第 4 章 类与对象

4.1 编程语言的发展

面向对象语言的特点：封装、继承和多态。
三角形有顶点、边、角、周长、面积、重心、垂心、外心、内心、旁心、内切圆、外接圆、旁切圆、欧拉圆……通过三角形的三条边的长度求出周长和面积……
特殊三角形有锐角三角形、直角三角形、钝角三角形、等腰三角形、等边三角形……
通过直角三角形的任意两条边的长度求出另外一条边的长度……
给出锐角三角形最大角的类别、给出直角三角形最大角的类别……

4.2 类

1. 面向对象语言中的类和对象分别是面向过程语言中的类型和变量的推广。

2. 类声明
   类头 {
       类体
   }

3. 类头和类体
（1）类头的格式为“class 类名”。
（2）类体包含属性和行为，即成员变量声明和方法定义。

4. 类名
   若首字符为字母，则该字母尽量用大写。

5. 成员变量
（1）也称域变量，有效性与声明的位置无关，但提倡在类体开头声明。
（2）可在声明时进行初始化，但在方法外不能进行赋值。

6. 方法
（1）方法头 {
        方法体
    }
（2）方法头的格式为“类型 方法名(形式参数表)”。若为构造方法，则开头无“类型”，“方法名”即为“类名”。
（3）方法体包含局部变量声明和语句。
（4）局部变量的有效性与声明的位置有关。
（5）形式参数相当于在方法体开头声明的局部变量。
   p.53 代码 2

```
class Lader {
    float above;      //梯形的上底(变量声明)
    float bottom;     //梯形的下底(变量声明)
    float height;     //梯形的高(变量声明)
    float area;       //梯形的面积(变量声明)
    float computerArea() {          //定义方法 computerArea
      area = (above+bottom)*height/2.0f;
      return area;
    }
    void setHeight(float h) {       //定义方法 setHeight
      height = h;
    }
}
```

“Lader”最好改为“Ladder”,“above”最好改为“topline”或“upperLine”,“bottom”最好改为“baseline”或“lowerLine”,“computerArea”最好改为“computeArea”。

pp.55-56 代码 3

```
public class A {
   int m = 10,sum = 0;          //成员变量，在整个类中有效
   void f() {
      if(m>9) {
         int z = 10;            //z 仅仅在该复合语句中有效
         z = 2*m+z;
      }
      for(int i=0;i<m;i++) {
          sum = sum+i;          //i 仅仅在该循环语句中有效
      }
      m = sum;                  //合法，因为 m 和 sum 有效
      z = i+sum;                //非法，因为 i 和 z 已无效
   }
}
```

7. 关于同名的成员变量和局部变量（包括形式参数）

局部变量(包括形式参数)优先。若指的是成员变量，则要在前面加上“this.”,“this”指代“当前对象名（本对象名)”。

p.56 代码 2

```
class Tom {
    int x = 10,y;
    void f() {
      int x = 5;
      y = x+x;//y 得到的值是 10，不是 20。如果方法 f 中没有"int x=5;", y 的值将是 20
    }
}
```

p.56 代码 3

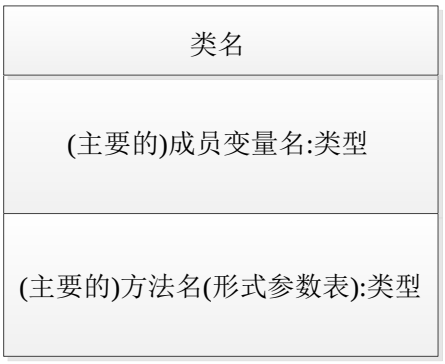
```
class Tom {
    int x = 10,y;
    void f() {
        int x = 5;
        y = x+this.x;       //y 得到的值是 15
    }
}
```

在编写一个方法时，还不知道以后哪个对象要调用该方法，而该方法里面却要涉及该对象，另外，就算提前知道了要调用该方法的对象，那极有可能不止一个对象。对于这种麻烦的局面，语言设计者想出了万能对象名表示法“this”（非 Java 首创）。

8. 尚未赋值的成员变量有默认值（p.60 倒数第 2 段），而尚未赋值的局部变量无默认值。
pp.56-57 代码 4

```
class InitError {
    int x = 10,y;          //y 的默认值是 0
    void f() {
        int m;             //m 没有默认值，但编译无错误
        x = y+m;           //无法通过编译，因为在使用 m 之前未指定 m 的值
    }
}
```

9. 类的 UML（统一建模语言）图



具体类的类名用正体，抽象类的类名用斜体。

4.3 构造方法与对象的创建

1. 构造方法

（1）类名(形式参数表){
        方法体
    }
    注意开头无“类型”，“方法名”即为“类名”。

（2）即初始化操作。可定义不止一个，也可不定义。

（3）若定义不止一个，则它们的形式参数表要互不相同。形式参数表互不相同指把形式参数名去除后互不相同。

p.58 代码 2

```
class Point {
    int x,y;
    Point() {
        x=1;
        y=1;
    }
    Point(int a,int b) {
        x=a;
        y=b;
    }
}
```

pp.58-59 代码 3

```
class Point {
    int x,y;
    Point() {                          //是构造方法
        x = 1;
        y = 1;
    }
    void Point(int a,int b) {    //不是构造方法（该方法的类型是 void）
        x = a;
        y = b;
    }
    int Point() {                      //不是构造方法（该方法的类型是 int）
        return 12;
    }
}
```

（4）若不定义，则按默认构造方法处理。其形式参数表和方法体都为空。

2. 创建对象

（1）声明对象名

类名 对象名;

对象名是引用类型变量，用于存储对象的引用，可看成起始地址。

注意此时还没有对象。

Java 的引用类型接近 Pascal 的简陋版指针类型，而远离 C 的豪华版指针类型。因而，在有些场合，使用 Java 编程比使用 C 编程轻松多了，也不太会犯错了，当然，程序的执行效率也甭想有多高了。

（2）引入对象

对象名 = new 构造方法名(实际参数表);

注意此时已有对象。

（3）对象的内存模型

声明对象名后：

对象名

引入对象后：

对象名　　对象

若类无成员变量，其对象也有占用的空间。

p. 59 代码 3

```
class XiyoujiRenwu {
    float height,weight;
    String head, ear;
    void speak(String s) {
       System.out.println(s);
    }
}
public class Example4_1 {
    public static void main(String args[]) {
        XiyoujiRenwu zhubajie;          //声明对象
        zhubajie = new XiyoujiRenwu();//为对象分配变量(使用 new 和默认的构造方法)
    }
}
```

注意尚未赋值的成员变量有默认值。

XiyoujiRenwu zhubajie;

zhubajie

zhubajie = new XiyoujiRenwu();

zhubajie　height weight　head　　ear

pp. 59-60 代码 4

```
class Point {
    int x,y;
    Point(int a,int b) {
        x = a;
        y = b;
    }
}
public class Example4_2 {
   public static void main(String args[]) {
      Point p1,p2;                    //声明对象 p1 和 p2
      p1 = new Point(10,10);          //为对象 p1 分配变量(使用 new 和类中的构造方法)
      p2 = new Point(23,35);          //为对象 p2 分配变量(使用 new 和类中的构造方法)
   }
}
```

Point p1, p2;

p1

p2

p1 = new Point(10, 10);

p1 x y

p2 = new Point(23, 35);

p2 x y

3. 使用对象

（1）点运算符

（2）体现属性（使用成员变量）

对象名.成员变量名

（3）体现行为（调用方法）

对象名.方法名(实际参数表)

（4）体现封装

pp.62-63 代码 1

```
class XiyoujiRenwu {
    float height,weight;
    String head, ear;
    void speak(String s) {
       head = "歪着头";
       System.out.println(s);
    }
}
public class Example4_3 {
    public static void main(String args[]) {
        XiyoujiRenwu zhubajie,sunwukong;      //声明对象
        zhubajie = new XiyoujiRenwu();        //为对象分配变量
        sunwukong = new XiyoujiRenwu();
        zhubajie.height = 1.80f;              //对象给自己的变量赋值
        zhubajie.head = "大头";
        zhubajie.ear = "一双大耳朵";
```

```
        sunwukong.height = 1.62f;              //对象给自己的变量赋值
        sunwukong.weight = 1000f;
        sunwukong.head = "秀发飘飘";
        System.out.println("zhubajie 的身高: "+zhubajie.height);
        System.out.println("zhubajie 的头:"+zhubajie.head);
        System.out.println("sunwukong 的重量:"+sunwukong.weight);
        System.out.println("sunwukong 的头:"+sunwukong.head);
        zhubajie.speak("俺老猪我想娶媳妇");                    //对象调用方法
        System.out.println("zhubajie 现在的头:"+zhubajie.head);
        sunwukong.speak("老孙我重 1000 斤,我想骗八戒背我");      //对象调用方法
        System.out.println("sunwukong 现在的头:"+sunwukong.head);
    }
}
```

XiyoujiRenwu zhubajie, sunwukong;

zhubajie

sunwukong

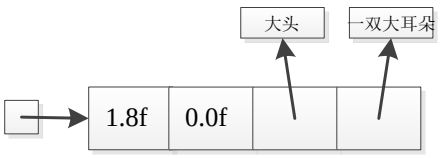
zhubajie = new XiyoujiRenwu();

zhubajie height weight head ear

sunwukong = new XiyoujiRenwu();

sunwukong height weight head ear

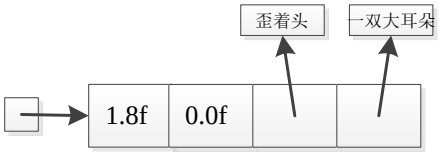
zhubajie.height = 1.8f;
zhubajie.head = "大头";
zhubajie.ear = "一双大耳朵";



zhubajie height weight head ear

sunwukong.height = 1.62f;
sunwukong.weight = 1000.0f;
sunwukong.head = "秀发飘飘";

zhubajie.speak("俺老猪我想娶媳妇");



sunwukong.speak("老孙我重 1000 斤,我想骗八戒背我");

4. 对象名和对象

（1）对象名和对象是两码事。

（2）要注意是否让对象名指向了对象。

（3）可让多个对象名同时指向某个对象。

（4）若没有任何对象名指向某个对象，则该对象会被垃圾回收器自动回收。Java 有构造方法，但无析构方法（C++有）。

p.66 11.1-2 “内存泄露”改为“内存泄漏（memory leak）”，“内存溢出”即“内存不足（out of memory）”。

p.65 代码 1

```
class Point {
    int x,y;
    void setXY(int m,int n){
        x = m;
        y = n;
    }
}
```

setXY 不是构造方法。

```
public class Example4_4 {
   public static void main(String args[]) {
      Point p1,p2;
      p1 = new Point();
      p2 = new Point();
      System.out.println("p1 的引用:"+p1);
      System.out.println("p2 的引用:"+p2);
      p1.setXY(1111,2222);
      p2.setXY(-100,-200);
      System.out.println("p1 的 x,y 坐标:"+p1.x+","+p1.y);
      System.out.println("p2 的 x,y 坐标:"+p2.x+","+p2.y);
      p1 = p2;
      System.out.println("将 p2 的引用赋给 p1 后: ");
      System.out.println("p1 的引用:"+p1);
      System.out.println("p2 的引用:"+p2);
      System.out.println("p1 的 x,y 坐标:"+p1.x+","+p1.y);
      System.out.println("p2 的 x,y 坐标:"+p2.x+","+p2.y);
   }
}
```

Point p1, p2;

p1

p2

p1 = new Point();

p1 x y

p2 = new Point();

p2 x y

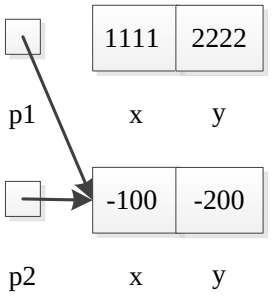
p1.setXY(1111, 2222);

p1 x y

p2.setXY(-100, -200);

p2 x y

p1 = p2;



4.4 类与程序的基本结构

1. 一个 Java 应用程序（工程）包含一个或多个类。

2. 只能在其中一个类中定义一个主方法（main），该类称为主类。

p.67 代码 1

```
public class Rect {
   double width;                    //矩形的宽
   double height;                   //矩形的高
   double getArea()  {
      double area = width*height;
      return area;
   }
}
```

方法 getArea 的局部变量 area 可不引入。

p.67 代码 2

```
public class Lader {
   double above;                    //梯形的上底
   double bottom;                   //梯形的下底
   double height;                   //梯形的高
   double getArea() {
      return (above+bottom)*height/2;
   }
}
```

p.67 代码 3

```
public class Example4_5 {
   public static void main(String args[]) {
      Rect ractangle = new Rect();
      ractangle.width = 109.87;
      ractangle.height = 25.18;
      double area=ractangle.getArea();
      System.out.println("矩形的面积:"+area);
      Lader lader = new Lader();
      lader.above = 10.798;
      lader.bottom = 156.65;
      lader.height = 18.12;
      area = lader.getArea();
      System.out.println("梯形的面积:"+area);
    }
}
```

“ractangle”最好改为“rectangle”。

3. 提倡一个源文件只包含一个类。

4.5 参数传递

1. 传递机制

都是值传递。

2. 基本数据类型参数

（1）实际参数的取值范围不能大于对应形式参数的取值范围。

（2）参数的值只进不出。

pp.68-69 代码 1

```
class Computer{
   int add(int x,int y){
      return x+y;
   }
}
public class Example4_6 {
   public static void main(String args[]){
      Computer com = new Computer();
      int m = 100;
      int n = 200;
      int result = com.add(m,n);          //将 m、n 的值“传值”给参数 x、y
      System.out.println(result);
      result = com.add(120+m,n*10+8);
                                  //将表达式 120+m 和 n*10+8 的值“传值”给参数 x、y
      System.out.println(result);
   }
}
```

类 Computer 无成员变量。

3. 引用类型参数

参数的值也只进不出，但所指对象的值感觉上好像可进可出。

p.70 代码 1

```
public class Battery {
    int electricityAmount;
    Battery(int amount){
        electricityAmount = amount;
    }
}
```

p.70 代码 2

```
public class Radio {
    void openRadio(Battery battery){
        battery.electricityAmount = battery.electricityAmount - 10;
                                                          //消耗了电量
    }
}
```

类 Radio 无成员变量。

p.70 代码 3

```
public class Example4_7 {
   public static void main(String args[]) {
      Battery nanfu = new Battery(100);       //创建电池对象
      System.out.println("南孚电池的储电量是:"+nanfu.electricityAmount);
      Radio radio = new Radio();              //创建收音机对象
      System.out.println("收音机开始使用南孚电池");
      radio.openRadio(nanfu);                 //打开收音机
      System.out.println("目前南孚电池的储电量是:"+nanfu.electricityAmount);
   }
}
```

Battery nanfu = new Battery(100);

（a）

nanfu

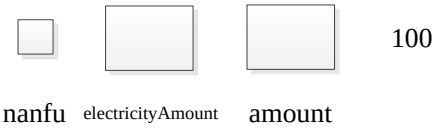
（b）

nanfu electricityAmount

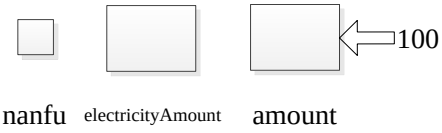
（c）

nanfu electricityAmount amount

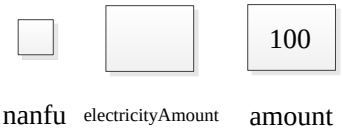
（d）



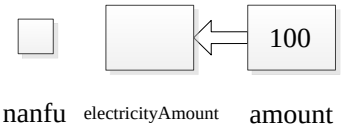
(e)



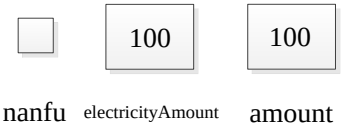
(f)



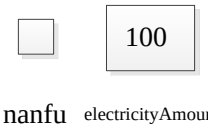
(g)



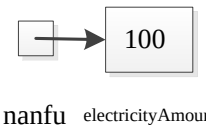
(h)



(i)

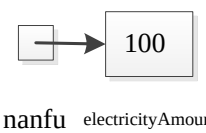


(j)

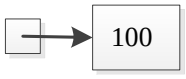



```
Radio radio = new Radio();
```

(k)



(l)

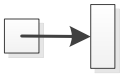


nanfu electricityAmount

radio

(m)

nanfu electricityAmount

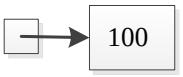


radio

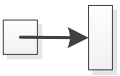
radio.openRadio(nanfu);

(n)

battery

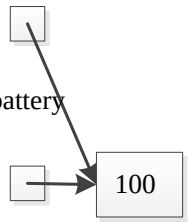


nanfu electricityAmount



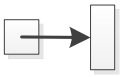
radio

(o)



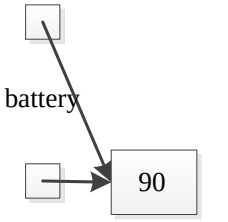
battery

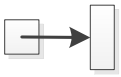
nanfu electricityAmount



radio

(p)

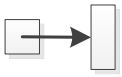




radio

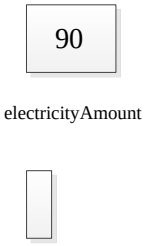
（q）

nanfu electricityAmount



radio

（r）



（s）都没了

Battery 类的对象有权使用该类的成员变量 electricityAmount，但 Radio 类的对象无权使用 Battery 类的成员变量 electricityAmount。但如果把 Battery 类的某个对象的引用通过参数传递给了 Radio 类的某个方法，那么 Radio 类的某个对象就可以通过调用这个方法，借助 Battery 类的那个对象使用 Battery 类的成员变量 electricityAmount 了。

4. 个数可变参数