内部类

概念：定义在类的内部的类

分类：

成员内部类：定义在类的内部，方法外；可以看做外部类的一个成员

方法内部类（局部）：定义在方法中

静态内部类：static修饰的内部类

匿名内部类：没有名字的内部类，只有一对大括号  
为什么要有前三个内部类的产生？

因为即使有属性和方法但是不能完全的描述出来类的属性。

成员内部类

定义：

调用格式：直接初始化编译会报错

外部类的类名.内部类的名字 对象名=外部类的对象.内部类的对象();

Outer out=new Outer();

Outer.Inner in=out.new Inner();

或者Outer.Inner in=new Outer().new Inner();

对象名.属性/方法

注意：

1. 内部类可以直接使用内部类的成员变量和成员方法，用private修饰的也可以直接使用
2. 外部类如果想访问内部类的成员变量和成员方法，则需要创建内部类的对象来调用。方法或者变量通过private修饰，只能外部类访问，其他类不能访问，提高了安全性。
3. 内部类一般会定义为private的不提供给外部访问，提高安全性。

静态内部类

static修饰的内部类

1. 静态内部类中可以定义静态方法

调用格式：

1. 在外部类中可以通过内部类名.静态方法
2. 外部类.静态内部类.静态方法();
3. 先创建内部类的对象，通过对象名.来调用方法();--不推荐使用
4. 静态内部类中是否可以定义非静态方法？--可以

调用格式：

1. 在外部类中使用可以先创建内部类的对象，通过对象名.调用方法
2. 在其他类中调用需要先创建内部类的对象

外部类.内部类 对象名=new 外部类.内部类();

Outer1.Inner1=new Outer1.Inner1();

1. 静态内部类只能访问外部类的静态属性和静态方法

方法内部类：

1. 方法内部类可以直接访问外部类的属性以及方法的局部变量
2. 内部类和外部类没有继承关系，不可以使用super关键字
3. 只在当前方法内部通过创建类的对象使用，需要先让类加载再创建对象。然后调用类内的方法。

匿名内部类：

定义格式： new 类名/接口名(){

重写方法；

}; //必须要有分号，代表的是一句代码

new Interf() {

@Override

public void method() {

}

}

匿名内部类本质是某个类的子类对象或者接口的实现类的对象

主要功能：实现接口或者继承某个类；主要目的：重写抽象方法

方法参数列表是接口类型/抽象类类型只能传入实现类对象/子类对象，只使用一次时使用匿名内部类

Lambda 表达式：

作用：实现函数式接口。重写抽象方法

从Jdk1.8之后开始，用于函数式接口重写抽象方法

函数式接口---只有一个抽象方法--@FunctionalInterface*//注解，给计算机看的注释，*

*//Lambda表达式，实现一个接口，重写里边的抽象方法  
// (参数列表)->{抽象方法进行重写的方法体}  
// Calc c=(int x,int y)->{return x+y;};  
// 从前往后推导出参数列表  
// Calc c=(x,y)->{return x+y;};  
// 当重写的方法的方法体只有一句时可以省略{}和return  
 /\* Calc c = (x, y) -> x + y;  
 System.out.println(c.sum(1, 5));\*/  
// 数组 排序* **int**[] arr={3,6,5,1,9};  
 *//注意：参数名可以随便给，但是不能和前边的重复  
 /\* ArraySort a=(int[] arr1)->{  
 Arrays.sort(arr1);  
 };\*/  
// 省略参数类型和{}，  
// 如果只有一个参数小括号也可以省略  
// ArraySort a=(arr1)-> Arrays.sort(arr1);  
// ArraySort a=arr1-> Arrays.sort(arr1);  
// 如果方法体里只有一个静态方法而且只有一个参数时可以如下简化  
// ::代表传递静态方法，参数从头到尾只操作一个* ArraySort a=Arrays::*sort*;