軟體品質保證與軟體測試

M2.2

Mission 2 Triangle

資訊三甲 D0745378 薛竣祐

執行畫面

一張含有 文字 的圖片

自動產生的描述

Triangle.java

package JUnit;  
  
public class Triangle {  
  
 double[] lines;  
 String type;  
 double delta = 0.0000000001;//精度誤差  
  
 Triangle(double a1, double a2, double a3) throws Exception {  
 lines = new double[]{a1, a2, a3};  
 }  
  
 Triangle setTriangleType() throws Exception {  
 if (checkUpright()) {  
 this.type = "正三角形";  
 } else if (checkSameAndNinety()) {  
 this.type = "等腰直角三角形";  
 } else if (checkNinety()) {  
 this.type = "直角三角形";  
 } else if (checkTwoSameLine()) {  
 this.type = "等腰三角形";  
 } else if (checkNormal()) {  
 this.type = "一般三角形";  
 } else if (checkNotTri()) {  
 throw new Exception("Not A Triangle");  
 }  
 return this;  
 }  
  
 boolean checkUpright() {//正三角形 (contain twoSame & Normal)  
 boolean sameLine = compareD(this.lines[0], this.lines[1])

&& compareD(this.lines[1], this.lines[2]);  
 return checkNormal() && sameLine;  
 }  
  
 boolean checkSameAndNinety() {//等腰直角三角形(contain twoSame & ninety & Normal)  
 return checkNinety() && checkTwoSameLine();  
 }  
  
 boolean checkNinety() {//直角三角形(contain Normal)  
 boolean is\_ninety = false;  
 for (int i = 0; i < 3; i++) {  
 is\_ninety = is\_ninety || pythagorean(

this.lines[i % 3],

this.lines[(i + 1) % 3],

this.lines[(i + 2) % 3]

)  
 }  
 return checkNormal() && is\_ninety;  
 }  
  
 boolean checkTwoSameLine() {//等腰三角形(contain Normal)  
 boolean same = false;  
 for (int i = 0; i < 3; i++) {  
 same = same || compareD(this.lines[i % 3], this.lines[(i + 1) % 3]);  
 }  
 return checkNormal() && same;  
 }

boolean checkNormal() {//三角形  
 boolean twoSumBigger = true;  
 boolean noZero = false;  
 for (int i = 0; i < 3; i++) {  
 twoSumBigger = twoSumBigger &&

this.lines[i % 3] + this.lines[(i + 1) % 3] > this.lines[(i + 2) % 3];  
 noZero = noZero || this.lines[i] > 0;  
 }  
 return twoSumBigger && noZero;  
 }  
  
 boolean checkNotTri() {//不是三角形  
 return !checkNormal();  
 }  
  
 boolean pythagorean(double a, double b, double c) {  
 return compareD(c \* c, a \* a + b \* b);  
 }  
  
 boolean compareD(double a, double b) {  
 //return if a == b (+-delta)  
 return Math.*abs*(a - b) <= delta;  
 }  
  
}

TriangleTest.java

package JUnit;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TriangleTest {  
 @Test  
 void setTriangleType() throws Exception {  
 *assertThrows*(Exception.class, () -> new Triangle(0,0,0).setTriangleType());  
 *assertThrows*(Exception.class, () -> new Triangle(-1,5,9).setTriangleType());  
 *assertThrows*(Exception.class, () -> new Triangle(3,3,6).setTriangleType());  
 *assertThrows*(Exception.class, () -> new Triangle(1,2,3).setTriangleType());  
 *assertEquals*("正三角形",new Triangle(1, 1, 1).setTriangleType().type);  
 *assertEquals*("等腰三角形",new Triangle(5, 5, 1).setTriangleType().type);  
 *assertEquals*("直角三角形",new Triangle(3, 4, 5).setTriangleType().type);  
 *assertEquals*("等腰直角三角形",new Triangle(1,1,Math.*pow*(2,0.5)).setTriangleType().type);  
 *assertEquals*("一般三角形",new Triangle(3, 5, 7).setTriangleType().type);

System.*out*.println("setTriangleType() Test End");

}  
  
 @Test  
 void checkUpright() throws Exception {  
 *assertTrue*(new Triangle(1, 1, 1).checkUpright());  
 *assertTrue*(new Triangle(2, 2, 2).checkUpright());  
 *assertFalse*(new Triangle(0, 0, 0).checkUpright());  
 *assertFalse*(new Triangle(-1,-1,-1).checkUpright());  
 *assertFalse*(new Triangle(-1,-2,-3).checkUpright());  
 *assertFalse*(new Triangle(1,2,3).checkUpright());

System.*out*.println("checkUpright() Test End");  
 }

@Test  
 void checkSameAndNinety() throws Exception {  
 *assertTrue*(new Triangle(1,1,Math.*pow*(2,0.5)).checkNinety());  
 *assertTrue*(new Triangle(1,1,1.41421356237).checkNinety());  
 *assertFalse*(new Triangle(0,0,3).checkNinety());  
 *assertFalse*(new Triangle(1,1,2).checkNinety());

System.*out*.println("checkNinety() Test End");  
 }  
  
 @Test  
 void checkNinety() throws Exception {  
 *assertTrue*(new Triangle(3,4,5).checkNinety());  
 *assertTrue*(new Triangle(5,12,13).checkNinety());  
 *assertFalse*(new Triangle(1,2,3).checkNinety());  
 *assertFalse*(new Triangle(1,1,7).checkNinety());  
 *assertFalse*(new Triangle(-1,-2,-3).checkNinety());

System.*out*.println("checkNinety() Test End");  
 }  
  
 @Test  
 void checkTwoSameLine() throws Exception {  
 *assertFalse*(new Triangle(3,3,6).checkTwoSameLine());  
 *assertFalse*(new Triangle(3,3,7).checkTwoSameLine());  
 *assertTrue*(new Triangle(3,3,5.5).checkTwoSameLine());

System.*out*.println("checkTwoSameLine() Test End");  
 }  
  
 @Test  
 void checkNormal() throws Exception {  
 *assertTrue*(new Triangle(3,4,5).checkNormal());  
 *assertTrue*(new Triangle(5,12,13).checkNormal());  
 *assertFalse*(new Triangle(1,2,3).checkNormal());  
 *assertFalse*(new Triangle(1,1,7).checkNormal());  
 *assertFalse*(new Triangle(-1,-2,-3).checkNormal());  
 *assertTrue*(new Triangle(1, 1, 1).checkNormal());  
 *assertTrue*(new Triangle(2, 2, 2).checkNormal());  
 *assertFalse*(new Triangle(0, 0, 0).checkNormal());  
 *assertFalse*(new Triangle(-1,-1,-1).checkNormal());  
 *assertFalse*(new Triangle(-1,-2,-3).checkNormal());  
 *assertFalse*(new Triangle(1,2,3).checkNormal());

System.*out*.println("checkNormal() Test End");  
 }  
  
 @Test  
 void checkNotTri() throws Exception {  
 *assertTrue*(new Triangle(0,0,0).checkNotTri());  
 *assertTrue*(new Triangle(-1,5,6).checkNotTri());  
 *assertTrue*(new Triangle(1,2,3).checkNotTri());  
 *assertTrue*(new Triangle(3,3,6).checkNotTri());

System.*out*.println("checkNotTri() Test End");  
 }  
  
 @Test  
 void pythagorean() throws Exception {  
 *assertTrue*(new Triangle(0,0,0).pythagorean(3.0,4.0,5.0));  
 *assertFalse*(new Triangle(0,0,0).pythagorean(3.0,4.0,4.9));  
 *assertFalse*(new Triangle(0,0,0).pythagorean(3.0,4.0,4.999));  
 *assertTrue*(new Triangle(0,0,0).pythagorean(3.0,4.0,4.99999999999));

System.*out*.println("pythagorean() Test End");  
 }  
  
}