**类继承和集合对象的使用**

Shape接口类：

**public** **interface** Shape {

**public** **double** getLength();

**public** **double** getArea();

**public** **boolean** validate();

**public** **void** init();

**public** **int** getId();

}

BaseShape抽象类：

**public** **abstract** **class** BaseShape **implements** Shape{

**protected** **double** length;

**protected** **double** area;

**protected** **int** id;

**static** **int** *counter* = 1;

**public** **void** init() {

**if**(validate()) { //判定是否合法形状

calculate(); //计算面积、周长

}

}

**public** **double** getLength() {

**return** length;

}

**public** **double** getArea() {

**return** area;

}

**public** **boolean** validate() {

**return** **true**;

}

**protected** **abstract** **void** calculate();

**public** **int** getId() {

**return** id;

}

}

Trangle三角形类：

**class** Trangle **extends** BaseShape{

**double** botton, leftSide, rightSide;

**public** Trangle(**double** a,**double** b,**double** c){

botton = a;

leftSide = b;

rightSide = c;

id = *counter*;

*counter* = *counter* +1;

}

**protected** **void** calculate(){

length = botton + leftSide + rightSide;

area = Math.*sqrt*((length/2)\*(length/2-botton)\*(length/2-rightSide)\*(length/2-leftSide));

}

**public** **boolean** validate(){

**if**(leftSide+rightSide<=botton || botton+rightSide<=leftSide || leftSide+botton<=rightSide) //判定是否合法三角形

**return** **false**;

**else**

**return** **true**;

}

}

Lader梯形类:

**class** Lader **extends** BaseShape{

**double** above, botton, height, leftSide, rightSide;

**public** Lader(**double** a,**double** b,**double** c,**double** d,**double** e){

botton = a;

leftSide = b;

above = c;

rightSide = d;

height = e;

id = *counter*;

*counter* = *counter* +1;

}

**protected** **void** calculate(){

length = botton + leftSide + above + rightSide;

area = (botton + above)\*height/2;

}

**public** **boolean** validate(){

**double** m=leftSide, n=rightSide, p=Math.*abs*(botton-above);

**if**(m+n>p && n+p>m && p+m>n) //判定是否合法梯形

**return** **true**;

**else**

**return** **false**;

}

}

EquicruralLader扩展梯形类:

**public** **class** EquicruralLader **extends** BaseShape{

**double** above, botton, height, yao;

**public** EquicruralLader(**double** a,**double** b,**double** c){

botton = a;

yao = b;

above = c;

id = *counter*;

*counter* = *counter* +1;

}

**protected** **void** calculate(){

length = botton + 2\*yao + above;

height = Math.*sqrt*(yao\*yao - ((above - botton)/2)\*((above - botton)/2));

area = (botton + above)\*height/2;

}

**public** **boolean** validate(){

**double** m=yao, n=yao, p=Math.*abs*(botton-above);

**if**(m+n>p && n+p>m && p+m>n) //判定是否合法梯形

**return** **true**;

**else**

**return** **false**;

}

}

EquicruralTrangle扩展三角形类:

**public** **class** EquicruralTrangle **extends** BaseShape{

**double** botton, yao;

**public** EquicruralTrangle(**double** a,**double** b){

botton = a;

yao = b;

id = *counter*;

*counter* = *counter* +1;

}

**protected** **void** calculate(){

length = botton + 2\*yao;

area = Math.*sqrt*((length/2)\*(length/2-botton)\*(length/2-yao)\*(length/2-yao));

}

**public** **boolean** validate(){

**if**(yao+yao<=botton) //判定是否合法三角形

**return** **false**;

**else**

**return** **true**;

}

}

Circle圆类：

**public** **class** Circle **extends** BaseShape{

**public** **double** r;

**public** Circle(**double** a){

r = a;

id = *counter*;

*counter* = *counter* +1;

}

**protected** **void** calculate(){

length = 2\*r\*3.14;

area = 3.14\*r\*r;

}

}

Test测试类：

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class Test{

public static void main(String[] args){

Shape s1 = new Trangle(6,8,10);

Shape s2 = new Trangle(1,2,3);

//第一个参数为底，第二个参数为高

Shape s3 = new EquicruralTrangle(12,12);

Shape s4 = new EquicruralTrangle(10,5);

Shape s5 = new Lader(1,15,12,15,13);

Shape s6 = new Lader(1,10,4,4,5);

//第一参数为上底，第二参数为下底，第三参数为腰

Shape s7 = new EquicruralLader(2,8,5);

Shape s8 = new EquicruralLader(2,16,5);

Shape s9 = new Circle(3);

Shape s10 = new Circle(5);

ArrayList<Shape> list = new ArrayList<Shape>();

System.out.println("过滤不合法的形状:");

if(!s1.validate())

System.out.println(" 1号对象是不合法的");

else{

s1.init();

list.add(s1);

}

if(!s2.validate())

System.out.println(" 2号对象是不合法的");

else{

s2.init();

list.add(s2);

}

if(!s3.validate())

System.out.println(" 3号对象是不合法的");

else{

s3.init();

list.add(s3);

}

if(!s4.validate())

System.out.println(" 4号对象是不合法的");

else{

s4.init();

list.add(s4);

}

if(!s5.validate())

System.out.println(" 5号对象是不合法的");

else{

s5.init();

list.add(s5);

}

if(!s6.validate())

System.out.println(" 6号对象是不合法的");

else{

s6.init();

list.add(s6);

}

if(!s7.validate())

System.out.println(" 7号对象是不合法的");

else{

s7.init();

list.add(s7);

}

if(!s8.validate())

System.out.println(" 8号对象是不合法的");

else{

s8.init();

list.add(s8);

}

if(!s9.validate())

System.out.println(" 9号对象是不合法的");

else{

s9.init();

list.add(s9);

}

if(!s10.validate())

System.out.println(" 10号对象是不合法的");

else{

s10.init();

list.add(s10);

}

System.out.println("按周长排序:");

Collections.sort(list,new Comparator<Shape>(){

public int compare(Shape o1, Shape o2) {

if(o1.getLength() > o2.getLength()){

return 1;

}

if(o1.getLength() == o2.getLength()){

return 0;

}

return -1;

}

});

for(int i = 0;i < list.size();i++){

System.out.print(list.get(i).getId()+" ");

}

System.out.println();

System.out.println("按面积排序:");

Collections.sort(list,new Comparator<Shape>(){

public int compare(Shape o1, Shape o2) {

if(o1.getArea() > o2.getArea()){

return 1;

}

if(o1.getArea() == o2.getArea()){

return 0;

}

return -1;

}

});

for(int i = 0;i < list.size();i++){

System.out.print(list.get(i).getId()+" ");

}

}

}

运行结果:

