**上海电力学院**

**《Java语言程序设计》实验**



**（2018/2019 学年第2学期）**

**实验名称**实验2 类与对象

**院 （系） 计算机科学与技术**

**专 业 计算机科学与技术**

**班 级 2018053**

**学 号 20181389**

**姓 名 洪勇**

**任课老师 张 维**

2.1实验目的

（1) 了解类的成员设计  
（2) 掌握类与对象的关系

（3) 理解静态方法与实例方法的使用差异；

2.2 知识要点

**2.2.1 类**

   作为java程序的基本要素，类由成员变量（也称属性）和成员方法组成。属性包括常量和变量，方法则包括构造方法和一般方法，在定义类时,它们习惯上按如下次序排列:  
  
class Order { // final 属性  
  
// 属性  
  
// 构造方法  
// 方法  
}  
类的属性一般定义为私有的，而将类的方法定义为public。

**2.2.2  对象创建与构造方法**

类是创建对象的模板，而对象是类的实例。对象是一组属性和方法的封装体。创建对象用到保留字new。

格式为： 引用变量 = new 构造方法（参数）

有关构造方法有以下几点说明：

（1）创建对象时将用到构造方法；系统自动调用参数匹配的构造方法为对象初始化。

（2）构造方法的名称必须与类名同名，构造方法没有返回类型;

（3）一个类可提供多个构造方法，这些方法的参数不同。

（4）如果程序中未提供构造方法，系统自动提供空的无参构造方法。

**2.2.3  对象引用变量与对象关系**

对象创建后，将对象的引用赋值给某个引用变量，就可以通过该变量访问对象的成员属性和方法。

成员的调用通过“.”运算符实现，格式为：

对象名.属性

对象名.方法（参数）

以下结合实例介绍对象引用变量与对象的关系。

class Point {

int x,y;  //属性变量

public Point(int x1,int y1) {  //构造方法

    x=x1;  //将参数值赋值给对象的属性变量

    y=y1;

}

public static void main(String args[]) {

    Point p1=new Point(2,3);  //创建点（2，3）用p1引用

    Point p2=new Point(3,5);  //创建点（3，5）用p2引用

    Point p3=p1;             //p3和p1代表同一对象

    p1.x=18;                 //修改p1的x属性值

    System.out.println(p3.x);   //输出 p3的x属性值

    p1=p2;                   //p1“叛变”，改为和p2指向同一点

    System.out.println(p1.x);

}

}

【运行结果】

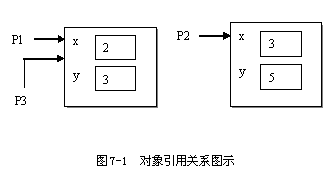
  18

  3

【有关说明】

（1）main方法中首先创建两个对象，用带阴影的方框表示，分别用p1和p2两个引用变量存储其引用地址。

（2）语句Point p3=p1;的作用是定义一个Point类型的引用变量p3和p1值向同一对象；如图1所示。



（3）语句p1.x=8；的作用是修改p1所引用对象的x1值，由于p1和p3是“代表”同一对象，因此，输出p3.x和p1.x是一样的,也就是18.

(4)  语句p1=p2将p1改为指向和p2所指的对象,这样,再输出 p1.x则结果为3。

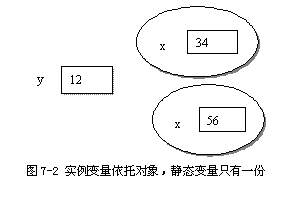
【重要提示】 对象和对象引用变量是两个不同的概念，对象引用变量中存放的是对象的地址。通常访问对象需要借助对象引用变量，将对象的地址赋值给对象引用变量，则该引用变量就“代表”这个对象，但是引用变量与对象并不是固定的关系，通过更改引用变量的值，可让其“代表”另一对象，也可以通过引用变量之间的赋值让多个引用变量指向同一对象。如果一个对象没有任何引用变量指向它，则要访问该对象就再也不可能，Java垃圾回收器将自动回收该对象占用的内存。

**2.2.4 静态空间与实例空间**

带有static修饰的属性称为静态属性或称类变量；带有static修饰的方法称为静态方法，在静态方法内可以直接访问静态属性或调用其他静态方法，静态属性在分配空间时与对象无关，静态空间的引用是与类关联，一个类的静态属性成员只有一份，访问静态成员可以通过类名（在自己类中也可以直接访问），也可以通过对象，静态属性是共享的，该类的任何对象均可访问它。

【思考】静态属性外层包裹的是类，因此必须指明是访问哪个类的静态成员，如果是在自己类中则可以省略，由于对象是由类创建，自然对象能标识其所属类，因此也允许用对象做前缀访问静态属性。非静态属性也称实例变量，也就是说，要通过实际的对象才能访问，其外层包裹的是对象，不同对象其实例变量的值通常是不同的。

例如，以下类中y是属于类的属性变量，而x是属于对象的实例变量，y是依托类的，在类装载时将给其分配空间并赋值，整个类只有一个y；而x只有创建了具体对象才存在，不同对象有各自的x属性，如图7-2所示。



class demo {

int x;

Static int y=12;

 }

【重要提示】对象方法中可以访问静态空间的方法或属性；反之，在静态方法中绝对不允许在没有指定对象的情况下访问属于对象的成员属性或方法。

2.3 样例程序

**样例1:** 创建5个学生对象给一个学生数组赋值，每个学生属性有：学号、姓名、年龄

（1）将学生按学号排序输出；  
（2）给所有学生年龄加1；  
（3）统计大于20岁的学生人数。

【参考程序】

public class Student{

    int num;

    int age;

    String name;

    public String toString( ) {

       String s = "学号:" + num + "，";

       s += "姓名:" + name + "，";

       s += "年龄:" + age;

       return s;

    }

    public Student(int Num,int Age,String Name) {

       num = Num;

       age = Age;

       name = Name;

    }

   public static void main(String args[]) {

       Student s1 = new Student(3,18,"张三");

       Student s2 = new Student(1,21,"小路");

       Student s3 = new Student(33,20,"John");

       Student s4 = new Student(13,20,"Lucy");

       Student s5 = new Student(8,17,"Jack");

       Student s[] = {s1,s2,s3,s4,s5};

       System.out.println("班级学生名单如下：");

       output(s);  //第1次调用output方法输出数组

       /\* 将学生按学号排序 \*/

       for (int i=0;i<s.length-1;i++)

         for (int j=i+1;j<s.length;j++)

           if (s[i].num>s[j].num) {

               Student tmp=s[i];

               s[i]=s[j];

               s[j]=tmp;

            }

       System.out.println("按学号由小到大排序...");

       output(s);   //第2次调用output方法输出数组

for (int i=0;i<s.length;i++) //将所有学生年龄加1

            s[i].age++;

       System.out.println("所有学生年龄加1后...");

       output(s);  //第3次调用output方法输出数组

       /\* 以下统计大于20岁的学生个数 \*/

       int count=0;

       for (int i=0;i<s.length;i++)

          if (s[i].age>=20)

             count++;

       System.out.println("大于20岁人数是：" + count);

}

 /\* 以下方法输出学生数组的所有元素  \*/

     static void  output(Student s[]) {

        for (int i=0;i<s.length;i++)   
    System.out.println(s[i]);

      }

}

【说明】本例包括了对象访问的典型操作，创建对象调用构造方法，直接输出对象将调用对象的toString()方法，通过数组管理学生对象，这里演示如何通过数组元素访问对象的编程技巧。由于要多次输出学生数组，所以程序中编写了一个output方法。

【运行结果】

班级学生名单如下：

学号:3，姓名:张三，年龄:18

学号:1，姓名:小路，年龄:21

学号:33，姓名:John，年龄:20

学号:13，姓名:Lucy，年龄:20

学号:8，姓名:Jack，年龄:17

按学号由小到大排序...

学号:1，姓名:小路，年龄:21

学号:3，姓名:张三，年龄:18

学号:8，姓名:Jack，年龄:17

学号:13，姓名:Lucy，年龄:20

学号:33，姓名:John，年龄:20

所有学生年龄加1后...

学号:1，姓名:小路，年龄:22

学号:3，姓名:张三，年龄:19

学号:8，姓名:Jack，年龄:18

学号:13，姓名:Lucy，年龄:21

学号:33，姓名:John，年龄:21

大于20岁人数是：3

【编程技巧】

(1) 如何根据构造方法的要求创建对象；

(2) 如何给对象数组赋值。

(3) 如何通过数组元素操作访问各对象的属性。

(4) 输出对象会自动调用对象的toString()方法。

**样例2：**在以下复数类的基础上扩充乘法运算。  
   public class ComplexNumber {   
     private double x, y; //私有类变量   
     public ComplexNumber(double real, double imaginary) {   
        x = real; y = imaginary;  
     }   
     public String toString() { return "{" + x + "," + y + "}"; }   
     public static ComplexNumber add(ComplexNumber a, ComplexNumber b) {  
        return new ComplexNumber(a.x + b.x, a.y + b.y);  
     }   
     public ComplexNumber add(ComplexNumber a) {

return new ComplexNumber(x + a.x, y+a.y);

}   
}

 编写一个测试程序调用以上复数类，实现以下求解调用，由命令行提供数据：   
    命令“java test  + 2 3 4 5 ”意思是求 (2,3i)与(4,5i)之和.  
    命令“java test  \* 4 6 3 5 ”意思是求 (4,6i)与(3,5i)之积.

【参考程序】

public class ComplexNumber {

  private double x, y;   //复数的实部、虚部

  /\*\* 构造函数 \*/

   public ComplexNumber(double real, double imaginary) {

     this.x = real;

     this.y = imaginary;

   }

  public String toString() { return "{" + x + "," + y + "}"; }

/\*\* 两复数相加 \*/

  public static ComplexNumber add(ComplexNumber a, ComplexNumber b) {

     return new ComplexNumber(a.x + b.x, a.y + b.y);

   }

/\*\* 本复数加上一个复数 \*/

  public ComplexNumber add(ComplexNumber a) {

      return new ComplexNumber(this.x + a.x, this.y+a.y);

   }

/\*\* 两复数相乘 \*/

  public static ComplexNumber multiply(ComplexNumber a, ComplexNumber b) {

    return new ComplexNumber(a.x\*b.x - a.y\*b.y, a.x\*b.y + a.y\*b.x);

  }

/\*\* 本复数乘上一个复数 \*/

  public ComplexNumber multiply(ComplexNumber a) {

      return new ComplexNumber(x\*a.x - y\*a.y, x\*a.y + y\*a.x);

   }

  public static void main(String args[]) {

     ComplexNumber x,y,z;

    /\* 以下根据命令行参数构造两复数 \*/

x=new ComplexNumber(Double.parseDouble(args[1]), Double.parseDouble(args[2]));

    y=new ComplexNumber(Double.parseDouble(args[3]), Double.parseDouble(args[4]));

    if (args[0].equals("+"))  //判断是做加法还是乘法

        z= add(x,y);　 //用静态方法

    else

        z=x.multiply(y);  //用实例方法

     System.out.println("result="+z);

   }

 }

【说明】访问静态方法不依赖对象，使用静态方法通常用类名做前缀，例如：ComplexNumber.add(x,y)，在ComplexNumber类的内部可以省略直接写add(x,y)，我们也可以用对象做前缀访问静态方法，例如写成x.add(x,y)，因为静态方法是共享的。实例方法则必须通过对象进行访问。

【编程技巧】

(1) 区分静态方法和实例方法的调用特点。静态方法执行复数相加要两个复数作为参数，而实例方法只要一个复数作为参数，因为调用实例方法时必须通过一个复数作前缀才能调用，也即将该复数与参数中的复数相加。

(2) 两个字符串的比较要用equals方法。

2.4 上机练习

      **基本题**

1. 编写一个类实现银行帐户的概念。包括的属性有：帐号、储户姓名、地址、存款余额，包括的方法有：存款、取款、查询、计算利息、累加利息等。

**（源代码）**

**package** java实验二.experment2\_1;

**import** java.util.Scanner;

**class** Admin{

**int** num;

**int** n\_money;

String name;

String address;

Admin(**int** Num,**int** N\_money,String Name,String Address) {

num=Num;

n\_money=N\_money;

name=Name;

address=Address;

}

**float** input(**float** s\_money) {

System.***out***.println("您要存"+s\_money+"元");

n\_money+=s\_money;

System.***out***.println("您的账户余额为"+n\_money+"元");

**return** n\_money;

}

**float** output(**float** o\_money) {

System.***out***.println("您要取"+o\_money+"元");

n\_money-=o\_money;

System.***out***.println("您的账户余额为"+n\_money+"元");

**return** n\_money;

}

**void** sure() {

System.***out***.println("您的账户余额为"+1.00000000000\*n\_money+"元");

}

**double** interest(**double** rate) {

**double** i,j;

i=(**double**)(0.01\*rate)\*n\_money;

j=n\_money;

j+=i;

System.***out***.println("您的利息为"+i);

System.***out***.println("您的账户余额为"+1.00000000000\*j+"元");

**return** j;

}

**double** re\_interest(**double** rate,**int** count) {

**double** f\_interest=n\_money;

**for**(**int** i=0;i<count;i++) {

f\_interest\*=(**double**)(1+0.01\*rate);

}

System.***out***.println("您的利息为"+(f\_interest-n\_money));

System.***out***.println("您的账户余额为"+1.00000000000\*f\_interest+"元");

**return** f\_interest;

}

}

**public** **class** Java20181389\_洪勇\_2\_1 {

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

Admin s1;

s1=**new** Admin(1,2,"hy","yh");

Admin s2;

s2=**new** Admin(2,2,"hy","yh");

Admin s3;

s3=**new** Admin(3,2,"hy","yh");

Admin s[]= {s1,s2,s3};

System.***out***.println("请输入你的账户");

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

**int** n\_num=scanner.nextInt();

**int** target=0;

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

**if**(s[i].num==n\_num)

target=i;

}

**if**(target==0)

System.***out***.println("账户输入错误");

**else**

s[target].sure();

System.***out***.println("请输入您想操作的指令");

System.***out***.println("1.存入"+"\t"+"2.取出"+"\t"+"3.计算每日利息"+"\t"+"4.计算累计利息"+"\t"+"5.查询");

String instruction=scanner.next();

**switch**(instruction) {

**case** "1":System.***out***.println("您要存？元");

**float** s\_money=scanner.nextFloat();

s[target].input(s\_money);

**break**;

**case** "2":System.***out***.println("您要取？元");

**float** o\_money=scanner.nextFloat();

s[target].output(o\_money);

**break**;

**case** "3":System.***out***.println("您的活期利率为？%");

**double** rate=scanner.nextDouble();

s[target].interest(rate);

**break**;

**case** "4":System.***out***.println("您的活期利率为？%");

**double** re\_rate=scanner.nextDouble();

System.***out***.println("您存储的天数为？天");

**int** count=scanner.nextInt();

s[target].re\_interest(re\_rate, count);

**break**;

**case** "5":System.***out***.println(s[target]);

**break**;

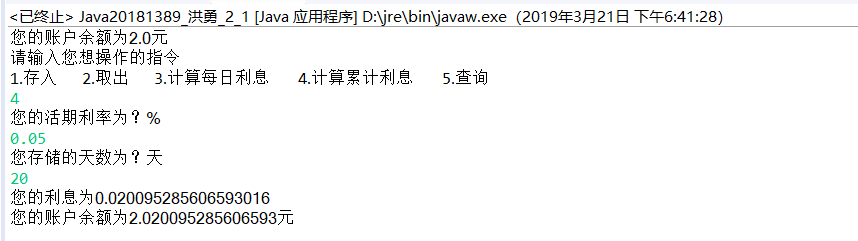
**default**:System.***out***.println("您输入指令有误！");

}

}

}

**（运行结果截图）**



2）编写一个类实现桌子的概念，包括的属性有长、宽、高和颜色，包括的方法有：计算体积、显示桌子信息、设置桌子颜色及长宽高。

**（源代码）**

**package** java实验二.experment2\_2;

**import** java.util.Scanner;

**class** Desk{

**double** length=0;

**double** width=0;

**double** highth=0;

String color=**null**;

Desk(**double** Length,**double** Width,**double** Highth,String Color){

length=Length;

width=Width;

highth=Highth;

color=Color;

}

**double** volume(**double** length,**double** width,**double** highth) {

**double** volume=(**double**)length\*width\*highth;

**return** volume;

}

**void** shows() {

System.***out***.println("Length:"+length);

System.***out***.println("Width:"+width);

System.***out***.println("Highth:"+highth);

System.***out***.println("Volume:"+volume(length,width,highth));

System.***out***.println("color:"+color);

}

}

**public** **class** Java20181389\_洪勇\_2\_1 {

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

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

Desk Desk=**new** Desk(0,0,0,"");

System.***out***.println("请输入桌子的长度");

Desk.length=scanner.nextDouble();

System.***out***.println("请输入桌子的宽度");

Desk.width=scanner.nextDouble();

System.***out***.println("请输入桌子的高度");

Desk.highth=scanner.nextDouble();

System.***out***.println("请输入桌子的颜色");

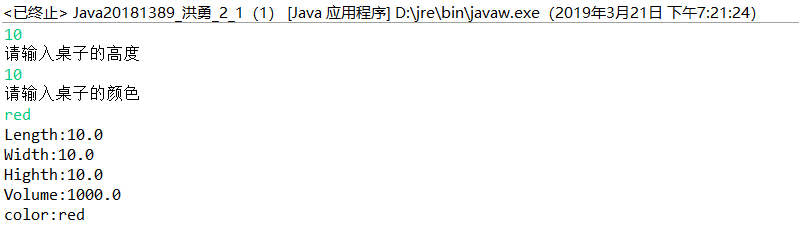
Desk.color=scanner.next();

Desk.shows();

}

}

**（运行结果截图）**



3）编写一个类实现一个圆，属性包括圆心、半径，方法主要包括显示的圆的信息、求面积等。圆心是Point类的对象，有两个属性x,y，代表横纵坐标。

**（源代码）**

**package** java实验二.experment2\_3;

**import** java.util.Scanner;

**class** Circle{

**public** **double** x,y,radius;

**public** **double** acre;

**public** **void** setX(**double** x) {

**this**.x=x;

}

**public** **double** getX() {

**return** x;

}

**public** **void** setY(**double** y) {

**this**.y=y;

}

**public** **double** getY() {

**return** y;

}

**public** **void** setRadius(**double** radius) {

**if**(radius<0)

**this**.radius=0;

**else**

**this**.radius=radius;

}

**public** **double** getRadius() {

**return** radius;

}

**public** **double** Acre(**double** radius) {

**double** acre=3.14\*radius\*radius;

**this**.acre=acre;

**return** acre;

}

**public** **void** shows() {

System.***out***.println("x:"+x);

System.***out***.println("y:"+y);

System.***out***.println("r:"+radius);

System.***out***.println("acre:"+acre);

}

}

**public** **class** Java20181389\_洪勇\_2\_3 {

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

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

System.***out***.println("请输入圆心坐标");

System.***out***.println("x:");

**double** x=scanner.nextDouble();

System.***out***.println("y:");

**double** y=scanner.nextDouble();

Circle Circle=**new** Circle();

Circle.setX(x);

Circle.getX();

Circle.setY(y);

Circle.getY();

System.***out***.println("请输入圆的半径");

**double** r=scanner.nextDouble();

Circle.setRadius(r);

Circle.getRadius();

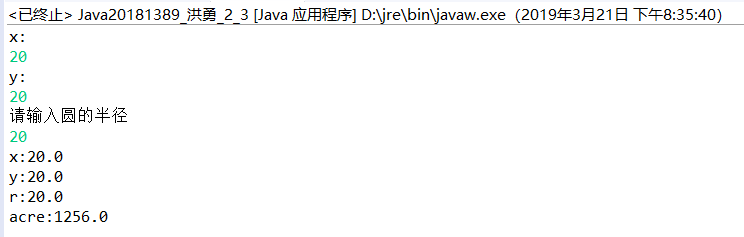
Circle.Acre(r);

Circle.shows();

}

}

**（运行结果截图）**



4) 将以下程序补充完整,调试分析

public class Java\_3 {  
      //点的坐标  
      int x,y;  
      public Java\_3() {}  
       //\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*  
      public Java\_3(int x,int y){\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}  
      //\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*  
      public Java\_3(Java\_3 p){\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}  
  
     //以对象的形式返回当前点的位置  
     public Java\_3 getLocation(){  
       //\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*  
        Java\_3 p=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //实例化一个Java\_3对象p,其坐标是(x,y)   
        //\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*  
        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //返回对象p  
     }  
  
     //返回点的横坐标  
     public int getX(){return x;}  
  
     //返回点的纵坐标  
     public int getY(){return y;}  
  
    //把当前点移到新的位置(x，y)上  
     public void move(int x,int y){this.x = x;this.y = y;}  
  
    //以(x，y)的格式返回点的位置  
     public String toString(){return "("+x+","+y+")";}  
  
     //在原有坐标上分别增加x和y   
     public void translate(int x,int y){this.x += x;this.y += y;}  
  
     public static void main(String args[]){  
       //\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*  
        Java\_3 p=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //生成一个对象(5,5)  
        System.out.println("x="+ p.x+" y="+ p.y);   
        System.out.println("Location is"+ p.toString()); //以(x,y)的方式打印坐标的位置  
       //\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*  
        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //在原有位置上增加(3，4)  
        System.out.println("x="+ p.x+" y="+ p.y); //打印横坐标和纵坐标的值  
        System.out.println("Location is"+ p.toString()); //以(x,y)的方式打印坐标的位置  
     }  
  }

**package** java实验二.experment2\_4;

**public** **class** Java20181389\_洪勇\_2\_4 {

//点的坐标

**int** x,y;

**public** Java20181389\_洪勇\_2\_4() {}

//\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*

**public** Java20181389\_洪勇\_2\_4(**int** x,**int** y){

**this**.x=x;

**this**.y=y;

}

//\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*

**public** Java20181389\_洪勇\_2\_4(Java20181389\_洪勇\_2\_4 p){

x=p.x;

y=p.y;

}

//以对象的形式返回当前点的位置

**public** Java20181389\_洪勇\_2\_4 getLocation(){

//\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*

Java20181389\_洪勇\_2\_4 p=**new** Java20181389\_洪勇\_2\_4();

p.x=**this**.x;

p.y=**this**.y;//实例化一个Java\_3对象p,其坐标是(x,y)

//\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*

**return** p; //返回对象p

}

//返回点的横坐标

**public** **int** getX(){**return** x;}

//返回点的纵坐标

**public** **int** getY(){**return** y;}

//把当前点移到新的位置(x，y)上

**public** **void** move(**int** x,**int** y){**this**.x = x;**this**.y = y;}

//以(x，y)的格式返回点的位置

**public** String toString(){**return** "("+x+","+y+")";}

//在原有坐标上分别增加x和y

**public** **void** translate(**int** x,**int** y){**this**.x += x;**this**.y += y;}

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

//\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*

Java20181389\_洪勇\_2\_4 p=**new** Java20181389\_洪勇\_2\_4(5,5); //生成一个对象(5,5)

System.***out***.println("x="+ p.x+" y="+ p.y);

System.***out***.println("Location is"+ p.toString()); //以(x,y)的方式打印坐标的位置

//\*\*\*\*\*\*\*\*\*Found\*\*\*\*\*\*\*\*

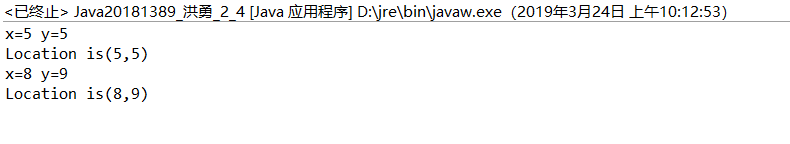
p.translate(3, 4);; //在原有位置上增加(3，4)

System.***out***.println("x="+ p.x+" y="+ p.y); //打印横坐标和纵坐标的值

System.***out***.println("Location is"+ p.toString()); //以(x,y)的方式打印坐标的位置

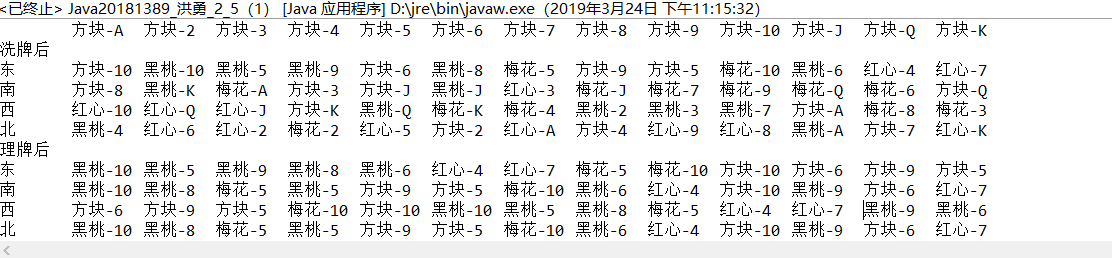
}

}



      **提高题**

设计一个扑克洗牌算法。将52张牌（不包括大、小王）用对象的形式表示，每张牌包括花色和大小。大小按如下序号排列：（2、3、4、5、6、7、8、9、10、J、Q、K、A）， 花色为： ♠ 、♣、♥、♦ ，首先将扑克按顺序存放到一个一维数组中，然后按洗牌算法进行洗牌,最后，按东、南、西、北进行分发，显示扑克时将同一花色放在一起。



**package** java实验二.experment2\_6;

**class** Card {

**private** String color;

**private** String num;

**public** **void** setcolor(String color) {

**this**.color=color;

}

**public** String getcolor() {

**return** color;

}

**public** **void** setnum(String num) {

**this**.num=num;

}

**public** String getnum() {

**return** num;

}

**public** String toString() {

**return** "\t"+color+"-"+num;

}

}

**public** **class** Java20181389\_洪勇\_2\_5 {

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

**final** **int** N=52,M=13;

Card[] card=**new** Card[N];

Card[] East=**new** Card[M];

Card[] South=**new** Card[M];

Card[] West=**new** Card[M];

Card[] North=**new** Card[M];

String[] color= {"黑桃","红心","梅花","方块"};

String[] num= {"A","2","3","4","5","6","7","8","9","10","J","Q","K"};

**for**(**int** i=0;i<N;i++) {

card[i]=**new** Card();

card[i].setcolor(color[i/13]);

card[i].setnum(num[i%13]);

}

**for**(**int** i=0;i<M;i++)

East[i]=**new** Card();

**for**(**int** i=0;i<M;i++)

South[i]=**new** Card();

**for**(**int** i=0;i<M;i++)

West[i]=**new** Card();

**for**(**int** i=0;i<M;i++)

North[i]=**new** Card();

System.***out***.println("洗牌前");

**for**(**int** i=0;i<N;i++) {

System.***out***.print(card[i].toString());

**if**((i+1)%13==0)

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

}

**for**(**int** i=0;i<N;i++) {

**int** index=(**int**)(Math.*random*()\*100000)%52;

String exchange=card[index].getcolor();

card[index].setcolor(card[i].getcolor());

card[i].setcolor(exchange);

String re\_exchange=card[index].getnum();

card[index].setnum(card[i].getnum());

card[i].setnum(re\_exchange);

}

System.***out***.println("洗牌后");

String[] position= {"东","南","西","北"};

**for**(**int** i=0;i<N;i++) {

**if**(i%13==0)

System.***out***.print(position[i/13]);

**if**(i>=0&&i<=12)

East[i]=card[i];

**else** **if**(i>=13&&i<=25)

South[i-13]=card[i];

**else** **if**(i>=26&&i<=38)

West[i-26]=card[i];

**else**

North[i-39]=card[i];

System.***out***.print(card[i].toString());

**if**((i+1)%13==0)

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

}

System.***out***.println("理牌后");

System.***out***.print(position[0]);

**for**(**int** j=0;j<4;j++) {

**for**(**int** k=0;k<M;k++) {

**if**(East[k].getcolor().equals(color[j]))

System.***out***.print(East[k].toString());

}

}

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

System.***out***.print(position[1]);

**for**(**int** j=0;j<4;j++) {

**for**(**int** k=0;k<M;k++) {

**if**(South[k].getcolor().equals(color[j]))

System.***out***.print(East[k].toString());

}

}

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

System.***out***.print(position[2]);

**for**(**int** j=0;j<4;j++) {

**for**(**int** k=0;k<M;k++) {

**if**(West[k].getcolor().equals(color[j]))

System.***out***.print(East[k].toString());

}

}

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

System.***out***.print(position[3]);

**for**(**int** j=0;j<4;j++) {

**for**(**int** k=0;k<M;k++) {

**if**(South[k].getcolor().equals(color[j]))

System.***out***.print(East[k].toString());

}

}

}

}

2.5 思考练习

1) 以下程序的运行结果为：

public class My{  
   int value;  
   public static void main(String args[]) {  
       My x=new My();  
       if (x==null)  
           System.out.println("No Object");  
        else  
           System.out.println(x.value);       
     }  
}  
A. 0     B. 1      C. No Object     D. 编译错误    E. null

A

（2）以下程序的运行结果为：

public class A {   
  static int k=3;  
  public static void main(String[] args) {   
    int k=4;  
    A  x1=new A();  
    x1.k++;  
    A  x2=new A();  
    x2.k++;  
    k++;  
    System.out.println(x1.k);   
  }  
}   
A. 3     B. 4     C.5      D.6     E.7

C

3） 编译和运行以下程序结果为：  
public class A {   
  static int k=3;  
  static int m;  
  public static void main(String[] args) {  
     k++;  
     if (m==0)   
       System.out.println(k);   
     else  
       System.out.println(B.k);   
     k++;  
  }  
 }  
 class B {  
     static int k=6;  
 }   
A. 3     B. 4     C.5      D.编译错误     E.6

B

4）编译和运行以下程序结果为：

    1:  public class Q21  {

    2:      int maxElements;

    3:      void Q21()  {

    4:          maxElements = 100;

    5:          System.out.println(maxElements);

    6:      }

    7:     Q21(int i)  {

    8:          maxElements = i;

    9:          System.out.println(maxElements);

    10:     }

    11:     public static void main(String[] args) {

    12:          Q21 a = new Q21();

    13:          Q21 b = new Q21(999);

    14:      }

    15:  }

    A. 输出100 和 999.

    B. 输出999 和 100.

    C. 第2行出现编译错误,变量 maxElements未初始化.

D. 12行出现编译错误.

D

5）以下的程序的调试结果为

public class Scope{

   int i;

   public static void main(String argv[]){

      Scope s = new Scope();

      s.amethod();

   }

   public static void amethod(){

     System.out.println(i);

   }

 }

A. 输出结果为：0

B. 无输出

C. 编译错误

D. 输出null

C

### §2.6实验总结

将程序编写和测试过程中出现的各种问题（需要的话可以截图）、解决方法详细总结在这部分。

1. 类要写在最外面，公用类要重新用别的文件写；
2. 声明没有给空间，只有当new之后才能正式使用内存空间。