第6章访问控制权限

导入

- 问题
  - 程序应该更具可维护性
  - 变动的事物与不变的事物要区分开
  - 接口要公开:不该碰的不要碰
  - 实现要隐藏:可以更改内部工作方式
- 面向对象三大特征之一:封装
  - 包
  - 权限修饰符

6.1包:库单元

- 定义包
  - package cn.nku.liujiaXin
  - 文件除注释外的第一行
- 导入包 import java.util.*;
- 默认包
- 编译单元
  - Java源文件,后缀名必须是.java
  - 最多只能有一个public类,且必须和文件名相同,包括大小写
  - 编译后,每个类生成一个.class文件

创建独一无二的包名

- 域名的倒叙形式
  - cn.nku.liujiaXin
- 将包映射到文件夹
  - d:/java/demo/cn/nku/liujiaXin
- 如何找到类
  - CLASSPATH,包所在的位置
  - CLASSPATH=.;d:\java\lib.jar;d:\java

类名冲突

- 两个不同类库中包含相同名字的类
  - java.util.Date
  - java.sql.Date
- 导入包
  - import java.util.*;
  - import java.sql.*;
  - Date d = new Date();//冲突
  - java.util.Date d = new java.util.Date();//正确
- 练习2

静态导入

- 动态导入:包-类
- 静态导入:类-方法
- 能够更加方便的使用类中的静态方法

```
package test1;

public class Range {
    public static void print(int n) {
        System.out.print(n);
    }
}
```

```
package test2;
import static test1.Range.*;
public class Test{
    public static void main(String[] args) {
        print(5);
    }
}
```

6.2访问权限修饰符

- private , protected , public
- 包访问权限,friendly
- 修饰类
  - friendly和public
- 字段,方法
  - 全部都可以

	private	friendly	protected	public
同一类	√	√	√	√
同一文件		√	√	√
同一文件夹		√	√	√
其它文件夹(子类)			√	√
其它类				√

练习 4,5,6

单例模式

- 练习8,9

```
public class Soup{
    private Soup(){}
    private static Soup instance=null;
    public static Soup getInstance(){
        if(instance==null){
            instance = new Soup();
        }
        return instance;
    }
}
```

课堂练习

练习1：(1) 在某个包中创建一个类，在这个类所处的包的外部创建该类的一个实例。

练习2：(1) 将本节中的代码片段改写为完整的程序，并校验实际所发生的冲突。

练习4：(2) 展示**protected**方法具有包访问权限，但不是**public**。

练习5：(2) 创建一个带有**public**, **private**, **protected**和包访问权限域以及方法成员的类。创建该类的一个对象，看看在你试图调用所有类成员时，会得到什么类型的编译信息。请注意，处于同一个目录中的所有类都是默认包的一部分。

练习6：(1) 创建一个带有**protected**数据的类。运用在第一个类中处理**protected**数据的方法在相同的文件中创建第二个类。

练习8：(4) 效仿示例**Lunch.java**的形式，创建一个名为**ConnectionManager**的类，该类管理一个元素为**Connection**对象的固定数组。客户端程序员不能直接创建**Connection**对象，而只能通过**ConnectionManager**中的某个**static**方法来获取它们。当**ConnectionManager**之中不再有对象时，它会返回**null**引用。在**main()**之中检测这些类。

练习9：(2) 在**access/local**目录下编写以下文件（假定**access/local**目录在你的CLASSPATH中）：

```
// access/local/PackagedClass.java
```

然后在**accesss/local**之外的另一个目录中创建下列文件：

```
// access/foreign/Foreign.java
```

解释一下为什么编译器会产生错误。如果将**Foreign**类置于**access.local**包之中的话，会有所改变吗？

作业

- 提交练习8

提问