《Android 系统下 Java 编程详解》

作者：华清远见

第 2 章 面向对象程序设计初步

本章简介

本章从面向对象的概念出发，介绍了程序设计种类、面向对象程序设计的特征；介绍了面向对象程序设计中的对象、类、属性等重要概念；并描述了类、属性、方法的定义和声明；讲述了类的构造器的概念及使用，介绍对象的创建和使用。讲述如何通过类的定义来实现信息隐藏和封装；介绍了 Java 源文件的结构，package 和 import 的用法；JDK 中的常用的包。

2.1 面向对象概念

2.1.1 从结构化程序设计到面向对象程序设计

面向对象程序设计与结构化程序设计都是设计和构造程序的方法。近年来面向对象程序设计的方法越来越受到人们的重视，成为解决软件危机的新途径，而结构化程序设计方法的使用在逐渐减少。几乎每一种最新推出的程序开发工具或语言都采用了面向对象程序设计的思路，面向对象程序设计形成了一套与结构化程序设计具有很大差异的方法。

结构化程序设计方法又称面向过程设计方法，起源于 20 世纪 70 年代。在这之前的程序设计基本采用过程式程序设计，虽然汇编语言已经取代了机器语言，但是对于一般的程序设计人员而言，它还是太抽象、太隐晦了。如果计算机要得到更大的发展，必须发明一些比汇编语言更易于阅读、编写的程序语言。在这种需求的刺激下，结构化程序设计方式产生了机构化程序设计，主要特点是采用自顶向下、逐步求精的程序设计方法：使用 3 种基本控制结构构造程序，任何程序都可由顺序、选择、重复 3 种基本控制结构构造。结构化设计的根本目标就是把复杂的系统分解成简单模块的层次结构。例如，你要装修房子，以前的过程式程序设计要求你必须从客厅开始装修，然后是卧室、厨房、卫生间，顺序不能颠倒，客厅没装好之前，休想踏进你的卧室半步。而结构化程序设计方式将你的客厅、卧室、卫生间、厨房都独立成一个模块，互相之间可以互不干扰地进行。

虽然结构化程序设计解决了软件设计开发中的一部分问题，但是它仍然存在不足。用结构化方法开发的软件，其稳定性、可修改性和可重用性都比较差，这是因为结构化方法的本质是功能分解，从代表目标系统整体功能的单个处理着手，自顶向下不断把复杂的处理分解为子处理，这样一层一层地分解下去，直到仅剩下若干个容易实现的子处理功能为止，然后用相应的工具来描述各个最低层的处理。因此，结构化方法是围绕实现处理功能的“过程”来构造系统的。然而，用户需求的变化大部分是针对功能的，因此，这种变化对于基于过程的设计来说是灾难性的。用这种方法设计出来的系统结构常常是不稳定的，用户需求的变化往往造成系统结构的较大变化，从而需要花费很大代价才能实现这种变化。

结构化程序设计的局限性催生了面向对象的思想。面向对象（OO）并不是一个新概念，它在 20 世纪 70 年代就出现了，但是，因为受到软/硬件的限制，直到 90 年代，它才为大众所接受并成为程序设计的主流思想。面向对象的方法与结构化的程序设计根本区别在于把系统看成一起工作来完成某项任务的对象集合，而对象是系统对消息做出响应的事物，所以面向对象方法中最值得关注的不是它应该做什么，而是它如何做出反应，也就是消息，这是和结构化设计的根本不同。

在面向对象的设计思想指导下，产生了第一个面向对象的语言——20 世纪 60 年代的 Simula-67。它的目的是解决模拟问题。一个典型的应用是模拟银行系统的运作（银行是最早采用计算机的行业之一），将银行的出纳部门、客户、业务等实体模拟成一个个的对象，把这些在程序的运行中除了状态外其他方面都一样的对象归纳在一起，就成了更高抽象层面的“类”。在面向对象思想指导下产生了成功的面向对象编程语言——Smalltalk。并且，在此基础上又诞生了应用更加广泛的面向对象编程语言——C++及 Java。

2.1.2 面向对象特征

1．对象唯一性

每个对象都有自身唯一的标识，通过这种标识，可找到相应的对象。在对象的整个生命期中，它的标识都不改变，不同的对象不能有相同的标识。

2．抽象性

抽象性是指将具有一致的数据结构（属性）和行为（操作）的对象抽象成类。一个类就是这样一种抽象，它反映了与应用有关的重要性质，而忽略其他一些无关内容。任何类的划分都是主观的，但必须与具体的应用有关。

3．封装性

封装性就是把对象的属性和服务结合成一个独立的单位，并尽可能隐蔽对象的内部细节。

4．继承性

继承性是子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类的基础上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

继承性是面向对象程序设计语言不同于其他语言的最重要的特点，是其他语言所没有的。

5．多态性

多态性是指相同的操作或函数、过程可作用于多种类型的对象上并获得不同的结果。不同的对象收到同一消息可以产生不同的结果，这种现象称为多态性。

2.2 面向对象程序设计

2.2.1 知识准备：面向对象编程术语

- 类（Class）：类是对某种类型的对象定义变量和方法的原形。它表示对现实生活中一类具有相同特征的事物的抽象，是面向对象编程的基础。
- 对象（Object）：也称为实例（Instance），是类的具体存在。
- 属性（Attribute）：也称为成员变量、实例变量（Instance Variable）、字段（Field），表示一个对象所具有的状态。本书中，通常使用变量或者属性来称呼它。
- 方法（Method）：用于表示对象可以执行的动作，它包括名称、参数、返回类型，以及方法体等内容。
- 构造器（Constructor）：也称为“构造方法”，用于创建和初始化一个实例的特殊方法，包括名称、参数及“方法体”等，需要注意的是，它的名称必须和类名一致。
- 包（Package）：可以使用它来组织编写类，使得类的组织更加有序。

2.2.2 知识准备：对象

我们每天的生活、工作，无时无刻不在和“对象”打交道——衣服、食物、房子、汽车等。当处理这些对象时，不会将这些对象的属性（对象所具有的特点）和操作分开。如进出“房间”时，不会将“房门”这个属性和“开门”这个操作分开，它们是联系在一起的。

面向对象的编程思想力图使得程序和现实世界中的具体实体完全一致。

人们通过观察对象的属性和行为来了解对象。对象的属性描述了对象的状态，对象的行为描述了对象的功能。对象一般有如下特性：

（1）有一个名字以区别于其他对象。

（2）有一些状态用来描述它的某些特征。

（3）有一组操作，每个操作决定了对象的一种功能或行为。

这样，可以让程序员乃至非专业人员更好地理解程序。它涉及一个从“具体”到“抽象”，再从“抽象”到“具体”的过程。所谓“从‘具体’到‘抽象’”，也就是将现实世界中的一个个具体的“物体”（或称为“实体（Entity）”）相应的特征和行为抽象出来，并且将各种具有相同特征的“物体”分为一个个的“类”，如“汽车”类、“人”类、“房子”类等；而所谓“从‘抽象’到‘具体’”，就是将上面抽象出来的对应物体的“类”，使用具体的计算机语言来描述，比如，使用 Java 语言来描述“汽车”类、“人”类、“房子”类等，和使用 C++语言来描述这些类是有区别的。

正如前面所述，“类”相对于现实世界中的“实体种类”（Entity Category），如汽车、人类、房子等，它是现实生活中某类实体的抽象。而对象，或者实体（Instance），指的是这些种类中的一个个具体存在，如 Benz-600、Santana-2000 等具体的汽车，或者张三、李四等具体的个人。类和对象是面向对象编程思想中的核心和基础。类是作为对象的蓝图而存在的，所有的对象都依据相应的类来产生，在面向对象的术语中，这个产生对象的过程称为“实例化”。

用一个具体的例子来进一步说明“类”和“对象”之间的联系与区别。以汽车为例，只要是汽车，都应该有以下一些特性：轮子、引擎、方向盘、刹车等组件，可以通过这些组件来操作汽车，改变汽车的状态，如加速、转向、减速等，这些都是汽车的共性。具体到某辆汽车，它可能有 80cm 的轮子、40cm 的方向盘、A6 引擎，它是一个确定的实例。使用“汽车类”这个蓝图，就可以生产出一辆辆的汽车“实例”。这和盖房子一样，根据一张图纸，可以盖出任意数量的房子，而这些房子都有一样的特征。

2.2.3 知识准备：类

如果说一切都可以成为对象，那么是什么决定了某一类对象的外观和行为呢？

类是对某个对象的定义，用来描述一组具有相同特征对象的：

（1）应包括的数据。

（2）行为特征。

类包含有关对象动作方式的信息，包括它的名称、方法、属性和事件。实际上类本身并不是对象，因为它不存在于内存中。当引用类的代码运行时，类的一个新的实例，即对象，就在内存中创建了。一个类能从这个类在内存中创建多个相同类型的对象。因此，类应该包括两个方面的内容。

（1）属性：用来描述对象的数据元素称为对象的属性（也称为数据/状态）。

（2）方法：对对象的属性进行的操作称为对象的方法（也称为行为/操作）。

2.2.4 知识准备：类的声明

Java 语言中类的声明，也称类的定义，其语法规则如下：

```
[< modifiers>] class < class_name> {
[<attribute_declarations>]
[<constructor_declarations>]
[<method_declarations>]
}
```

其中各组成部分的含义如下：

<modifiers>为修饰符，可用的有 public、abstract 和 final 等关键字（关键字是 Java 语言中赋以特定含义，并用做专门用途的单词，类名、方法名和属性名及变量名不能是关键字），用于说明所定义的类有关方面的特性。对于各种关键字和它们的含义及各自的适用范围，请看后续章节的介绍。

类成员有 3 种不同的访问权限：

（1）公有（ public 关键字修饰 ）类成员可以在类外访问。

（2）私有（ private 关键字修饰 ）类成员只能被该类的成员函数访问。

（3）保护（ protected 关键字修饰 ）类成员只能被该类的成员函数或派生类的成员函数访问。

class 关键字表明这是一个类的定义，将告诉你一个新类型的对象看起来是什么样的。

<class_name>是类的名字，类名一般使用一个用于表示这个类的名词来表示。

<attribute_declarations>是属性（Attribute）声明部分。

<constructor_declarations>是构造器（Constructor）声明部分。

<method_declarations>是方法（Method）声明部分。

可以将上述的“学生”实体的基本特征当成“学生”类的属性，然后，再定义一些方法来对这些属性进行操作。这里，将这些操作属性的方法定义得很简单：每个属性都有相应的设置（setter）和获取（getter）方法，设置方法将传入的参数赋给对象的属性，而获取方法取得对象的属性。

源文件：Student.java

```
public class Student {
 // 定义属性
 String name;
 String sex;
 int grade;
 int age;

 // 定义属性“name”的设置方法
 public void setName(String _name) {
      name = _name;
 }

 // 定义属性“name”的获取方法
 public String getName() {
      return name;
 }

 // 定义属性“Sex”的设置方法
 public void setSex(String _sex) {
      sex = _sex;
 }

 // 定义属性“Sex”的获取方法
 public String getSex() {
      return sex;
 }

 // 定义属性“grade”的设置方法
 public void setGrade(int _grade) {
      grade = _grade;
 }

 // 定义属性“grade”的获取方法
 public int getGrade() {
      return grade;
 }

 // 定义属性“age”的设置方法
 public void setAge(int _age) {
      age = _age;
 }

 // 定义属性“age”的获取方法
 public int getAge() {
      return age;
 }
}
```

2.2.5 知识准备：属性的声明

类的定义中所包含的数据称为属性，也称为数据成员，比如 2.2.4 节示例中的 Student 类中定义了 4 个属性——name、sex、grade、age。属性声明的语法规则如下：

```
[< modifiers>] <data_type> < attr_name>;
```

其中：

- < modifiers>为修饰符，可用的有 public、private、protected、final、static 等，用于说明该属性的一些性质。
- <data_type>是该属性的数据类型，属性可以是 Java 基本类型的一种，如下面的示例中 MyClass 类中声明的属性 classID 等，也可以是任何类型的对象，可以通过引用它进行通信。例如 MyClass 类中声明的属性 myTeacher。
- <attr_name>是属性名称，属性名称的首字母一般采用小写方式。

```
MyClass.java
```

```
public class MyClass {
 int classID;
 String className;
 int studentNumber;
 Teacher myTeacher;

}
Teacher.java
public class Teacher {

 String name;
 int id;
 String course;
}
```

2.2.6 知识准备：方法的声明

类的定义中还可以包含方法的声明，Java 的方法决定了对象可以接收什么样的信息，方法的基本组成部分包括名称、参数、返回值和方法体。其语法规则如下：

```
< modifiers> <return_type> <name>([< argu_list>]) {
[< method body>]
}
```

其中：

- < modifiers>为修饰符，可用的有 public、private、protected、abstract、static 和 final，用于说明方法的属性。
- <return_type>是该方法的返回值类型，可以是任何合法的 Java 数据类型。
- <name>是方法名。
- <argu_list>是方法的参数列表，包括要传给方法的信息的类型和名称，如有多个参数，中间用逗号“,”分隔。
- <method body>是方法体，有 0 到多行 Java 语句。

下面在 2.2.5 节示例 MyClass 类中声明一个名为 ChangeTeacher 的方法，当大多数同学对老师不满意时，可以使用这个方法来换个给力的老师，并返回老师的名字。

```
public class MyClass {
 int classID;
 String className;
 int studentNumber;
 Teacher myTeacher;
 public String ChangeTeacher(Teacher t){
  System.out.print("change a teacher");
  this.myTeacher = t;
  return t.name;
 }
}
```

注意：

1．参数列表传递的实际上是引用，引用的对象类型必须和参数类型一致，不然编译器会报错。

2．Java 语言中类、方法及属性声明的次序并无严格要求。

2.2.7 知识准备：构造器（构造方法）

在 Java 程序中，每个类都必须至少有一个构造器（Constructor）。构造器是创建一个类的实例（对象）时需要调用的一个特殊的方法。

利用构造器可以产生一个类的实例，并且提供了一个地方用来定义创建类的实例时都需要执行的初始化（initialize）代码。构造器的定义语法如下：

```
<modifier>  <class_name>  ( [<argument_list>] )
{
  [<statements>]
}
```

从上面的语法中可以看出，它和类中的方法定义很类似：可以有访问修饰符 modifier、有自己的方法名称、有参数列表、有方法体，因此，可以将构造器当成一个特殊的方法（在许多资料里面，就是将 Constructor 称为构造方法），这个方法的作用就是用来产生一个类的实例。但是要认识到构造器与普通方法的区别，主要表现在以下 3 个方面。

（1）修饰符：和方法一样，构造器可以有任何访问的修饰：public、protected、private 或者没有修饰。不同于方法的是，构造器不能有以下非访问性质的修饰：abstract、final、native、static 或者 synchronized。

（2）返回值：方法能返回任何类型的值或者无返回值（void），构造器没有返回值，也不需要 void。

（3）命名：构造器使用和类相同的名字，通常为名词。而方法则不同，通常为描述一个操作的动词。按照习惯，方法通常用小写字母开始，而构造器通常用大写字母开始。

下面定义一个用来表示“美食”的类 Food。

源文件：Food.java

```
public class Food {
   private  String food_name ;
 public Food(String name){
      this.food_name = name;
 }
}
```

在 Food 类中，定义了一个属性 food_name，还定义了一个构造器，在构造器中传入一个字符串类型的参数，将参数值赋给属性 food_name。此时，就可以通过这个构造器来实例化这个类，如下所示。

```
Food myDinner = new Food("pizza");
```

这样，就得到了一个 food_name 名为“pizza”的实例，还可以再创建一个 food_name 名为“cola”的实例来搭配你的晚餐。

如果在程序中没有定义任何的构造器，则编译器将会自动加上一个不带任何参数的构造器。默认的构造器不带任何的参数，也没有“方法体”。

通过上面的示例，在 Food 类中定义了一个带一个参数的构造器。如果上面的 Food 类没有定义构造器，则编译器会自动加上一个构造器：

```
public class Food {
   private  String food_name ;
 public Food(){
 }
}
```

所以，这时可以用下面的语句来实例化这个类：

```
Food myDinner = new Food();
```

如果在程序中定义了构造器，则编译器将不再提供默认的构造器，即使定义的构造器同样没有参数。如果再使用默认构造器的话，编译器会报错。

2.2.8 知识准备：对象的创建和使用

正确声明了 Java 类之后，便可以在其他的类或应用程序中使用该类了。使用类是通过创建该类的对象并访问对象成员来实现的。对象成员是指一个对象所拥有的属性或可以调用的方法，现阶段可以理解为对象所属的类中定义的所有属性和方法。一个对象的声明实际上只是对该类的引用，需要使用 new+构造器（又称为构造方法）创建对象，此时存储器会给此对象相应的存储空间。当对象被成功创建后，可以使用“对象名.对象成员”的方式访问对象成员。

2.2.9 任务一：创建并引用一个对象

1．任务描述

创建一个美食类，并引用该类的对象，列出一天的食谱。

2．技能要点

（1）声明类、属性及方法。
（2）对象的创建和引用。

3．任务实现过程

（1）定义一个用于表示“食物”的类“Food”；在类中声明一个 food_name 属性；给 food_name 属性添加 get()、set()方法。

（2）在 Recipe 类中，通过默认构造方法创建 3 个 “Food”对象，通过 setFood_name(String name)方法给 food_name 赋值，并通过 getFood_name()方法获得 food_name 属性值，并输出。

源文件：Recipe.java

```
public class Recipe {
 public static void main(String args[]) {
      Food breakfast = new Food();
      breakfast.setFood_name("bread");
      Food lunch = new Food();
      lunch.setFood_name("nuddle");
      Food dinner = new Food();
      dinner.setFood_name("pizza");
      System.out.print("my breakfast is "+breakfast.getFood_name()+"\n");
      System.out.print("my lunch is "+lunch.getFood_name()+"\n");
      System.out.print("my dinner is "+dinner.getFood_name());
 }
}
class Food {
   private  String food_name ;

 public String getFood_name() {
      return food_name;
 }
 public void setFood_name(String foodName) {
      food_name = foodName;
 }
}
```

程序运行结果：

```
my breakfast is bread
my lunch is nuddle
my dinner is pizza
```

提示：

Java 中创建的对象并不用担心销毁问题。因为 Java 对象不具备和基本类型一样的生命周期，由 new 创建的对象，会一直保留下去，直到 Java 的垃圾回收器辨别出此对象不会再被引用，自动释放该对象的内存空间。

2.2.10 技能拓展任务：带参数构造器的声明与使用

1．任务描述

改写任务一的程序，使用带参数构造器，实现同样的输出结果。

2．技能要点

带参数构造器的声明和引用。

3．任务实现过程

（1）定义一个用于表示“食物”的类“Food”；在类中声明一个 food_name 属性；给 food_name 属性添加 get()、set()方法。

（2）给 Food 类添加两个构造方法，一个带参数，一个不带参数。

（3）在 Recipe 类中，通过带参数构造方法创建两个“Food”对象，通过 getFood_name()方法获得 food_name 属性值，并输出。

（4）在 Recipe 类中，通过不带参数构造方法创建一个“Food”对象，通过 setFood_name(String name)方法给 food_name 赋值并输出。

源文件：Recipe.java

```
public class Recipe {
 public static void main(String args[]) {
      Food breakfast = new Food("bread");
      Food lunch = new Food("noddle");
      Food dinner = new Food();
      dinner.setFood_name("pizza");
 System.out.print("my breakfast is "+breakfast.getFood_name()+"\n");
      System.out.print("my lunch is "+lunch.getFood_name()+"\n");
      System.out.print("my dinner is "+dinner.getFood_name());
 }
}
class Food {
   private  String food_name ;

 public String getFood_name() {
      return food_name;
 }
 public void setFood_name(String foodName) {
      food_name = foodName;
 }
 public Food(String name){
      this.food_name = name;
 }
 public Food(){}
}
```

程序运行结果：

```
my breakfast is bread
my lunch is nuddle
my dinner is pizza
```

注意：

因为类“Food”定义了一个带参数的构造器，所以，编译器不会给它加上一个默认的不带参数的构造器，此时，如果还试图使用默认的构造器来创建对象，将不能通过编译。如需要不带参数构造器，则自行定义使用。

2.3 信息的封装和隐藏

2.3.1 知识准备：信息的封装

封装指的是将对象的状态信息（属性）和行为（方法）捆绑为一个逻辑单元的机制。

Java 中通过将数据封装声明为私有的（private），再提供一个或多个公开的（public）方法实现对该属性的操作，以实现下述目的：

- 隐藏一个类的属性和实现细节，仅对外公开接口，控制在程序中属性的可读和修改的访问级别。

- 增强安全性，防止对封装数据的未经授权的访问。使用者只能通过事先定制好的方法来访问数据，可以方便地加入控制逻辑，限制对属性的不合理操作。
- 有利于保证数据的完整性。
- 便于修改，增强代码的可维护性。
- 实现封装的关键是不要让方法直接访问其他类的属性，程序应该只能通过指定的方法与对象交互数据。封装赋予对象“黑盒”特性，这是实现重用性和可靠性的关键。

2.3.2 知识准备：信息的隐藏

如果允许用户对属性直接访问，可能会引起一些不必要的问题，如声明了一个 Group 类表示一个程序开发小组，由属性 number 来记录小组成员数。如果允许程序随意给 number 属性赋值，将值设置为 1000，虽然这在语法上没有问题，但是，我们知道一个程序小组不可能有这么多的编程人员。如果在程序的其他部分用到了这个 number 属性，可能出现问题。因此，应该将属性定义为私有的（private），只有类本身才可以访问这个属性，外部程序或者其他类不能访问它。可以定义一些 public 或 Default 方法来访问这些属性，在方法中加入一些逻辑判断的方法来操作属性。将 number 的属性值设置为 2~100 之间，小于 2 人时计做两人，大于 100 人时计做 100 人。示例如下：

源文件：Group.java

```
public class Group {

 private int number;
 public void setNumber(int s_number) {
      if (s_number > 100) {
           this.number = 100;
      } else if (s_number< 2) {
           this.number = 2;
      } else {
           this.number = s_number;
      }
 }

}
```

2.4 Java 源文件结构

Java 语言的源程序代码由一个或多个编译单元（Compilation unit）组成，每个编译单元只能包含下列内容（空格和注释除外），如表 2-1 所示。

表 2-1 Java 源文件结构

结构	作用	要求
package 语句	声明类所在的包	0 或 1 个，必须放在文件开始
import 语句	从特定的包引入类	0 或多个，必须放在所有类定义之前
public classDefinition	公共类定义部分	0 或 1 个
classDefinition	非公共类定义部分	0 或多个
interfaceDefinition	接口定义部分	0 或多个

注意：

（1）需要特别注意的是，Java 是严格区分大小写的。

（2）定义为 public 的类名必须与 Java 文件名称完全一致，每个 Java 源文件只能有一个定义为 public 的类，但可以有几个非 public 的类名。

2.4.1 知识准备：package 语句

在大型项目开发中，为了避免类名的重复，经常使用“包”来组织各个类。例如，公司用开发了一个类 Food，公司乙也开发了一个类 Food，现在需要同时用到这两个类，为了将这两个类区分开来，将这两个文件放在两个不同的目录下。那么，在使用的时候如何准确地找到需要的类呢？这就需要用到“包（package）”这个概念。

在 Java 程序中可以用 package 来指明这两个类的引用路径：将甲公司开发的 Student 类放到一个包（package）中，而将乙公司开发的 Student 类放到另一个包（package）中。为了避免不同的公司之间类名的重复，Sun 建议使用公司 Internet 域名的倒写来当做包名，例如，使用域名 farsight.com.cn 的倒写 cn.com.farsight 来作为包的名称。如果现在有一个名为 Student 的类，将它放到目录 cn\com\farsight 下，然后再在程序中加入如下语句：

```
package cn.com.farsight;
```

这样，这个类就可以和其他的名称为 Student 的类区分开了。

package 语句的基本语法如下：

```
package  <top_pkg_name> [ .<sub_pkg_name>] * ;
```

在同一个项目组中，经常用功能或模块的名称来作为子包名，例如，现在开发一个学员管理系统，一个用于华清远见的管理模块，一个用于清华附中的管理模块，这两个功能模块中均有一个 Student 的类，这时可以将华清远见管理模块中使用的 Student 放到 cn\com\farsight\college 目录下，而将清华附中管理模块的 Student 类放到 cn\com\farsight\school 中，这样，这两个文件就不会冲突了。

注意，当将开发的类放到包中时，必须将类的源文件放到与包名的元素相一致的目录结构中，就如上面的学生类 Student，如果在文件中加入了 package 语句：package cn.com.farsight，则必须将这个 Student 类文件放到 cn\com\farsight 目录下。在磁盘目录结构中，使用“\”（Windows）或“/”（Unix/Linux/BSD）来分隔各个层级的目录，而在 Java 类文件中，使用“.”来分隔包的层次。

Java 的核心包都放在 Java 包及其子包中，而将很多扩展包放在 javax 包及其子包中。在 Java 核心包中，也有一些重复的类名，就是通过不同的包来区分的，例如，在 java.sql 包中，有一个 Date 类，而在 java.util 中，也有一个 Date 类。通过将它们放在不同的包中，就可以区分这两个类。

一般来说，如果在程序中使用 package 将程序打包，则将程序放到对应的目录下，这样，才不违背 package 的设计初衷。如果将类的源文件（.java 文件）都放到同一个路径下，只是将编译后的 class 文件放到不同的目录下，上面提到的文件名冲突的问题依然存在（同一个路径下不允许两个文件同名），因此，需要将源文件保存在不同的路径中。这时，可以使用如下命令来编译文件：

```
javac cn\com\farsight\college\Student.java
```

在 cn 目录的上一级目录下执行上述命令。

如果程序是一个带 main 方法的应用程序，可以在 cn 目录的上一级目录下执行这个文件，命令如下：

```
java cn.com.farsight.college.Student
```

注意编译和执行时的分隔符。因为在编译时，需要指明的是文件的路径，所以使用“\”来分隔；而在执行类文件时，需要指明的是包名称，所以使用“.”来分隔。

下面来看一个使用 package 的例子。

源文件：Student.java

```
package cn.com.farsight;

public class Student {
```

```
    // 定义属性
    private String studentId;

    // 定义属性“studentId”的设置方法
    public void setStudentId(String student_Id) {
         studentId = student_Id;
    }

    // 定义属性“studentId”的获取方法
    public String getStudentId() {
         return studentId;
    }
}
```

这个程序最好放在路径 cn\com\farsight 下，然后再在 cn 的上一层路径执行如下的指令编译程序：

```
javac cn\com\farsight\Student.java
```

而被编译后的 class 文件必须放置在 cn\com\farsight 路径下，这样才可以成功地被其他程序引用。

如果 Student 是一个应用程序（有一个 main()方法），那么，为了执行这个程序，必须在“cn”的上一级目录执行如下的指令：

```
java cn.com.farsight.Student
```

2.4.2 知识准备：import 语句

在编译器定位你所创建的类所访问的其他类的过程中，包（package）扮演了重要的角色。当编译器碰到一个类对另一个类的引用时，它会在当前的包中和设置的 CLASSPATH 中寻找这个类，以检查这个类是否能在这些路径中找到。

import 语句应该出现在 package 语句之后（如果有的话），类的定义之前。package 语句只能有一个，但是 import 语句可以有多个。

可以使用 import 来引入包中的一个类，也可以用 import 来引入指定包中的所有类。

import 语句的基本语法如下：

```
import  <pkg_name>[.<sub_pkg_name>].<class_name>;
```

或

```
import <pkg_name>[.<sub_pkg_name>].*;
```

- 引入一个类：

```
import cn.com.farsight.college.Student
```

- 引入指定包中的所有类使用通配符“*”：

```
import cn.com.farsight.college.*
```

这样就可以引入包 cn.com.farsight.college 中的所有类。

这两种方式对于引入相应的类并没有什么区别。但是，如果只是需要一个包中有限的几个类，建议采用第一种方式（写明引入的类名）会让人一目了然。另外，需要注意的是，通过 import 引入包中的类的时候，它并不会递归地执行引入动作，比如，通过下面的语句：

```
import cn.com.farsight.*;
```

引入了 cn.com.farsight 这个包中的所有类，但它并不会引入 cn.com.farsight.school 和 cn.com.farsight.colleage 中的类。要使用这两个包中的类，还需要将它们使用“import”语句分别引入。

Java 编译器默认为所有的 Java 程序引入了 JDK 的 java.lang 包中所有的类（import java.lang.*;），其中定义了一些常用类：System、String、Object、Math 等。因此可以直接使用这些类而不必显式引入。但使用其他非无名包中的类则必须先引入、后使用。

另外，可以通过在类名前加上不同的限制符，如 public 等来控制类的适用范围。关于这些限制符及相应的使用范围，将在后续章节介绍。

注意：

使用 import 并不会将相应的类或者包加载到 class 文件（或者 Java 源文件），也不会包含到 Java 源文件（或者 class 文件）中，它的作用仅仅是对需要用到的类进行定位（location）。它表示程序中用到某个类的时候，如果没有在类前指定包名，应该到当前目录或者 import 指定的包中去寻找（注意 java.lang 包是默认引入的）。

2.4.3 任务二：package 语句和 import 语句实例

1. 任务描述

编写一个 Parent 类，在构造方法中输出“我是 parent”，放在 parent 包中，编写一个 Child 类，在构造方法中输出“我是 child”放在 child 包中，child 包是 parent 包的子包。使用 package 语句生成包，并使用 import 引用子包。

2. 技能要点

- 掌握包的创建和使用。
- 通过给包命名创建包的层次。

3. 任务实现过程

（1）在 scr 默认包中创建 parent 包，在此包中创建 Parent 类，不带参数，在构造方法中输出“I am a parent”。

（2）创建 parent 的子包 parent.child 包，在此包中创建 Child 类，不带参数，在构造方法中输出“I am a child ”。

（3）在 Parent 类中使用 import 语句引入 parent.child 包，并创建 Parent 类和 Child 类对象，实现输出。

```
Parent.java
package parent;

import parent.child.Child;

public class Parent {
 public Parent(){
      System.out.print("I am a parent");
 }
 public static void main(String[] args) {
      Child c = new Child();
      Parent p = new Parent();
 }
}
Child.java
package parent.child;

public class Child {

 public Child(){
      System.out.print("I am a child \n");
 }
}
```

运行结果：

```
I am a child
I am a parent
```

2.5 JDK 中常用的包

在 JDK 核心类库中，提供了几千个类，通过这些类，可以基本实现用户界面设计、输出/输入、网络和日期等基本的目的。这些类都按照它们的功能、作用放在不同的包（package）中，作为 Java 开发人员，应该熟悉这些包，并掌握下面的这些比较常用的包。

- java.lang——包含一些 Java 语言的核心类，如 String、Math、Integer、System 和 Thread，提供常用功能。使用这个包中的类可以不用“import”语句来显式引入。在默认情况下，编译器会将这个包自动引入到任何的 Java 程序中，所以，这个包中的类可以直接在程序中使用。
- java.net——包含执行与网络相关的操作的类。
- java.io——包含能提供多种输入/输出功能的类。
- java.util——包含一些实用工具类及数据结构类，如定义系统特性、使用与日期日历相关的函数、集合、堆栈等。
- java.sql——包含用于访问数据库的类。

虽然 JDK 中已经定义了大量的类供开发人员使用，但是，还是需要在此基础上定义自己的类来实现我们的目的，当然，在定义自己的类时，可以使用类库中的类。

2.6 本章小结

本章从面向对象的概念出发，介绍了程序设计种类，面向对象程序设计的特征和常用术语，使读者对面向对象的概念和特点有初步了解。介绍了面向对象程序设计中的对象、类、属性等重要概念，使读者可以掌握类、属性、方法的定义和声明，构造器的概念及使用；介绍了对象的创建和使用，使读者可以掌握类与对象的创建和使用方法，并了解到通过类的定义来实现信息隐藏和封装；介绍了 Java 源文件的结构，使读者可以掌握 package 和 import 的用法；熟悉 JDK 中常用的包。

课后练习题

一、选择题

1．下面的说法错误的是（　　）。
A．在 Java 中用引用来操纵对象
B．在 Java 中类和对象的关系是父子关系
C．一个类中可以有多个构造器
D．构造器用来申请内存空间和初始化变量
2．对于构造方法，下列叙述错误的是（　　）。
A．构造器必须与类名相同
B．构造器的返回类型只能是 void 型，且书写格式是在方法名前加 void 前缀
C．类的默认构造器是无参构造器
D．创建新对象时，系统会自动调用构造器
3．下面说法错误的是（　　）。
A．用 package 语句可以导入一个包
B．用 import 语句可以导入一个包中的类
C．package 语句必须是文件中的第一行非注释代码
D．import 语句可以放在代码中的任何地方
4．下面包中的类不需要显式导入就可以直接使用的是（　　）。
A．java.net
B．java.io
C．java.util
D．java.lan
5．下面说法正确的是（　　）。
A．一个.java 文件中可以有多个 public 类
B．一个.java 文件中可以有多个类，编译后生成一个 class 文件

C．一个.java 文件中有几个类，编译后就生成几个 class 文件

D．一个 Java 类中必须有一个 main 方法

二、简答题

1．简述类和对象的区别和联系。

2．简述类的构造器。

3．简述什么是封装及封装的意义。

三、编程题

1．汽车有 3 个属性（型号、颜色、车牌号）编写一个 Car 类来描述它，要能用带参数的构造器赋值。2．编写一个计算器类，里面有两个操作数及加、减、乘、除 4 个方法。编写应用程序生成该类的对象，并使用它的方法进行计算。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218