1. 使用内部类的原因

内部类定义在另一个类中的类，那么为什么需要使用内部类呢？

1. 内部类可以对同一个包中的其他类隐藏
2. 内部类方法可以访问定义这个类的作用域中的数据，包括私有数据

二． 内部类访问外围类对象

一个内部类方法可以访问自身的数据字段，也可以访问创建它的外围类对象的数据字段。因为内部类的对象总有一个隐式引用，指向创建它的外部类对象，外围类的引用在内部类的构造器中设置，编译器会修改所有的内部类构造器，添加一个对应外围类引用的参数。如果内部类中不设置构造器，则会自动生成一个默认的无参构造器。

我们可以将内部类声明为private，这样，只有外围类方法能够构造内部类对象。只有内部类可以是private，其他的类只能是public或者protected。

1. 内部类的特殊语法

在内部类中使用外部类的引用时，正规语法为OuterClass.this，表示外部类引用。如TalkingClock为外部类，beep为外部类中的参数，那么内部类调用时使用的就是TalkingClock.this.beep

可以使用这种语法形式来调用内部类对象的构造器，如下所示，这里的this指的就是外部类：

ActionListener listener = this.new TimePrinter();

如果内部类是一个公共的内部类，那么任何一个外部类对象都可以创建该内部类，如下所示：

var jabber = new TalkingClock(1000, true);

TalkingClock.TimePrinter listener = jabber.new TimePrinter();

因此，内部类中的static变量必须是final的，不然被多个外部类对象调用时不安全。

1. 内部类原理

内部类是一个编译器现象，与虚拟机无关，编译器会把内部类转换为常规的类文件（用$美元符号）分割外部类名和内部类名。如，TalkingClocl$TimePrinter。在编译器生成的内部类中，通过反射可以查询到生成了一个额外的实例字段this$0，对应外围类的引用。

在外部类中，编译器也添加了一个静态方法access$0

public Boolean access$0(TalkingClock);

它将返回作为参数传递的哪个对象的beep字段，内部类方法在需要beep字段时，将会调用这个access$0方法，如下所示：

if(TalkingClock.access$0(Talking.this))

这样做，是很有风险的，因为本来beep字段是private的，但是通过内部类可以访问到这个beep字段。而编译器为TalkingClock类中添加了access$0方法，那么与TalkingClock类同在一个包中的类就可以通过调用这个类来获取beep字段。这种方法虽然只能通过虚拟机指令调用，但是也是由风险的。

1. 局部内部类

如果内部类对象只使用一次，则在一个方法中局部地定义内部类，这种局部内部类的优势是完全隐藏自己，作用于只限于声明这个局部类的代码块中，声明局部类时不能有访问说明符。如下所示：

public void start(){

class TimePrinter implements ActionListener

{

}

}

同时，局部内部类还可以访问局部变量，这些局部变量必须是final的。

1. 匿名内部类

在使用局部内部类时，如果指向创建这个类的一个对象，且不需要类的名称，那么这个类称为匿名内部类。语法如下所示：

new SuperType(construction parameters){

inner class methods and data

}

SuperType可以是接口或者类。若是接口，则内部类需要实现改接口；若是类，则内部类需要扩展这个类。实现接口时，不能有任何构造参数

new InterfactType(){

}

在扩展类时，可以有参数

var count = new Person(“Dree”){}

在构造参数列表后面有大括号，就是在定义匿名内部类。由于匿名内部类没有类名，那么就没有构造器，但是可以提供一个对象初始化块：

var count = new Person(“Dree”){{ }}

1. 双括号初始化

在初始化数组列表时，我们可能都用过这样一种语法：

invite(new ArrayList<String>(){{ add(“ddd”); add{“ddsf”};}});

他使用了两层大括号，这里可以看出使用了匿名内部类，外层括号表示建立了一个匿名内部类，内层的括号则是一个对象初始化块。

1. 获取静态方法的类名

通常，在记录日志时需要获取当前类的类名，使用的一般是this.getClass()。然而静态方法是没有this的，因为静态方法不属于任何一个对象，因此this对它是无效的，应该使用如下表达式：

new Object(){}.getClass().getEnclosing();

new Object(){} 建立Object的匿名子类的一个匿名对象；

getEnclosingClass得到它的外围类，就是包含这个静态方法的类。

1. 静态内部类

当我们只需要将内部类隐藏在一个类中，而不需要调用外部类中的对象时，可以将内部类声明为static，这样就不会有外部类的引用了。静态内部类中可以有静态字段和静态方法。