1. 内部类为非static：

当某个外围类对象创建了一个内部类对象是，此内部类对象必定会秘密地捕获一个指向那个外围类对象的引用。所以它能访问器外围对象的所有成员，而不需要任何特殊条件。

在拥有外部类对象之前是不可能创建内部类对象的，这是因为内部类对象那个会暗暗的连接到创建它的外部类对象上。

2. 内部类是private类型时

权限其实是相当于1个变量，外部类是无法访问这个内部类的。当内部类向上转型为接口类型时，就有了用武之地。

private内部类个累的设计者提供了一种途径，通过这种方式可以完全阻止任何依赖于类型的编码，并且完全隐藏了实现细节，此外，从给客户端程序员的角度来看，由于不能访问任何新增加的、原本不属于公共的方法。

3. 匿名内部类

相当于是创建了一个继承自某个接口的匿名类对象，通过new表达式返回的引用被自动向上转型为对接口的引用。

匿名内部类、或局部内部类如果想调用在其外部定义的对象，那么必须被声明为final；

匿名内部类可使用实例初始化（也是构造代码块）的方式模拟构造方法，如下图：

```
public Inner getInner(final int x){
    return new Inner(){
        {
            System.out.println("实例初始化！ "+x);
        }
        public void use(){
            System.out.println("调用use方法！ ");
        }
    };
}
```

4. 静态内部类/嵌套内部类

静态内部类不会与外围类对象之间有关系。也就是没有保存外围类对象的引用，相当于两个独立的类。内部类的创建不会依赖于外围类具体的对象。
静态内部类可包含static数据、static字段。

它也只能访问外围类的静态成员。

5. 接口内部也可写内部类，自动为public、static的。

6. 内部类的继承必须明确声明只想明确指向外围类对象的引用。可参考《java编程思想》212页。

7. 内部类没有覆盖这种说法。除非显示的声明。否则内部类只是存在于

6. 一个内部类不管被嵌套多少层，都能透明的访问所有它所嵌入的外围类的所有成员变量。