第七章：包、接口

教学内容：

1. 包
2. 接口

一、 包

- Java 通过引入包的机制，来对这些类进行分门别类的管理。功能上有点类似于文件系统中的目录。
- 包的作用
  - 合理组织程序文件
  - 提供名字空间
  - 实现更多层面的访问控制

如在 SDK 中，大多数被分组进名为 java 的核心包中，我们也可以创建自己的包和包含进此包的相关的类。同一包中类名是唯一的(可以理解为在同一个文件夹中不可以有同名文件)。

创建包的语法：

```
package com.iweb.test23;

public class Test {

    public static void main(String[] args) {
        System.out.println("hello world");
    }

}
```

在一个 java 的源文件中只能有一句 package 语句，并且通常位于文件的第一行。

引入包的语法：

```
import java.util.Scanner;
import java.util.*;
```

两种写法都可以。java 机制会自动引入 java.lang 包，其它的需要使用 import 引入。

```
package com.iweb.test23;//创建包
import java.util.Date;//引入包
public class Test {

    public static void main(String[] args) {
        Date date = new Date();
        System.out.println("现在的时间:" + date);
    }

}
```

Java 常用类库：

包名	主要功能
java.applet	提供了创建applet需要的所有类
java.awt.*	提供了创建用户界面以及绘制和管理图形、图像的类
java.beans.*	提供了开发Java Beans需要的所有类
java.io	提供了通过数据流、对象序列以及文件系统实现的系统输入、输出
java.lang.*	Java编程语言的基本类库
java.math.*	提供了简明的整数算术以及十进制算术的基本函数
java.rmi	提供了与远程方法调用相关的所有类
java.net	提供了用于实现网络通讯应用的所有类
java.security.*	提供了设计网络安全方案需要的一些类
java.sql	提供了访问和处理来自于Java标准数据源数据的类
java.test	包括以一种独立于自然语言的方式处理文本、日期、数字和消息的类和接口
java.util.*	包括集合类、时间处理模式、日期时间工具等各类常用工具包
javax.accessibility	定义了用户界面组件之间相互访问的一种机制
javax.naming.*	为命名服务提供了一系列类和接口
javax.swing.*	提供了一系列轻量级的用户界面组件，是目前Java用户界面常用的包

二、 接口

接口的概念：

- 1、Java 中不允许类的多继承，但在解决实际问题过程中，仅仅依靠单一继承在很多情况下都不能将问题的复杂性表述完整。通过接口可以实现多继承。
- 如：Father 类有 playFootball()，Mother 类有 sing()，如果采用类的继承来产生一个 Son 类，则它只能从一个类中继承。要么继承 Father，要么继承 Mother。
- 2、接口在面向对象的设计与编程中应用非常广泛，特别是实现软件模块间的连接方面有着巨大的优势。

接口的声明和注意事项：

- 如果一个抽象类中所有的方法都是抽象的，就可以将这个类用另外一种方式来定义，也就是接口定义。
- [public] interface 接口名 [extends 父接口名列表]
- {
- 数据类型 常量名=常数;
- 返回值 方法名([参数列表]);
- }
- **几点说明：**
- **1、interface 是接口的关键字，定义接口和定义类相似。并被编译为 class 文件。**

- **2、接口的访问控制符只有 public，如果使用 public 修饰符，则可以被所有类和接口使用，且接口名与文件名相同。如果不使用 public，则接口只能被同一个包中的类和接口使用。**
- **3、接口中所有的方法都是 public abstract 即公共的抽象方法。**
- **4、接口中可以有数据成员，这些数据成员默认都是 public static final 即公共类常量。**

接口声明的示例：

```
public interface Father {

     void playFootball();

}


public interface Mother {

     void sing();
}


public class Son implements Father, Mother {

     @Override
     public void sing() {
          System.out.println("Son is sing");
     }
```

```
    @Override
    public void playFootball() {
        System.out.println("Son is playFootball");
    }

}
```

- 接口的实现通过类来完成，在定义类时使用“implements 接口名列表”短语，并在类体中实现接口中的抽象方法。接口和实现类之间的关系实质上是继承的关系 ；
- 一个类可以实现多个接口，从而实现多继承。
- **注意：**
  - **在类声明部分，用 implements 关键字指明该类将要实现哪些接口。**
  - **实现接口的类必须在类体中给出所有方法的实现，否则该类应该声明为抽象类。**
  - **接口中的方法都是 public 的，所以实现方法时也必须加上 public，否则编译报错。**
- **总结：**
  - **一个接口可以继承另一个接口**
  - **Java 中不允许类的多继承，但允许接口的多继承**
  - **在 java 中一个类可以实现多个接口**

- **一个类在继承另一个类的同时，可以实现多个接口。**

接口的示例：

```
public interface TestInterface {

    public final static int COUNT=100;//public final static可以省略

    public abstract int getNumber();//public abstract可以省略

}
```

接口和抽象类的区别：

1、抽象类和接口都不能直接实例化，如果要实例化，只能通过子类进行实例化。

2、抽象类要被子类继承，接口要被类实现。

3、接口所有的方法都是抽象方法，抽象类中可以可以有抽象方法也可以有实例方法。

4、接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。

5、抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。

6、抽象方法只能声明，不能实现，接口是设计的结果 ，抽象类是重构的结果

7、抽象类里可以没有抽象方法

8、抽象方法要被实现，所以不能是静态的，也不能是私有的。

9、接口可继承接口，并可多实现接口，但抽象类只能单根继承。