第 5 章　接口和包

学习了 Java 继承，理解了父类和子类之间的隐藏和覆盖关系后，就理解了 Java 运行的多态性。但 Java 不支持多继承性，也就是一个类只能有一个父类。单继承性使 Java 易于管理，因为从编程的复杂度来讲，多继承会提高程序的耦合性，而且对程序的易用性也有影响，但这与现实世界可能出现多继承相矛盾，因此，Java 提供了接口。

例如，有两个 VCD 产品，其功能不尽相同，或完全不同，但其面板完全相同，即不同功能的 VCD 产品使用了相同的面板，从 Java 语言的角度，这个面板就是接口，而且两个 VCD 产品还可以有相同的外壳，这个外壳也是接口，这样，两个 VCD 产品就使用了两个接口。

大多数人认为，接口的意义在于实现多继承，其实接口和继承是完全不同的东西。对象通过接口对外提供服务。在面向对象的范畴中，接口是一个抽象的概念，是指系统对外提供的所有服务。系统的接口描述系统能够提供哪些服务，但不包含服务实现的细节，具体的实现细节在类中定义。没有继承关系的类可以实现相同的接口。

站在使用者的角度，对象中所有向使用者公开的方法的声明构成了对象的接口。接口是实现系统之间松耦合的有力手段，接口还提高了系统的可扩展性。

当应用很复杂，Java 文件很多，这时候就需要对这些文件做一下归档和管理，Java 就提供了包，包类似于操作相同的文件夹。

- Java 接口的定义与继承
- Java 接口的实现
- Java 接口与类的关系，特别是接口与抽象类的区别
- Java 包的声明、创建和使用方法
- JAR 命令的用法

5.1　基础知识

5.1.1　接口

Java 不支持多继承性，也就是一个类只能有一个父亲。单继承性使 Java 易于管理，但为了克服单继承的缺点，Java 使用了接口，一个类可以实现多个接口。

1. 接口的定义

接口的定义包括接口声明和接口体。Java 使用关键字 interface 声明接口，格式如下：

[修饰符] interface 接口名 [extends 父接口列表]//接口声明

```
{
  接口体
}
```

其中：

① Java 语言接口的定义包含接口的声明和接口体两部分。接口的声明确定接口的名字、访问限制及继承的父接口等；接口体由包括在花括号内的变量和方法组成，变量定义在方法的前面。

② 变量默认定义为 public static final 的，即最终常量，方法默认定义为 public abstract，即接口中只能包含抽象方法，默认修饰符在定义时可以省略。

③ interface，接口定义的关键字。接口名是接口的名字，接口名必须是合法的 Java 标识符。

④ extends 子句表示接口的继承，与类的继承不同的是一个接口可有多个父接口，用逗号隔开。与子类继承抽象父类相似，类如果实现了一个接口，那么必须实现接口里面的所有抽象方法，否则该类要被定义为抽象类。

⑤ 接口没有构造方法，不能被实例化。

2. 接口的实现

一个类通过使用关键字 implements 实现一个或多个接口。如果使用多个接口，用逗号隔开接口名，格式如下：

```
[类修饰符]class 类名[extends 父类名] [ implements 接口列表]{//类声明
    ……
}
```

如果一个类实现了某个接口，那么这个类必须实现该接口的所有方法，即为这些方法提供方法体。需要注意的是在类中实现接口的方法时、方法的名字、返回类型，参数个数及类型必须与接口中的完全一致。特别注意，接口中的方法被默认是 public 的，所以类在实现接口方法时，一定要用 public 来修饰方法。

例如：

```
interface Information{    //定义接口
    String college="清华大学"; //接口常量
    void setName(String name); //接口方法
    String getName();
}
interface Course{
    String professional="计算机应用";
    void setCourse(String course);
    String getCourse();
}
//实现接口
public class Teacher implements Information,Course{
    String name,course;
    //重写接口方法，注意不要丢掉 public
    public void setName(String name){
        this.name=name;
    }
    public String getName(){
        return "教师姓名："+name;
    }
    public void setCourse(String course){
        this.course=course;
    }
    public String getCourse(){
        return "主讲课程："+course;
```

```
    }
    public static void main(String args[]){
        Teacher t=new Teacher();
        t.setName("张强");
        t.setCourse("Java 程序设计");
        System.out.println(t.getName());
        //使用接口常量
        System.out.println("所在院校："+t.college);
        System.out.println("技术专业："+t.professional);
        System.out.println(t.getCourse());
    }
}
```

3. 接口的继承

接口支持多继承，子接口可以继承父接口成员，当一个非抽象类实现一个子接口时，它必须实现所有接口（包含父接口）的所有方法。

例如：

```
interface Information{    //定义接口
    String college="清华大学"; //接口常量
    void setName(String name); //接口方法
    String getName();
}
interface Course extends Information{ //接口的继承
    String professional="计算机应用";
    void setCourse(String course);
    String getCourse();
}
//实现接口
public class Teacher implements Course{
    String name,course;
    //实现父接口的方法
    public void setName(String name){
        this.name=name;
    }
    public String getName(){
        return "教师姓名："+name;
    }
    public void setCourse(String course){
        this.course=course;
    }
    public String getCourse(){
        return "主讲课程："+course;
    }
    public static void main(String args[]){
        Teacher t=new Teacher();
        t.setName("张强");
        t.setCourse("Java 程序设计");
        System.out.println(t.getName());
        //使用接口常量
        System.out.println("所在院校："+t.college);
        System.out.println("技术专业："+t.professional);
        System.out.println(t.getCourse());
    }
}
```

4. 接口回调

接口回调是指可以把实现某一接口的类创建的对象的引用赋给该接口声明的接口变量

中，那么该接口变量就可以调用被类实现的接口中的方法。

例如：

```
interface Information{    //定义接口
    String college="清华大学"; //接口常量
    void setName(String name); //接口方法
    String getName();
}
interface Course{
    String professional="计算机应用";
    void setCourse(String course);
    String getCourse();
}
//实现接口
public class Teacher implements Information,Course{
    String name,course;
    //实现接口的方法
    public void setName(String name){
        this.name=name;
    }
    public String getName(){
        return "教师姓名："+name;
    }
    public void setCourse(String course){
        this.course=course;
    }
    public String getCourse(){
        return "主讲课程："+course;
    }
    public static void main(String args[]){
        Teacher t=new Teacher();
        Information info=t; //接口回调
        Course c=t;
        //调用 Teacher 类实现的方法
        info.setName("张强");
        c.setCourse("Java 程序设计");
        System.out.println(info.getName());
        //使用接口常量
        System.out.println("所在院校："+t.college);
        System.out.println("技术专业："+t.professional);
        System.out.println(c.getCourse());
    }
}
```

5. 接口和抽象类

接口和抽象类虽然有相似之处，即它们都可能有抽象方法。但却有本质的不同。首先在代码编写方面，它们的语法要求存在差异。从这个角度讲，接口是纯粹抽象的类，而抽象类是一般类到接口之间的过渡。表 5.1 列出了接口和抽象类在语法方面的不同。

表 5.1　接口和抽象类的区别

接口	抽象类
静态常量	一般变量、常量、静态变量、静态常量
抽象方法	成员方法、静态方法、抽象方法、抽象静态方法
使用关键字interface	使用关键字 abstract

5.1.2　包

1. 包的概念

Java 系统中存在大量的类和接口，为了更好地组织它们，Java 提供了包机制，包是类和接口的组织方式，包类似于操作系统中的文件夹，将相近功能的类打入同一包，将不同功能的类打入不同的包内，实现类的分类组织。每个 Java 类都被包含在一个包中，如果在源文件中没有给出包名，系统会给每一个 Java 源文件创建一个无名包。

2. 包的声明

使用关键字 package 声明一个包，格式如下：

package 包名;

例如：package people;

声明包语句必须添加在源程序的第一行，表示该程序文件声明的全部类都属于这个包，包的名字一般都是由小写单词组成。

3. 包的创建

（1）手动创建包

首先编译源文件，然后在当前目录创建一个文件夹，文件夹的名称与包名一致，然后将生成的字节码文件放入该文件夹中，此时运行 Java 程序命令修改为：java 包名.类名。

例如：

```
package people; //声明包
interface Information{
    String college="清华大学";
    void setName(String name);
    String getName();
}
public class People implements Information{
    protected String name;
    protected int age,number;
    public People(int age,int number){
        this.age=age;
        this.number=number;
    }
    public void setName(String name){
        this.name=name;
    }
    public String getName(){
        return "姓名："+name;
    }
    public static void main(String args[]){
        People p=new People(40,10000);
        p.setName("张强");
        System.out.println("工号："+p.number);
        System.out.println(p.getName());
        System.out.println("年龄："+p.age);
    }
}
```

手动创建文件夹 people，将编译好的.class 文件复制到当前目录的 people 文件夹下，运行 People.java 程序的命令为：java people.People

（2）自动创建包

Java 中使用 javac -d 命令将类文件生成到指定的路径，格式如下：

……>javac -d 类文件生成的路径 源文件名

如果将类文件生成在当前路径的包中，“类文件生成的路径”可以使用“.”代替。

编译上面程序的命令：javac -d . People.java

解释上面程序的命令：java people. People

4. 包的使用

① 如果某个类不在当前类的目录下，则使用包名.类名的方式使用类，例如，如果 People 类不在当前类目录下，其目录为 people，则使用 People 类的方法为：people.People=new people.People(40,10000);

② 如果要在程序中直接引用类名，则必须使用 import 语句，例如：

```
import people.People;
```

如果要导入包中所有的类，可以使用“*”，例如：

```
import people.*;
```

import 语句应该在程序的 package 语句后，例如：

```
package teacher;
import people.People; //导入 people 包中的类 People
```

5. Java 提供的常用包

- java.lang 包：java 的核心类库，包含了运行 java 程序必不可少的系统类，如基本数据类型、基本数学函数、字符串处理、线程、异常处理类等，系统缺省加载这个包，无需使用 import 导入。
- java.io 包：java 语言的标准输入/输出类库，如基本输入/输出流、文件输入/输出、过滤输入/输出流等。
- java.util 包：包含大量工具类/接口和集合框架类/接口，如 Arrays、List、Set、处理时间的 date 类、处理数组的 Vector 类以及 stack 和 HashTable 类。
- java.awt 包：构建图形用户界面（GUI）的类库，低级绘图操作 Graphics 类，图形界面组件和布局管理如 Checkbox 类、Container 类、LayoutManager 接口等，以及界面用户交互控制和事件响应，如 Event 类。
- java.swing 包：含 Swing 图形用户界面编程的相关类/接口，这些类可以构建平台无关的 GUI 程序。
- java.applet 包：Java 语言编写的一些小应用程序，这些程序是直接嵌入到页面中，由支持 Java 的浏览器（IE 或 Netscape）解释执行能够产生特殊效果的程序。
- java.net 包：实现网络功能的类库有 Socket 类、ServerSocket 类。
- java.sql 包：实现 JDBC 的类库。

6. JAR 文件包

JAR 文件就是 Java Archive File，顾名思义，它的应用是与 Java 息息相关的，是 Java 的一种文档格式。JAR 文件非常类似 ZIP 文件——准确的说，它就是 ZIP 文件，所以叫它文件包。JAR 文件与 ZIP 文件唯一的区别就是在 JAR 文件的内容中，包含了一个 META-INF/MANIFEST.MF 文件，这个文件是在生成 JAR 文件的时候自动创建的。

JAR 命令的用法如下：

jar {ctxu}[vfm0M] [jar-文件] [manifest-文件] [-C 目录] 文件名列表

其中：

- {ctxu}是 jar 命令的子命令，每次 jar 命令只能包含 ctxu 中的一个，常用选项如表 5.2 所示。

表 5.2　{ctxu}常用选项含义表

选项	含义
-c	创建新的 JAR 文件包
-t	列出 JAR 文件包的内容列表
-x	展开 JAR 文件包的指定文件或者所有文件
-u	更新已存在的 JAR 文件包（添加文件到 JAR 文件包中）

- [vfm0M]中的选项可以任选，也可以不选，它们是 jar 命令的选项参数，常用选项如表 5.3 所示。

表 5.3　[vfm0M]常用选项含义表

选项	含义
-v	生成详细报告并打印到标准输出
-f	指定 JAR 文件名，通常这个参数是必需的
-m	指定需要包含的 MANIFEST 清单文件
-0	只存储，不压缩，这样产生的 JAR 文件包会比不用该参数产生的体积大，但速度更快
-M	不产生所有项的清单（MANIFEST）文件，此参数会忽略-m 参数

- [jar-文件]即需要生成、查看、更新或者解开的 JAR 文件包，它是-f 参数的附属参数。
- [manifest-文件]即 MANIFEST 清单文件，它是-m 参数的附属参数。
- [-C 目录]表示转到指定目录下去执行这个 jar 命令的操作。它相当于先使用 cd 命令转至该目录下，再执行不带-C 参数的 jar 命令，它只能在创建和更新 JAR 文件包的时候可用。
- 文件名列表指定一个文件/目录列表，这些文件/目录就是要添加到 JAR 文件包中的文件/目录。如果指定了目录，那么 jar 命令打包的时候会自动把该目录中的所有文件和子目录打入包中。

例如，将本章的第一个例子生成的 class 文件制作成 jar 包，名称为 myjar.jar，且生成详细报告，并在屏幕上显示出来，命令如下：

```
Jar cvf myjar.jar Teacher.class Course.class Information.class
```

5.2　实践

5.2.1　实践目的

1）掌握接口的定义与使用，比较接口和抽象类之间的异同。
2）掌握包的创建与使用，了解 Java 系统提供的常用包。

5.2.2　实践要求

1）编写 Java 应用程序，完成接口的实现。
2）编写 Java 应用程序，体会抽象类和接口的区别。
3）编写 Java 应用程序，完成接口的继承，体会接口继承的原理。

4）将程序中的接口和类分别定义在不同的包中，体会包的使用。

5.2.3 实践内容

1. 基本实践

1）定义一个抽象父类 Information，它包含两个抽象方法 getName()、getNumber()，从 Information类派生出 Student 和 Teacher 类，这两个类都用 getName()、getNumber()方法获得姓名和工号信息，编译并运行程序，体会抽象类和抽象方法的使用。记录程序运行结果。

```
public class UseInfo{
    public static void main(String[] args){
        Student s =new Student("黎明",101);
        Teacher t = new Teacher("王芳",2001);
        System.out.println("学生所在学校是："+s.college);
        System.out.println("教师所在学校是："+t.college);
        System.out.println("学生姓名："+s.getName()+","+"学生学号："+s.getNumber());
        System.out.println("教师姓名："+t.getName()+","+"教师工号："+t.getNumber());
    }
}
abstract class Information{
     public static final String college="清华大学";
     public abstract String getName(); //抽象方法
     public abstract int getNumber();//抽象方法
}
class Student extends Information {
    String name;
    int number;
    public Student(String name,int number){
        this.name=name;
        this.number=number;
    }
    public String getName(){   //获取姓名
        return name;
    }
    public int getNumber(){ //获取工号
        return number;
    }
}
class Teacher extends Information {
    String name;
    int number;
    public Teacher(String name,int number){
        this.name=name;
        this.number=number;
    }
    public String getName(){
        return name;
    }
    public int getNumber(){
        return number;
    }
}
```

2）为基本实践 1）添加一个接口 Course，在 Teacher 类中添加一个字符串变量 course，代码如下：

```
interface Course{
    String professional="计算机应用";
    public void setCourse(String course);//设置教师教授课程为参数 course
```

```
    public String getCourse();//返回教师教授课程
}
```

要求 Teacher 类必须实现该接口，写出程序，并记录运行结果。

3）为基本实践 1）Information类添加 getCollege()方法，代码如下：

```
abstract class Information{
    public static final String college="清华大学";
    public abstract String getName(); //抽象方法
    public abstract int getNumber();//抽象方法
    public String getCollege(){
    return college;
    }
}
```

编译并运行程序，记录程序运行结果，体会抽象类中方法种类。

4）修改基本实践 3）Information类 college 变量的控制符，删除 static final，编译并运行程序，记录程序运行结果，体会抽象类中变量种类。

5）修改基本实践 4），将 Information类 college 变量和 getCollege()均修改为 static 类型，编译并运行程序，记录程序运行结果，体会抽象类中静态变量和静态方法的使用方法。

6）将基本实践 1）中的抽象类 Information更改为接口 Information，修改程序，并记录运行结果。试着修改接口中变量和方法的控制符，调试程序，记录运行结果，写出错误原因，体会接口和抽象类的不同。

7)修改基本实践 2),要求接口 Course 继承接口 Information,Teacher 类实现了接口 Course，编写程序，编译并运行程序，记录程序运行结果，体会接口的继承。

8）将基本实践 7）的接口 Information、Course 和类 Teacher 和 Student 分别定义在不同的包中，体会包的使用。

```
package information;
public interface Information{
    public static final String college="清华大学";
    public abstract String getName(); //抽象方法
    public abstract int getNumber();
}
```

Course.java：

```
package course;
import information.Information;
public interface Course extends Information{
    String professional="计算机应用";
    void setCourse(String course);
    String getCourse();
}
```

Teacher.java：

```
package teacher;
import course.Course;
public class Teacher implements Course {
    String name,course;
    int number;
    public Teacher(String name,int number){
        this.name=name;
        this.number=number;
    }
    public String getName(){
        return name;
    }
    public int getNumber(){
        return number;
```

```
        }
        public String getCourse(){
            return course;
        }
        public void setCourse(String course){
            this.course=course;
        }
}
packTest.java:
import information.Information;
import course.Course;
import teacher.Teacher;
public class packTest{
    public static void main(String[] args){
        Information s =new Student("黎明",101);   //接口回调
        Course t = new Teacher("王芳",2001);
        System.out.println("学生所在学校是："+s.college);
        System.out.println("教师所在学校是："+t.college);
        System.out.println("学生姓名："+s.getName()+","+"学生学号："+s.getNumber());
        System.out.println("教师姓名："+t.getName()+","+"教师工号："+t.getNumber());
        t.setCourse("Java 程序设计");
        System.out.println("主讲课程："+t.getCourse());
    }
}
 class Student implements Information {
    String name;
    int number;
    public Student(String name,int number){
        this.name=name;
        this.number=number;
    }
    public String getName(){      //获取姓名
        return name;
    }
    public int getNumber(){       //获取学号
        return number;
    }
}
```

试着修改变量和方法的访问控制符，调试程序，体会包对变量和方法访问权限的限制。

9）使用 jar 命令，将基本实践 8）的 class 文件制作成一个 JAR 文件。

2. 技术提高

1）定义接口 Information，代码如下：

```
public interface Information{
     public static final String college="清华大学";   //常量
     public abstract String getName(); //抽象方法
     public abstract int getNumber();
}
```

定义 Student 类和 Teacher 类实现该接口，编写应用程序输出他们的姓名、工号。记录程序代码及运行结果。

2）创建一个抽象类 ScoreInfo，定义一个静态常量 score，再创建两个抽象类 StudentInfo 和 TeacherInfo，继承 ScoreInfo 并实现实践 1）中接口 Information，抽象类 StudentInfo 包含一个抽象方法获取成绩 getScore()；抽象类 TeacherInfo 包含一个方法给出成绩 setScore(String score)，只有教师给出成绩后，学生才能获取成绩，代码如下：

```
abstract class ScoreInfo{
    public static int score;
```

```
}
abstract class TeacherInfo extends ScoreInfo implements Information{
    public abstract void setScore(int score);
}
abstract class StudentInfo extends ScoreInfo implements Information{
    public abstract int getScore();
}
```

试着使用 Student 类和 Teacher 类分别继承抽象类 StudentInfo 和 TeacherInfo，最后在 main 方法中输出学生的成绩信息。记录程序代码及运行结果。

3）修改技术提高 2），将接口 Information 定义在 information 包中，将 3 个抽象类定义在 info 包中，Student 类和 Teacher 类分别定义在 student 包和 teacher 包中，试着修改程序，输出学生的成绩信息。记录程序代码及运行结果。

4）使用 jar 命令，将技术提高 3）的文件制作成一个 JAR 文件。

3. 技术综合

1）声明一个包 student，包含一个接口 Information、一个抽象类 ClassInfo 和一个类 Student。接口 Information 中包含两个方法 getName()、getNumber()，抽象类 ClassInfo 中也存在两个抽象方法 setClass (String)和 getClass()。类 Student 实现接口 Information 并且继承了抽象类 ClassInfo，编写应用程序输出学生的学号、姓名、班级。

2）在技术综合 1）的基础上再声明一个包 teacher，存在一个抽象类 CollegeInfo，一个教师类 Teacher。抽象类 CollegeInfo 中存在两个方法 setCollege(String)和 getCollege()。要求 Teacher 类实现接口 Information 和继承抽象类 CollegeInfo，编写应用程序输出教师的学校、工号和姓名。

4. 常见错误与处理

错误 1：使用抽象类 Test 时出现：Test is abstract, cannot be instantiated。

处理办法：错误原因是对抽象类进行了实例化。创建抽象类的对象时，不能使用 new 关键字实例化。

错误 2：类 A 为抽象类 Test 的子类，Test 类中有一个抽象方法 f()，编译时出现 A is not abstract and does not override abstract method f()。

处理办法：抽象类的子类必须实现抽象父类全部的抽象方法，除非该子类也是抽象类。

错误 3：使用接口编译时出现：attempting to assign weaker access privileges; was public。

处理办法：接口中的成员默认权限都是 public 的，作为接口的实现类，实现接口中的抽象方法时，其权限不能低于 public，处理的方式就是在方法前加上 public。

错误 4：程序中使用了 JFrame 类，编译时出现 cannot find symbol; symbol: class JFrame。

处理办法：Java 系统中，java.lang 包中的类被自动导入，其他包中的类必须使用 import 语句导入，比如 JFrame 类在 javax.swing 包中，应在程序中加入 import javax.swing.JFrame。

5.3　综合练习

一、选择题

1．某个类相对主目录的路径是 demo1\ demo2\ demo3，那么类包的声明语句是（　　）。

A．package demo2.demo3　　B．Package demo1.demo2.demo3

C．Package Demo1.Demo2.Demo3　　D．package demo1.demo2.demo3

2．接口可以继承多个父接口，父接口之间可以使用（　）符号隔开。

A.. B.， C.： D．/

3．下列叙述中，错误的是（　）。

A．Java 中，方法的重载是指多个方法可以共享同一个名字

B．Java 中，用 abstract 修饰的类称为抽象类，它不能实例化

C．Java 中，接口是不包含成员变量和方法实现的抽象类

D．Java 中，构造方法可以有返回值

4．下列选项中，用于定义接口的关键字是（　）。

A．interface B．implements C．abstract D．class

5．下列选项中，用于实现接口的关键字是（　）。

A．interface B．implements C．abstract D．class

6.现有类 ClassA 和接口 InterB，以下描述中表示类 ClassA 实现接口 InterB 的语句是(　)。

A．class ClassA implements InterB B．class InterB implements ClassA

C．class ClassA extends InterB D．class InterB extends ClassA

7．下列（　）导入包的语法正确。

A．import util.Date; B．import java.util.*;

C．package java.util.*; D．package java.util.Date ;

8．包的访问控制的开放性介于（　）。

A．私有和受保护 B．受保护和公有

C．私有和公有 D．以上都不是

9．下列关于类、包和源文件的描述中，不正确的是（　）。

A．一个包可以包含多个类

B．一个源文件中，可能有一个公共类

C．属于同一个包的类在默认情况下可以相互访问

D．系统不会为源文件创建默认包

10．下列关于类、包和源文件的说法中，错误的是（　）。

A．一个文件可以属于一个包 B．一个包可包含多个文件

C．一个类可以属于一个包 D．一个包只能含有一个类

11．为了使包 ch4 在当前程序中可见，可以使用的语句是（　）。

A．import ch4.*; B．package ch4.*;

C．ch4 import; D．ch4 package;

12．在使用 interface 声明一个接口时，只可以使用哪一个修饰符修饰该接口（　）。

A．private B．protected C．private protected D．public

13．（　）接口的定义是正确的。

A．interface B
{ void print() { } ;}

B．abstract interface B
{ void print() ;}

C．abstract interface B extends A1,A2 //A1、A2 为已定义的接口
{ abstract void print(){ };}

D．interface B
{ void print();}

14．关于接口的描述，以下（　）说法是正确的。

A．给出公有的静态常量数据成员　　B．不能定义其他数据成员
C．只能定义公有的抽象方法　　D．不能定义其他形式的成员方法

二、填空题

1．接口中定义的变量只能是_____。
2．接口中没有_____方法，所有的成员方法都是_____方法。
3．将当前文件夹下的所有文件打包为 myjar.jar 的命令是_____。
4．在 Java 程序中，通过类的定义只能实现单重继承，但通过_____的定义可以实现多重继承关系。
5．在 Java 程序中定义接口所使用的关键字是_____。
6．一个 Java 程序可以定义_____个程序包，一个包包含多个文件或类。
7．在运行时，由 java 解释器自动引入，而不用 import 语句引入的包是_____。
8．_____包是 java 的核心类库，包含了运行 java 程序必不可少的系统类。
9．JAR 文件就是_____的简称。
10．定义一个类 classA，使用一个接口 InterA，且父类为 FatherA 的声明语句为_____。

三、判断题

1．一个类如果实现了某个接口，那么它必须重载该接口中的所有方法。　（　）
2．因为 Java 不支持多重继承，所以定义类时 implements关键字后面只能说明一个接口名。（　）
3．接口是特殊的类，所以接口也可以继承，子接口将继承父接口的所有常量和抽象方法。（　）
4．接口中的所有方法都没有被实现。　（　）
5．实现一个接口，则在类中一定要实现接口中的所有方法。　（　）
6．接口可以声明成 final。　（　）
7．接口的所有方法默认都是 public、abstract 和 non-static 的。　（　）
8．一个子类不可以有多个父类，但是可以实现任意数量的接口。　（　）
9．类头定义主要说明类的名字、父类名及接口名。　（　）
10．没有定义访问控制符的类属性和类方法可以被同一个包中的其他类和对象访问。（　）
11．类 A 和类 B 位于同一个包中，则除了私有成员，类 A 可以访问类 B 的所有其他成员。（　）

四、编程题

1．找出下列程序片段中的错误，用横线标注出来，并将错误改正过来（或说明错误原因）。

```
interface   computable
{
   final int max=100;
   void speak(String s);
   int f(int x);
}
class aaa implements computable
{
public int f(int x)
```

```
    { int sum=0;
      for(int i=1;i<=x;i++)
        { sum=sum+i;
        }
      }
}
```

2．阅读程序，写出程序的运行结果。

```
interface Animal
{
   void eat();
   void sleep();
}
class Zoo
{
private class Tiger implements Animal
   {   public void eat()
       {       System.out.println("tiger eat");
       }
       public void sleep()
       {       System.out.println("tiger sleep");
       }
   }
   Animal getAnimal()
   {       return new Tiger();
   }
}
class Test
{
   public static void main(String[] args)
   {     Zoo z=new Zoo();
         Animal an=z.getAnimal();
         an.eat();
         an.sleep();
   }
}
```

3．创建一个接口 Shape，其中有抽象方法 area，类 Circle、Rectangle 实现 area 方法计算其面积并返回。又有 Star 类实现 Shape 的 area 方法，其返回值是 0，Star 类另有一返回值 boolean 型方法 isStar；在 main 方法里创建一个数组，根据随机数向其中加入 Shape 的不同子类对象。然后将数组中元素依次取出，判断其是否为 Star 类，如是返回其个数，否则返回其面积。

4．定义一个接口 Sortable，包括一个抽象方法 int Compare(Sortable s)，表示需要进行比较大小，如果返回值大于 0 则表示大于；定义一个类 Student，要求实现此接口，必须重写接口中的抽象方法。Student 类中包括成绩 score 属性，重写 public String toString()方法，在比较大小时按照成绩的高低比较；定义一个类 Rectangle，要求实现此接口，必须重写接口中的抽象方法。Rectange 类中包括 length、width 属性，同时包括相应的构造方法，int area()，重写 public String toString()方法，在比较大小时按照面积的大小进行比较； 定义一个 Sort 类，其中定义方法 public static void SelectSort(Sortable [] a)按照选择方法进行降序或升序排序；定义一个 TestSort 类，测试以上定义的类。

5．定义一个接口 IPerson，封装一个方法：void print()，输出人员有关信息，利用 IPerson 接口规范，定义一个类 Teacher，表示某学校教师的最基本信息，包括 4 个成员变量：工号 num、姓名 name、工龄 workAge、职务 job，3 个成员方法：Teacher (String num,String name,int workAge,String job)、Teacher(String num,double workAge) 输出教师的工号和工龄 print()；再定义一个类 Director，表示教师中的主任，主任有专门的助理，包括 5 个成员变量：工号 num、

姓名 name、工龄 workAge、职务 job、助理名 assistantName，2 个成员方法：Director(String num,String name,int workAge,String job,String assistantName) 输出主任的工号和他的助理 print()；定义一个主类 Test，该类中利用 Teacher 类和 Director 类，输出每个教师的工号和工龄以及主任的助理名。

五、简答题

1．什么是接口？为什么要引入接口这个概念？
2．接口如何实现的，实现接口应注意什么？
3．接口的继承与类的继承有哪些异同？
4．简述接口和抽象类的区别。
5．什么是包？它的作用是什么？如何创建包？
6．包是如何使用的，给出几个系统常用的包。
7．什么是 jar 文件，简述其命令的用法。