第4章 类与对象

4.1 类和对象

4.2 参数传值

4.3 函数重载

4.4 Staitc关键字

4.5 This 关键字

4.6 包

导读

难点

- 参数传值
- 包和import语句

面向对象基础

- 面向对象编程的基本概念
  - OOP（Object-Oriented Programming）
    - 起源于Simula以及后来的Smalltalk语言
    - 面向过程：算法＋数据结构
      - 算法⇔函数
      - 数据结构⇔组织和表示数据
    - 面向对象：事物＋事物间的关系
      - 事物⇔类、对象
      - 关系⇔类之间的关系

面向对象基础

- 对象与类：
  - 对象：代表现实世界中可以明确标识的任何事物
    - 成员变量：表明对象的状态、属性
    - 方法：表明对象所具有的行为
    - 例如：一台PC机对象
      - 成员变量：CPU、内存、硬盘、.....
      - 方法：运算、输入、输出、.....
  - 类：构造**对象的模板**或蓝图
    - 对象是类的实例
    - 由类构造对象的过程称为创建类的实例

面向对象基础

- 类与对象
  - 用面向对象（OO）技术写程序的基本步骤：
    - 根据需要首先**定义类**
      - 指出类的名称、变量和方法，确定类成员被访问的权限等
    - 利用类**创建相应的对象**
      - 系统为该对象分配对应的内存空间
    - 程序运行时，由系统根据需要与**对象交换信息**

面向对象基础

- 面向对象的三要素：
  - 类的封装：将类的基本成分封装在类体之中，使之与外界分隔开
    - 例如：计算机由多个零件构成，每种零件由不同制造商生产，封装在机箱内，只提供用户使用计算机的接口；封装过程中注意安全层次
    - 实现封装的关键：绝不能让类中的方法直接访问其他类的实例变量，但可以访问它自己类的实例变量
    - 给予对象“黑盒”特征——**提高重用性和可靠性的关键**

面向对象基础

- 面向对象的三要素：
  - 类的继承：新的类继承原有类的基本特性
    - 提高程序的**可复用性**
    - 例如：汽车与（公共汽车、出租车、货车）之间就是父类与子类的关系
    - **在Java中，所有类都源自同一个类：Object**

面向对象基础

- 面向对象的三要素：
  - 类的多态性：是指一个名称具有多种功能，或者相同接口有多重实现方法
  - Java中多态的实现有三种方法进行：
    - **方法重载**：多个方法具有相同名称，但方法的参数个数和参数类型不同
    - **方法覆盖**：类派生过程中，子类与父类的方法不仅名称相同，参数也完全相同，只是功能有所不同
    - **接口**：本质是特殊的类，只定义方法的原型，而方法的实现在子类中具体给出

§4.2 类

类是组成Java程序的基本要素。类封装了一类对象的状态和方法。

类是用来定义对象的模板。

类的实现包括两部分：类声明和类体。

§4.2 .1 类声明

```
[public] class 类名 extends 父类  implements 接口1，接口2
{
        类体的内容
}
```

给类命名时，遵守下列编程风格（这不是语法要求的，但应当遵守）：

1．如果类名使用拉丁字母，那么名字的首字母使用大写字母，如Person，Car。

2．类名最好容易识别、见名知意。当类名由几个“单词”复合而成时，每个单词的首字母使用大写。

§4.2.2 类体

◆ 类声明之后的一对大括号“{”，“}”以及它们之间的内容称作类体，大括号之间的内容称作类体的内容。

◆类体的内容包括两部分：

① 变量的声明，用来刻画属性；

② 方法的定义，用来刻画功能。

§4.2.3　变量声明

编程风格

（1）一行只声明一个变量。

（2）变量的名字符合标识符规定。

（3）变量名字见名知意，避免容易混淆的变量名字。

§4.2.3 变量声明

1. 变量的类型：

类体中变量声明部分所声明的变量被称作类的**成员变量**。在方法体中声明的变量和方法的参数被称作**局部变量**。

2. 变量的有效范围：

成员变量在整个类内都有效；

局部变量只在声明它的方法内有效：方法参数在整个方法内有效，方法内的局部变量从声明它的位置之后开始有效。

3. 成员变量的隐藏：

如果局部变量的名字与成员变量的名字相同，则**成员变量被隐藏**，即这个成员变量在这个方法内暂时失效。

§4.2.3 变量声明

3. 实例变量与类变量：

在声明成员变量时，用关键字static给予修饰的称作**类变量**（类变量也称为static变量，静态变量），否则称作**实例变量**。

§4.2.4 方法声明

方法的定义包括两部分：方法声明和方法体。

1．方法声明

方法声明包括方法名和方法的返回类型，如：

```
返回类型    方法名( 形式参数 ) {
        …方法体
    }
```

2．方法体

方法声明之后的一对大括号“{”，“}”以及之间的内容称作方法的方法体。方法体的内容包括局部变量的声明和Java语句。

§4.2.4 方法声明

```
public class Person {
    int age;
    public int getAge(){
        return age;
    }

    public void setAge(int tempage){
    age=tempage;
    }
}
```

§4.2.5 构造方法

构造方法是一种特殊方法，它的名字必须与它所在的类的名字完全相同，而且没有返回类型。

```
public class Employee{
    public Employee(String n,double s)      {
            name=n;
            salary=s;
    }
    public Employee(){
            name="Anmous";
            salary=250;
    }
    ......
    private String name;
    private double salary;
}
```

示例：StudentConstructor类

- 类中未指明构造方法，系统给出默认构造方法，即参数为空，**所有成员均初始化称该类型的默认值（默认值有 哪些，运行**Initialization 类）

§4.2.6　方法重载

方法重载：

一个类中可以有多个方法具有相同的名字，但这些方法的参数必须不同，即或者是参数的个数不同，或者是参数的类型不同。

注意：方法的返回类型和参数的名字不参与比较

方法重载

- 方法重载有两种类型：**构造方法重载**和**普通方法重载**。
- **普通方法重载**
  - 例如：System.out.println方法

    public void println();

    public void println(boolean x);

    public void println(char x);

    public void println(int x);

    public void println(double x);

    public void println(String s);

    示例：

    CelsiusConverter类中的getFahrenheit方法

    StudentConstructor中的guessScore方法

§4.3 对象

§4.3.1 创建对象

1．对象的声明

People zhangPing;

2.为声明的对象分配内存

zhangPing=new People();

3．对象的内存模型

（1）声明对象时的内存模型

（2）对象分配内存后的内存模型

4．创建多个不同的对象

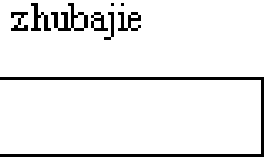


图 4.1 未分配实体的对象

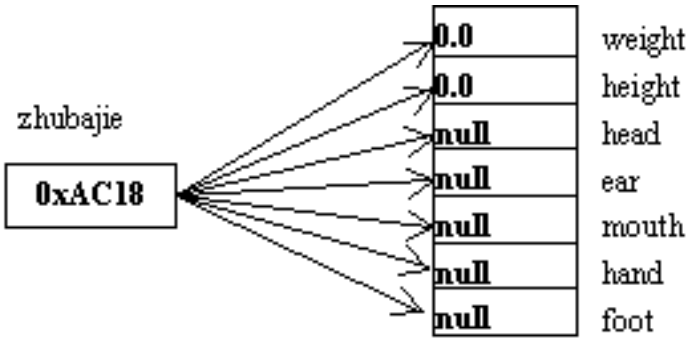


图 4.2 分配实体后的对象

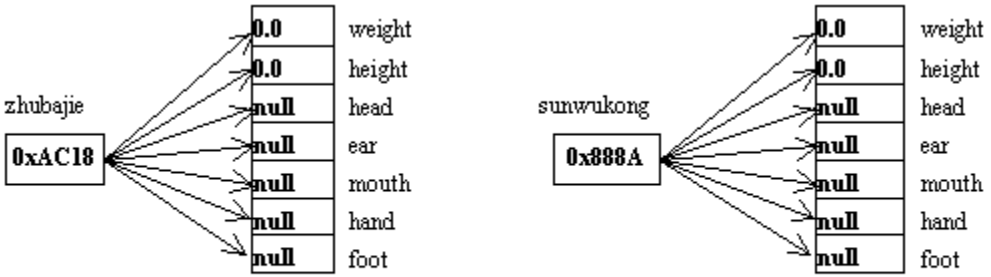


图 4.3 创建多个对象

§4.3.2　使用对象

1．通过使用运算符“.”　对象操作自己的变量（对象的属性）

2．使用运算符“.”，对象调用类中的方法（对象的功能）

§4.3.3 对象的引用和实体

当用类创建一个对象时，类中的成员变量在分配内存空间，这些内存空间称作该对象的实体或对象的变量，而对象中存放着引用。

一个类创建的两个对象，如果具有相同的引用，那么就具有完全相同的实体。

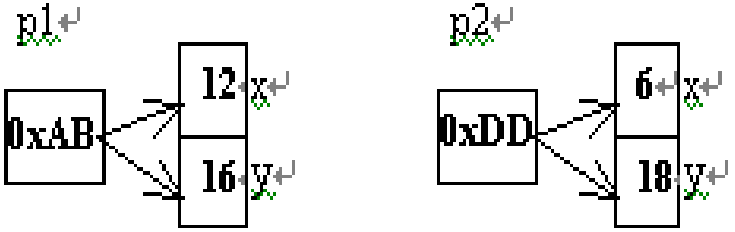


图 4.5 对象内存模型

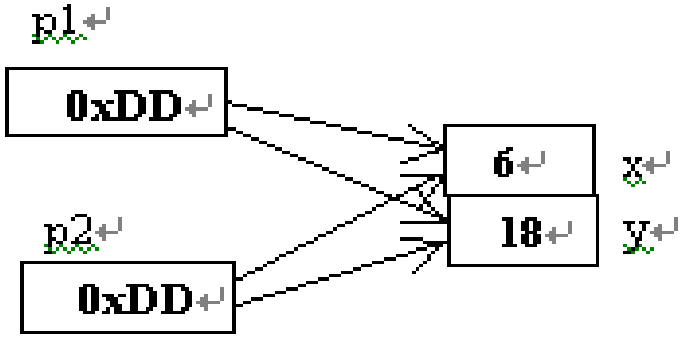


图 4.6 p1=p2 后的对象内存模型

§4.4　参数传值

§4.4.1 基本数据类型参数的传值

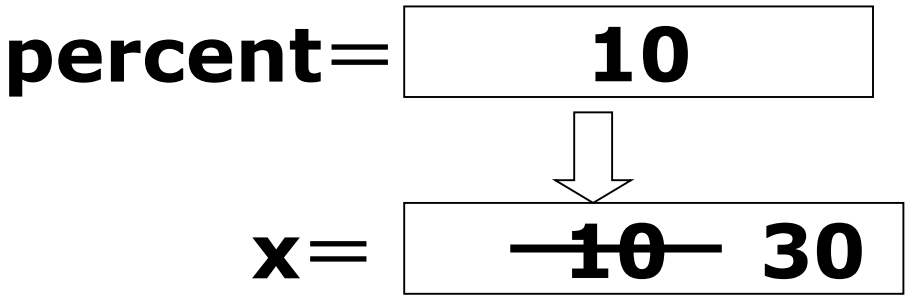
方法参数举例

```
//传基本类型
public static void tripleValue(double x)
{
    x=3*x;
}
//调用此方法，对传入的x的修改不会对外界起任何作用
double percent=10;
tripleValue(percent);
```

//percent的值仍然为10

类的方法参数

- 基本类型传递：值拷贝



§4.4　参数传值

§4.4.1 基本数据类型参数的传值

对于基本数据类型的参数，向该参数传递的值的级别不可以高于该参数的级别 。

void print(double a);

print(12.5d)

print(12.5f)

print(12)

§4.4.2　引用类型参数的传值

当参数是引用类型时，“传值”传递的是变量中存放的“引用”，而不是变量所引用的实体。

例4-5

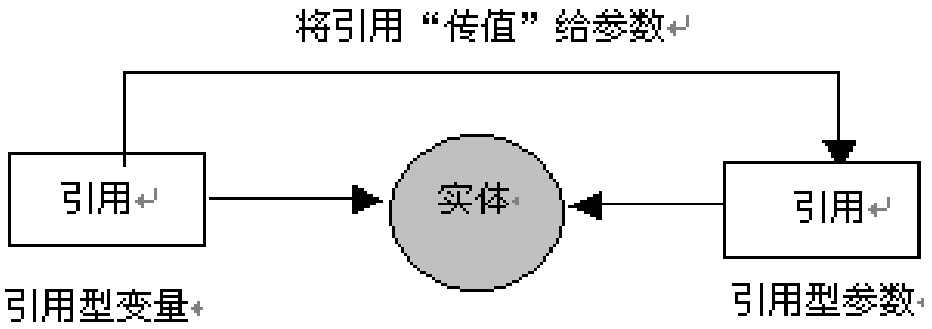


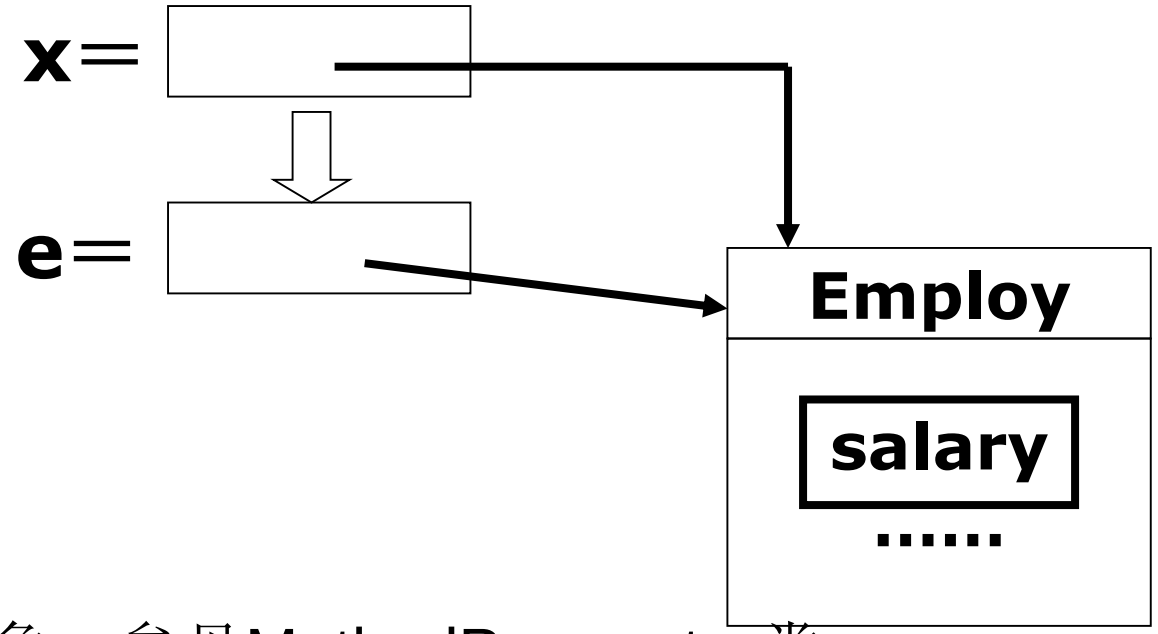
图 4.8　引用类型参数的传值

§4.4.2　引用类型参数的传值

```
class Employ{
    double salary;
    ......
}
//传对象的某个方法
public static void tripleSalary(Employ e)
{
    e.salary=200;
}
//调用该方法，会修改对象的值——引用传递
Employ x=new Employ();
tripleSalary(x);  //x.salary变成了200
```

§4.4.2　引用类型参数的传值

- 类对象传递：引用拷贝



数组也是对象，参见MethodParameter类

类的方法参数

- 类对象传递：引用拷贝
  - 注意欺骗性，看几个典型例子

```
static void change(Person p1){
    p1=new Person("张三");
}
static void change2(Person p1){
    p1.setName("张三");
}
```

```
public class Person {

private String name="未知“;

public Person(String tempname){
        name=tempname;
}

public String getName(){
        return name;
}
public void setName(String name
        this.name=name;
}
}
```

```
Person p=new Person("李四");
change(p);
System.out.println(p.getName()); //变化没有?
```

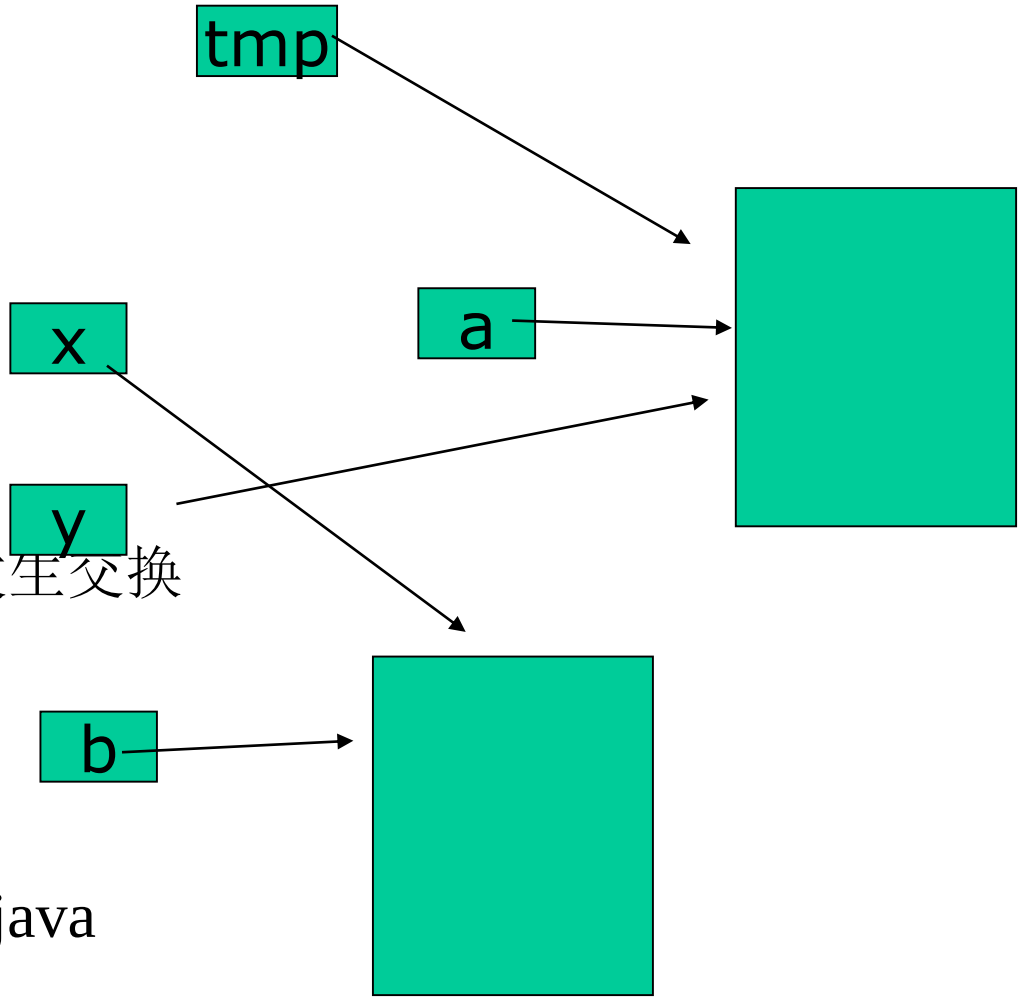
No

```
p=new Person("李四");
change2(p);
System.out.println(p.getName()); //变化没有?
```

yes

- 例3：交换两个Employ类的对象

```
public static void swap(Employ x,Employ y){
    Employ tmp=x;
    x=y;
    y=tmp;
}
//调用上述交换函数
Employ a=new Employ();
Employ b=new Employ();
swap(a,b);  //a、b并没有发生交换
```



例子：MethodParameter.java

§4.4.3　可变参数

可变参数是指在声明方法时不给出参数列表中从某项直至最后一项参数的名字和个数，但这些参数的类型必须相同。可变参数使用“...”表示若干个参数，这些参数的类型必须相同，最后一个参数必须是参数列表中的最后一个参数。

例如：`public void f(int ... x)`

例 MethodParameter2.java

§4.6 static关键字

- 用static 修饰的变量 **类变量**
- 没有用static 修饰的变量 **实例变量**

- 方法声明中用static 修饰的方法—**类方法**
- 方法声明中不用static 修饰的方法—**实例方法**

```
public class Test{
    private int x;
    static private int y;            // static private int y=10;
    static {
        y=20;
    }
    Test(int t){
        x=t;
    }
    public static void main(String args[]){
        System.out.println(Test.y);
        Test t1=new Test(32);
        Test t2=new Test(52);
        System.out.println(t1.y);
        System.out.println(t2.y);
        t1.y=100;
        System.out.println(t1.x);
        System.out.println(t2.x);
        t1.x=300;
        System.out.println(t1.x);
        System.out.println(t2.x);
        System.out.println(t1.y);
        System.out.println(t2.y);
    }
}
```

§4.6.1 实例变量和类变量的区别

如果类中有类变量，那么所有对象的这个类变量都分配给相同的一处内存，改变其中一个对象的这个类变量会影响其它对象的这个类变量。也就是说对象共享类变量。

Static变量应用例子：
建立一个Book类，包括title，salenumber和gensalenumber三个成员变量，分别表示书名，一本书的出售量，以及所有书的出售量。再设计一个sale方法，使得每卖出一本书，相应的出售量有所增加。
建立一个BookTestStaticVarial类，在main函数中模拟有3种书，假设一共卖出书籍5次，统计每种书的销售量和总销售量。

参见Book.java 和BookTest.java

§实例方法和类方法的区别

方法声明时，方法类型前面不加关键字static修饰的是实例方法、加static修饰的是类方法（静态方法）。

§4.2.8 几个值得注意的问题

1．对成员变量的操作只能放在方法中，方法可以对成员变量和该方法体中声明的局部变量进行操作。

2．需要注意的是：实例方法既能对类变量操作也能对实例变量操作，而类方法只能对类变量进行操作。

3．一个类中的方法可以互相调用，实例方法可以调用该类中的其它方法；类中的类方法只能调用该类的类方法，不能调用实例方法。

§4.7 this 关键字

this是Java的一个关键字，表示某个对象。this可以出现在实例方法和构造方法中，但不可以出现在类方法中。

§4.7.1　在构造方法中使用this

this关键字出现在类的构造方法中时，代表使用该构造方法所创建的对象。

例　Rectangle.java

§4.7.2　在实例方法中使用this

当this关键字出现实例方法中时，代表正在调用该方法的当前对象。

当实例成员变量在实例方法中出现时，默认的格式是：this.成员变量。

当static成员变量在实例方法中出现时，默认的格式是：类名.成员变量。

例　thisusage(2).java

§4.8　包

包是Java语言中有效地管理类的一个机制。

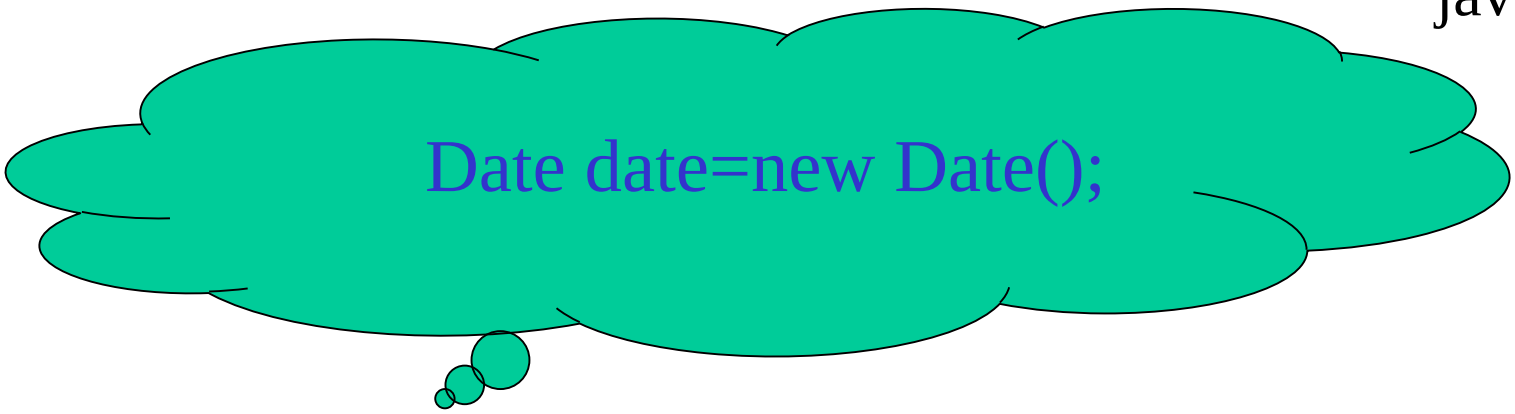
4.8.1 使用包的必要性

Java是跨平台的网络编程语言，用Java语言编写的类或应用程序常在网络中使用。对于世界各地应用Java语言的程序员来说，他们极有可能对不同的类使用了相同的名字。

例如Date类：

java.sql.Date?

java.util.Date?



4.8.1 使用包的必要性

JVM在运行时知道调用哪个类库中的Date类吗?

4.8.1 使用包的必要性

为避免类和接口命名冲突，Java提供了包机制，允许将相关的类和接口放在一个特定的包中，访问类或接口时需要包含所属包的信息，这样就可以通过包名来限定类名，从而避免命名冲突。

此外，**使用包还可以进行访问级别控制**：同一个包中的类，相互之间有不受限制的访问权限，而在不同包中，只有public类可被其他类访问。

§4.8.1 包语句

包的本质是文件夹，在创建包后，通过关键字package声明包语句。

package语句的一般格式为：

package 包名;

例如，声明**com.java.sun**包：

package com.java.sun

§4.8.1 包语句

1.包名可以是一个合法的标识符，也可以是由多个有层次结构的合法的标识符构成。由多个标识符组成时，各标识符之间使用“.”隔开。

2.包名一般使用小写字母。

3.每个源文件最多只能有一条包声明语句，并且包声明语句必须放在源文件的所有语句的前面，即必须放在源文件的开始位置处。

§4.8.2 有包名的类的存储目录

程序如果使用了包语句，例如：

```
package tom.jiafei;
```

那么存储文件的目录结构中必须包含有如下的结构

```
...\tom\jiafei
```

如：

```
c:\temp\tom\jiafei
```

并且要将源文件编译得到的类的字节码文件保存在目录**c:\temp\tom\jiafei**中（源文件可以任意存放）。

§4.8.3 运行有包名的主类

假设**tom\jiefei**的上一层目录是**temp**，必须先进入到文件夹

```
cd  C:\temp
```

在该目录下运行：

```
java   tom.jiafei.主类名
```

注：主类名已更变为“包名.主类名”

§4.9 import 语句

一个类可能需要另一个类声明的对象作为自己的成员或方法中的局部变量，如果这两个类在同一个包中，当然没有问题。

如果一个类想要使用的那个类和它不在一个包中，要使用import语句完成使命。

§4.9.1 引入类库中的类

导入类库中的类：

```
import java.util.Date;    //单个类
import java.util.*;       //多个类
```

*只导入包中直接包含的类，不包括子目录中的类

常用Java内置包

java.lang

Java语言包是Java的核心类库，其中包含了运行Java程序必不可少的系统类。

java.awt

包含用于创建用户界面和绘制图形图像的所有类，这些类是一些“重量级”的组件。

javax.swing

是Java的扩展类库，包含了用于创建用户界面的“轻量级”组件。因为swing包中的绝大部分组件完全由自己来显示每个组件。

java.applet

包含了创建java applet所必需的类以及applet与其applet 上下文通信的类。

java.io

提供了可以让我们以输入、输出流的形式来读写文件的各种输入、输出类。

java.util

包含了各种实用工具类和接口，如Random、Date、Calendar、Collection、EventListener等。

java.net

包含了构建网络应用程序的所必需的类，如Socket、ServerSocket、URL等类。

java.sql

提供使用访问并处理存储在关系数据库中的数据的API，如DriverManager 、Connection 、Statement等。

java.text

提供以与自然语言无关的方式来处理文本、日期、数字和消息的类和接口。

§4.9.1 引入类库中的类

导入自定义包中的类：

```
import tom.jiafei.Flower;

import tom.jiafei.*;
```

§4.9.3 使用无包名的类

如果一个类想使用无名包中的类，只要将这个无包名的类的字节码和当前类保存在同一目录中即可。

§4.9.4 避免类名混淆

1．区分无包名和有包名的类

如果想同时使用tom.jiafei包中的A类和无名包中的A类，就不能省略包名，例如：

```
A a1=new A();
tom.jiafei.A a2=new tom.jiafei.A();
```

2．区分有包名的类

如果一个源文件引入了两个包中同名的类，那么在使用该类时，不允许省略包名，比如：

```
tom.jiafei.A  bird=new tom.jiafei.A();
sun.com.A  goat=new sun.com.A();
```

§4.10 访问权限

访问限制修饰符有private、protected和public，都是Java的关键字，用来修饰成员变量或方法。

§4.10.1 私有变量和私有方法

用关键字**private**修饰的成员变量和方法称为私有变量和私有方法。

对于私有成员变量或方法，只有在本类中创建该类的对象时，这个对象才能访问自己的私有成员变量和类中的私有方法。

不能用private修饰局部变量

§4.10.2 共有变量和共有方法

用public修饰的成员变量和方法被称为共有变量和共有方法。

§4.10.3　受保护的成员变量和方法

用protected修饰的成员变量和方法被称为受保护的成员变量和受保护的方法 。

§4.10.4 友好变量和友好方法

不使用private，public，protected修饰的成员变量和方法。

在同一包内，友好访问

- 访问权限不仅涉及到成员变量和方法

→类

§4.10.5 public类与友好类

类声明时，如果在关键字class前面加上public关键字，就称这样的类是一个public类 。

--可以在任何另外一个类中，为public类创建对象。

如果一个类不加public修饰，这样的类被称作友好类。

--在另外一个类中使用友好类创建对象时，要保证它们是在同一包中。

§4.14 小结

1.类是组成Java源文件的基本元素,一个源文件是有若干个类组成的。

2．类体可以有两种重要的成员：成员变量和方法。

3．成员变量分为实例变量和类变量。类变量被该类的所有对象共享；不同对象的实例变量互不相同。

4.除构造方法外,其它方法分为实例方法和类方法。类方法不仅可以由该类的对象调用,也可以用类名调用；而实例方法必须由对象来调用。

5．实例方法即可以操作实例变量也可以操作类变量，当对象调用实例方法时，方法中的成员变量就是指分配给该对象的成员变量，其中的实例变量和其它对象的不相同，即占有不同的内存空间；而类变量和其它对象的相同，即占有相的内存空间。类方法只能操作类变量，当对象调用类方法时，方法中的成员变量一定都是类变量，也就是说该对象和所有的对象共享类变量。

6．在编写Java源文件时，可以使用import语句引入有包名的类；也可以使用静态导入引入有包名类的类变量。

7．对象访问自己的变量以及调用方法受访问权限的限制。