上机作业2

**3-2**

注意：

1. 默认构造方法：Java会向没有给定任何构造方法的类中自动地加入一个没有参数且方法体为空的构造方法，通常将没有参数的构造方法称为默认构造方法。
2. 构造方法的特点：三个，书p56

代码：

**package** pracjava;

**public** **class** NewRectangle {

**private** **double** width; //私有变量，仅在该类中使用

**private** **double** height;

/\*默认构造方法

public NewRectangle{

}

\*/

//带参数构造方法

**public** NewRectangle(**double** x, **double** y){

width = x;

height = y;

}

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

**return** width\*height;

}

**public** **double** getPerimeter() {

**return** 2 \* (width+height);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

NewRectangle rect = **new** NewRectangle(3,4);

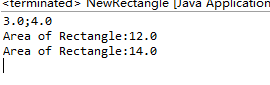
System.***out***.println(rect.width+";"+rect.height);

System.***out***.println("Area of Rectangle:"+ rect.getArea());

System.***out***.println("Area of Rectangle:"+ rect.getPerimeter());

}

}



**3-3**

注意：

1. 当前点与p点的问题：this.x and p.xget ; this.y and p.yget
2. 函数isInclude and isOverlapping是最费脑的…… 需要借助getPoint函数，这时注意点的取为：this.p.xget and this.p.yget
3. 判断一个点是否在矩形中，四个条件都不符合，则在矩形中。

p.xget() >= **this**.p.xget() + **this**.height

p.xget() <= **this**.p.xget()

p.yget()<=**this**.p.yget()

p.yget()>=**this**.p.yget() + **this**.width

1. 难点在（5）小问……关于是否在矩形中，只要判断一个矩形对角两个点是否在另一个矩形中即可。关于是否重叠，只要判断一个矩形的所有点，是否有一个点在另一个矩形中即可。
2. 这是一道很绕的题……

代码：

**package** pracjava;

**public** **class** NewRectangle\_ext {

**private** **double** width;

**private** **double** height;

Point p;

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Point p1 = **new** Point();

Point p2 = **new** Point(3,4);

System.***out***.println("distance:"+p2.distance(p1));

NewRectangle\_ext temp1 = **new** NewRectangle\_ext(2,3,1,1); //矩形1

NewRectangle\_ext temp2 = **new** NewRectangle\_ext(3,3,1,1); //矩形2

System.***out***.println(temp1.bPointIn(p2)); //判断点2是否在矩形1中存在

System.***out***.println(temp2.isInclude(temp1)); //判断矩形2中是否包含矩形1

System.***out***.println(temp2.isOverlapping(temp1)); //判断矩形2和矩形1是否有重合部分

}

**public** NewRectangle\_ext() {

**this**.width = 0;

**this**.height = 0;

}

**public** NewRectangle\_ext(**double** w,**double** h,**double** x,**double** y){

**this**.width = w;

**this**.height = h;

p = **new** Point(x,y);

}

**public** **boolean** bPointIn(Point p){

**if**(p.xget() >= (**this**.p.xget() + **this**.height)) {

**return** **false**;

}

**else** **if**(p.xget() <= **this**.p.xget()) {

**return** **false**;

}

**else** **if**(p.yget()<=**this**.p.yget()||(p.yget()>=(**this**.p.yget() + **this**.width))){

**return** **false**;

}

**else** {

**return** **true**;

}

}

**public** Point getPoint(**int** choose) {

Point p;

**if**(choose == 1) //左下角顶点

p = **new** Point(**this**.p.xget(),**this**.p.yget());

**else** **if**(choose == 2) //右下角顶点

p = **new** Point((**this**.p.xget()+**this**.height),**this**.p.yget());

**else** **if**(choose == 3) //右上角顶点

p = **new** Point((**this**.p.xget()+**this**.height),(**this**.p.yget()+**this**.width));

**else**

p = **new** Point(**this**.p.xget(),(**this**.p.yget()+**this**.width));

**return** p;

}

**public** **boolean** isInclude(NewRectangle\_ext temp) {

**if**(**this**.bPointIn(temp.getPoint(1))&&bPointIn(temp.getPoint(3)))

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isOverlapping(NewRectangle\_ext temp) {

**for**(**int** i=1;i<5;i++)

**if**(**this**.bPointIn(temp.getPoint(i)))

**return** **true**;

**return** **false**;

}

}

**class** Point{

**private** **double** x;

**private** **double** y;

//默认构造方法

**public** Point() {

x = 0;

y = 0;

}

//带参数构造方法

**public** Point(**double** x, **double** y) {

**this**.x = x;

**this**.y = y;

}

//函数

**public** **double** xget(){

**return** **this**.x;

}

//函数

**public** **double** yget(){

**return** **this**.y;

}

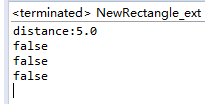
//求距离函数

**public** **double** distance(Point p) {

**return** Math.*sqrt*(Math.*pow*(Math.*abs*(**this**.x-p.xget()),2)+Math.*pow*(Math.*abs*(Math.*abs*(**this**.y-p.yget())), 2));

}

}



**3-9**

注意：

(1):参数向上转型是指把**子类的对象当作父类的对象看待**，所以在对父类的方法调用时，里面的参数为子类的对象。例如：c.ride(u)是对Cycle类中ride方法的调用，但参数属于子类Unicycle。

(2):重写方法就是重定义方法，与父类中不一样即可。

(3):难点处：书p81.向下转型是指将**一个父类类型的变量转换为一个子类类型的变量**，此时需要使用强制转换的语法。

格式：if(父类变量 instanceof 子类名)

((子类名)父类变量).子类方法();

代码：

**package** pracjava;

**public** **class** Cycle {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Cycle c = **new** Cycle();

Unicycle u = **new** Unicycle();

Bicycle b = **new** Bicycle();

Tricycle t = **new** Tricycle();

c.ride(u);//子类对象当作父类对象，向上转型

c.ride(b);

c.ride(t);

}

**public** **void** ride(Cycle c){

System.***out***.println("轮子的数量是:"+c.wheel());

**if**(c **instanceof** Unicycle) //向下转型，父类变量转换为子类变量

((Unicycle)c).balance();

**if**(c **instanceof** Bicycle)

((Bicycle)c).balance();

}

**public** **int** wheel() {

**return** 0;//返回车轮数量

}

}

**class** Unicycle **extends** Cycle{

**public** **int** wheel() {

**return** 1; //重写

}

**public** **void** balance() {

System.***out***.println("balance");

}

}

**class** Bicycle **extends** Cycle{

**public** **int** wheel() {

**return** 2;

}

**public** **void** balance() {

System.***out***.println("no balance");

}

}

**class** Tricycle **extends** Cycle{

**public** **int** wheel() {

**return** 3;

}

}

