otherwise	program returns `"Obtuse scalene triangle"`

Corresponding concrete test case

Case #	input values	expected result
C4.1.1	`a=8,b=7,c=6`	program returns `"Acute scalene triangle"`
C4.1.2	`a=10,b=6,c=8`	program returns `"Right scalene triangle"`
C4.1.3	`a=7,b=3,c=6`	program returns `"Obtuse scalene triangle"`

2. Isosceles triangles

In **all of the cases below** input values shall satisfy: `a==b` or `b==c` or `c==a` but **not all of the sides are equal to each other**

I have divided it into three subsets as above

Logical test cases

Case #	input values	expected result
L4.2.1	`a*a+b*b>c*c` and `b*b+c*c>a*a` and `c*c+a*a>b*b`	program returns `"Acute isosceles triangle"`
L4.2.2	`a*a+b*b==c*c` or `b*b+c*c==a*a` or `c*c+a*a==b*b`	program throws self-defined `ArgumentTypeException`
L4.2.3	Otherwise	program returns `"Obtuse isosceles triangle"`

Corresponding concrete test cases

Case #	input values	expected result
C4.2.1	`a=4,b=3,c=4`	program returns `"Acute isosceles triangle"`
C4.2.2	`a=2,b=2,c=2*sqrt(2)`	program throws self-defined `ArgumentTypeException` as `c` is not an integer
C4.2.3	`a=3,b=3,c=5`	program returns `"Obtuse ispsceles triangle"`

3. Equilateral triangles

Logical test case

Case #	input values	expected result
L4.3	`a==b` and `b==c`	program returns `"Equilateral triangle"`

Corresponding concrete test case

Case #	input values	expected result
C4.3	`a=3,b=3,c=3`	program returns `"Equilateral triangle"`

Corresponding test code of all the test cases above in Java

```
// Test 4: Other normal situations with different shapes
@Test
public void testNormal()
{
    // scalene triangles
    Triangle acuteScalene = new Triangle(8,6,7);  // acute
    Triangle rightScalene = new Triangle(10,6,8); // right
    Triangle obtuseScalene = new Triangle(7,3,6); // obtuse

    assertEquals("Acute scalene triangle",acuteScalene.judgeTriangle());
    assertEquals("Right scalene triangle",rightScalene.judgeTriangle());
    assertEquals("Obtuse scalene triangle",obtuseScalene.judgeTriangle());

    // isosceles triangles
    Triangle acuteIsosceles = new Triangle(4,3,4); // acute
    Triangle rightIsosceles = new Triangle(2,2,2*Math.sqrt(2)); //right
    Triangle obtuseIsosceles = new Triangle(3,3,5); // obtuse

    assertEquals("Acute isosceles triangle",acuteIsosceles.judgeTriangle());
    assertThrows(ArgumentTypeException.class,rightIsosceles::judgeTriangle);
    assertEquals("Obtuse isosceles triangle",obtuseIsosceles.judgeTriangle());

    // equilateral triangle
    Triangle equilateral = new Triangle(4,4,4);
    assertEquals("Equilateral triangle",equilateral.judgeTriangle());

}
```