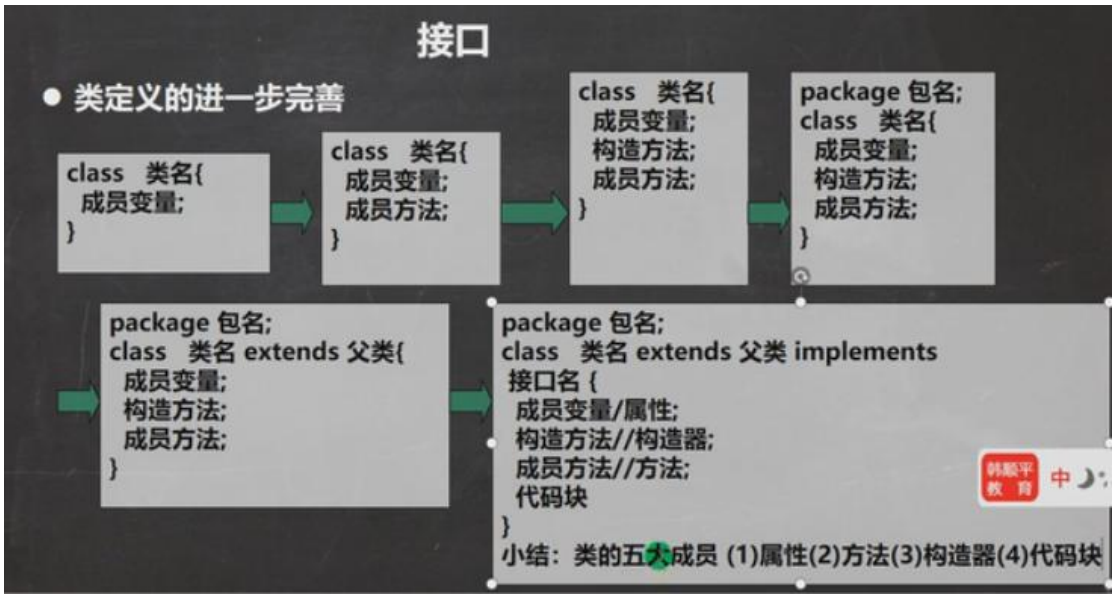

内部类：
一个类的内部类又完整嵌套了另一个类结构，被嵌套的类称为内部类（inner class），嵌套其他类的类称为外部类（outer class），是我们类的第五大成员（分别有：属性，方法，构造器和代码块）。内部类的最大特点就是可以直接访问私有属性，并且可以体现类与类之间的包含关系。

基本语法：

```
class Outer{ //外部类
        class Inner{    //内部类

          }
}
class Other{ //外部其他类
}
```

内部类的分类：
定义在外部类局部位置上（比如方法内）：
1）局部内部类（有类名）（其本质依然是一个类）
局部内部类是定义在外部类的局部位置，比如方法中，并且有类名
1. 可以直接访问外部类的所有成员，包含私有的
2. 不能添加访问修饰符，因为它的地位就是一个局部变量。局部变量是不能使用修饰符的，但是可以使用 final 修饰，因为局部变量也可以使用 final
3. 作用域：仅仅定义它的方法或代码块中
4. 局部内部类---访问--->外部类的成员[访问方式：直接访问]
5. 外部类---访问---->局部内部类的成员[访问方式：创建对象，再访问(注意:必须在作用域内)]
6. 外部其他类---不能访问局部内部类（因为局部内部类的地位是一个局部变量）
7. 如果外部类和局部内部类的成员重名时，默认遵循就近原则，如果想访问外

部类的成员，则可以使用（外部类名.this.成员）去访问，要访问外部类的成员，比如：System.out.print(“外部类成员” + 外部类名.this.成员)
2）匿名内部类（没有类名，重点）
1. 匿名内部类的局部位置，比如方法中，并且没有类名，匿名内部类使用一次就不能再使用了
匿名内部类的基本语法：

```
new 类或接口(参数列表){
        类体
}
```

2. 匿名内部类的语法比较奇特，因为匿名内部类既是一个类的定义，同时它本身也是一个对象，因此从语法上看，它既有定义类的特征，也有创建对象的特征
3. 可以访问外部类的所有成员，包含私有的
4. 不能添加访问修饰符，因为它的地位就是一个局部变量
5. 作用域：仅仅在定义它的方法或代码块中
6. 匿名内部类---访问---->外部类成员[访问方式：直接访问]
7. 外部其他类----不能访问----->匿名内部类（因为匿名内部类地位是一个局部变量）
8. 如果外部类和内部类的成员重名时，内部类访问的话，默认遵循就近原则，如果想访问外部类的成员，则可以使用（外部类名.this.成员）去访问

定义在外部类的成员位置上：
1）成员内部类（没有 static 修饰）
2）静态内部类（使用 static 修饰）