# 实验3：对象和类

## 实验目的和要求

掌握类和对象的概念

掌握类的定义及创建对象的方法

掌握通过对象引用变量访问对象的方法

掌握使用对象成员访问操作符（.）来访问对象的数据和方法

能够区分对象引用变量与基本类型变量的不同

掌握实例成员与静态成员的区别与使用方法

掌握对象作为方法参数及返回值

在数组中存储和处理对象

## **二、实验内容与结果及分析**

### P305 9.1

原题：（矩形类Rectangle）遵照9.2节中Circle类的例子，设计一个名为Rectangle的类表示矩形。这个类包括：

两个名为width和height的double型数据域，它们分别表示矩形的宽和高。Width和height的默认值都为1.

创建默认矩形的无参构造方法。

一个创建width和height为指定值的矩形的构造方法。

一个名为getArea（）的方法返回这个矩形的面积。

一个名为getPerimeter（）的方法返回周长。

画出该类的UML图并实现这个类。编写一个测试程序，创建两个Rectangle对象——一个矩形的宽为4而高为40，另一个矩形的宽为3.5而高为35.9.按照这个顺序显示每个矩形的宽、高、面积和周长。

源代码：

**public class T1 {**

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

**//P305 9.1**

**Rectangle r0 = new Rectangle(4,40);**

**Rectangle r1 = new Rectangle(3.5,35.9);**

**System.out.println("第一个矩形的宽、高、面积、周长为：");**

**System.out.println(r0.wedth+","+r0.height+","+r0.getArea()+","+r0.getPerimeter());**

**System.out.println("第二个矩形的宽、高、面积、周长为：");**

**System.out.println(r1.wedth+","+r1.height+","+r1.getArea()+","+r1.getPerimeter());**

**}**

**}**

**class Rectangle**

**{**

**double wedth = 1;**

**double height = 1;**

**Rectangle(){}**

**Rectangle(double wedth, double height)**

**{**

**this.wedth = wedth;**

**this.height = height;**

**}**

**public double getArea()**

**{**

**double s;**

**s = wedth \* height;**

**return s;**

**}**

**public double getPerimeter()**

**{**

**double l;**

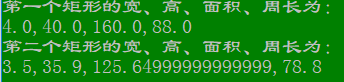
**l = 2\*(wedth +height);**

**return l;**

**}**

**}**

结果及截图：



### P306 9.2

原题：（股票类Stock）遵照9.2节中Circle类的例子，设计一个名为Stock的类。这个类包括：

一个名为symbol的字符串数据域表示股票代码。

一个名为name的字符串数据域表示股票名字。、一个名为previousClosingPrice的double型数据域，它储存的是前一日的股票值。

一个名为current Price的double型数据域，它储存的是当时的股票值。

创建一个名为个体Change Percent（）的方法，返回从previous Closing Price变化到currentPrice的百分比。

画出该类的UYML图并实现这个类编写一个测试程序，创建一个Stock对象，他的股票代码是ORCL，股票名字是Oracle Corporation，前一日收盘价是34.5.设置新的当前值为34.35，然后显示市值的变化的百分比。

源代码：

public class T2 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Stock s = new Stock("ORCL","Oracle Corporation");

s.setCurrentPrice(34.35);

System.out.println(s.getChangePercent());

}

}

class Stock

{

private String symbol;

private String name;

private double previousClosingPrice = 34.5;

private double currentPrice;

Stock(String symbol, String name)

{

this.symbol = symbol;

this.name = name;

}

public double getChangePercent()

{

double rate;

rate = (currentPrice-previousClosingPrice)/previousClosingPrice;

return rate;

}

public void setCurrentPrice(double currentPrice)

{

this.currentPrice = currentPrice;

}

}结果及截图：



### P309 9.6

原题：（秒表）设计一个名为StopWatch的类，该类包含：

具有访问器方法的私有数据域startTime和endTime。

一个无参构造方法，使用当前时间来初始化startTime。

一个名为start（）的方法，将startTime重设为当前时间。

一个名为stop（）的方法，将endTime设置为当前时间。

一个名为getElapsedTime（）的方法，以毫秒为单位返回秒表记录的流逝时间。

画出这个类的UML图并实现这个类。编写一个测试程序，用于测量使用选择排序对100000格尔数字进行排序的执行时间。

源代码：

**package** zuoye;

**import** java.util.Arrays;

**public** **class** T3 {

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

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

StopWatch t = **new** StopWatch();

**int**[] a = **new** **int**[100000];

**int** keyvalue;

**int** index;

**int** temp;

**for**(**int** i=0; i<a.length; i++)

{

a[i] = (**int**)Math.*random*()\*100000;

}

t.start();

**for**(**int** i=0; i<a.length; i++)

{

index = i;

keyvalue = a[i];

**for**(**int** j=i; j<a.length; j++)

{

**if**(a[j]<keyvalue)

{

index = j;

keyvalue = a[j];

}

}

temp = a[index];

a[index] = a[i];

a[i] = temp;

}

t.stop();

System.***out***.println(t.getElapsedTime());

}

}

**class** StopWatch

{

**private** **long** startTime;

**private** **long** endTime;

StopWatch()

{

startTime = System.*currentTimeMillis*();

}

**public** **long** getStartTime()

{

**return** startTime;

}

**public** **long** getEndTime()

{

**return** endTime;

}

**public** **void** start()

{

startTime = System.*currentTimeMillis*();

}

**public** **void** stop()

{

endTime = System.*currentTimeMillis*();

}

**public** **long** getElapsedTime()

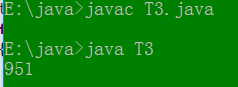
{

**return** endTime - startTime;

}

}

结果及截图：



### P307 9.9

原题：（几何：正n边形）在一个正n边形中，所有边的长度都相同，且所有角的度数都相同（即这个多边形是等边等角的）。设计一个名为RegularPolygon的类，该类包括：

一个名为n的int型的私有的数据域定义多边形的边数，默认为3。

一个名为side的double型私有数据域存储边的长度，默认值为1；

一个名为x的double型私有数据域定义多边形的中点的x坐标，默认值为0

一个名为y的double型私有数据域定义多边形的中点的y坐标，默认值为0

一个创建带默认值的正多边形的无参构造方法。

一个能创建带指定边数和边长度、中心在（0，0）的正多边形的构造方法。

一个能创建带指定边数和边长度、中心在（x，y）的正多边形的构造方法。

所有数据域的访问器和修改器。

一个返回多边形周长的方法getPerimeter（）。

一个返回多边形面积的方法gerArea（）。计算正多边形面积的公式为：

面积 = n\*s\*s/（4\*tan（π/n））；

源代码：

**public class T4 {**

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

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

**RegularPolygon r0 = new RegularPolygon();**

**RegularPolygon r1 = new RegularPolygon(6,4);**

**RegularPolygon r2 = new RegularPolygon(10,4,5.6,7.8);**

**System.out.println("第一个正多边形的周长和面积为：");**

**System.out.println(r0.getPerimeter()+","+r0.getArea());**

**System.out.println("第二个正多边形的周长和面积为：");**

**System.out.println(r1.getPerimeter()+","+r1.getArea());**

**System.out.println("第三个正多边形的周长和面积为：");**

**System.out.println(r2.getPerimeter()+","+r2.getArea());**

**}**

**}**

**class RegularPolygon**

**{**

**private int n = 3;**

**private double size = 1;**

**private double x = 0;**

**private double y = 0;**

**RegularPolygon(){}**

**RegularPolygon(int n,double size)**

**{**

**this.n = n;**

**this.size = size;**

**}**

**RegularPolygon(int n,double size,double x,double y)**

**{**

**this.n = n;**

**this.size = size;**

**this.x = x;**

**this.y = y;**

**}**

**public void setN(int n)**

**{**

**this.n = n;**

**}**

**public void setSize(double size)**

**{**

**this.size = size;**

**}**

**public void setZuoBiao(double x,double y)**

**{**

**this.x = x;**

**this.y = y;**

**}**

**public int getN()**

**{**

**return n;**

**}**

**public double getSize()**

**{**

**return size;**

**}**

**public double getX()**

**{**

**return x;**

**}**

**public double getY()**

**{**

**return y;**

**}**

**public double getPerimeter()**

**{**

**double l;**

**l = n\*size;**

**return l;**

**}**

**public double getArea()**

**{**

**double s;**

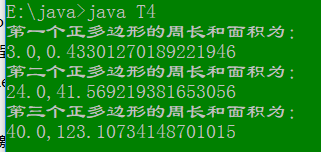
**s = n\*size\*size/(4\*Math.tan(Math.PI/n));**

**return s;**

**}**

**}**

结果及截图：



### P308 9.13

原题：设计一个名为Location的类，定位二维数组中的最大值及其位置。这个类包括公共的数据域row、column和maxValue，二维数组中的最大值及其下标用int型的row和column以及double型的maxValue储存。

编写下面的方法，返回一个二维数组中最大值的位置。

Public static Location location（double[][] a）

返回值是一个Location的实例。编写一个测试程序，提示用户输入一个二维数组，然后显示这个数组最大元素的位置。下面是一个运行示例：

Enter the number of rows and columns in the array：3 4

Enter the array：

23.5 35 2 10

4.5 3 45 3.5

35 44 5.5 9.6

源程序：

**import** java.util.Scanner;

**public** **class** T5 {

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

//P308 9.13

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Enter the number of rows and columns in the array:");

**int** row = in.nextInt();

**int** column = in.nextInt();

**double**[][] a = **new** **double**[row][column];

System.***out***.println("Enter the array:");

**for**(**int** i=0; i<a.length; i++)

{

**for**(**int** j=0; j<a[i].length; j++)

{

a[i][j] = in.nextDouble();

}

}

Location l = **new** Location();

System.***out***.println("The location of the largest element is "+l.*locatelargest*(a).*maxValue*+" at"+"("+l.*locatelargest*(a).*row*+","+l.*locatelargest*(a).*column*+")");

}

}

**class** Location

{

**public** **static** **int** *row*;

**public** **static** **int** *column*;

**public** **static** **double** *maxValue*;

**public** **static** Location locatelargest(**double**[][] a)

{

*maxValue* = a[0][0];

Location l = **new** Location();

**for**(l.*row*=0; l.*row*<a.length; l.*row*++)

{

**for**(l.*column*=0;l.*column*<a[l.*row*].length; l.*column*++)

{

**if**(*maxValue*<a[l.*row*][l.*column*])

{

*maxValue* = a[l.*row*][l.*column*];

**break**;

}

}

}

**return** l;

}

**public** **int** getRow()

{

**return** *row*;

}

**public** **int** getColumn()

{

**return** *column*;

}

}结果及截图：

