Java中内部类与匿名内部类

1. **内部类**
2. 内部类：将一个类定义在另外一个类的里面，里面的类的就称为内部类又称为内置类，嵌套类。

**class** Outer{

**class** Inter{

}} 编译后的class文件是：Outer.class 和 Outer$Inter.class。

1. 为什么把类定义在内部，什么时候需要定义内部类？

答：类定义在另外一个类内部，该内部类就可以直接访问外类中的成员，特别是private的成员。内部类其实也是一种封装。

当描述一个事物A时，发现该事物中还有其他事物（B，C等），而且这些其他事物（B，C等）还要访问该事物A的内容，这时就需要把这些其他事物（B,C等）定义为内部类来描述。例如：人体是一个类，其中包括很多器官和组织，而大脑又要访问人体的其他器官，可以把大脑单独定义成一个类，从而成为内部类。

1. 内部类的访问特点：
2. 内部类可以直接访问此外部类中的成员（private的可以直接被访问）；
3. 此外部类要访问内部类的内容，需要建立内部类的对象。
4. 内部类也可以放在类中与成员并列的位置，也可以放在局部位置上（成员方法中）（局部内部类）；如果内部类放在局部位置（成员方法中）中，内部类内部需要访问此方法中的局部变量，此局部变量必须定义为final的，即内部类在局部位置上时，内部类只能访问局部中**被final修饰的局部变量即常量**。**原因：**因为内部类定义在成员方法中，当执行方法时，会产生一个对象，但是方法执行完后就会释放，而这个对象再次访问这个方法的局部变量，肯定找不到了，之所以定义为final，是在定义个常量，即只是给常量起个名字，所以对象再次寻找时，找的就是一个常量而已，而常量并不是随着方法的消失而消失。

实例：**class** Outer {

**void** method() {

**final** **int** num = 10 ;

**class** Inter{

**void** show1() {

System.***out***.println(num);

}

}

Inter inter = **new** Inter();

inter.show1();

}

}

**public** **class** OuterInter {

**public** **static** **void** main(String[] args) {

**new** Outer().method();

}

}

1. 如果在外类的外部直接生成一个内部类(前提：不是private的)的对象方法是

**Outer.Inter inter = new Outer().new Inter();**

**inter.xxxx(); //就可以直接利用内部类的引用访问内部类的成员**

1. **如果内部类是静态的，则就差不多相当于一个外部类。**

**生成对象的方法：**

**Outer.Inter inter = new Outer.Inter(); //不需要先new一个外部类对象了**

1. **如果内部类是静态的，而且内部类的成员也是静态的：**

**直接就可以调用这些静态方法： Outer.Inter.xxx();**

**注意：如果内部类中定义了静态方法，则内部类必须定义为静态的。**

1. 为什么内部类能够直接访问外部类中的成员？

答：因为内部类拥有了外部类的引用，使用方法 **外部类名.this.外部类成员**。

实例：**class** Outer{

**int** num = 10;

**class** Inter{

**int** num = 9;

**void** show() {

**int** num = 8;

System.***out***.println(num);//调用的是此方法的局部变量

System.***out***.println(**this**.num);//this代表的是内部类的对象，

System.***out***.println(Inter.**this**.num);//同this一样，this本身就是内部类的对象，利用Inter在此修饰没有必要

System.***out***.println(Outer.**this**.num);//利用外部类修饰this，则this就代表的是外部类的对象

}

}

}

**public** **class** OuterInter {

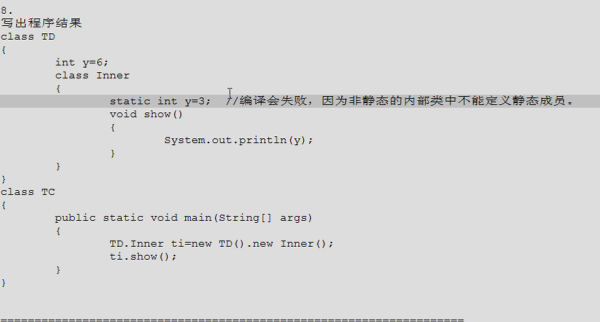
**public** **static** **void** main(String[] args) {

Outer.Inter inter = **new** Outer().**new** Inter();

inter.show();

}

}结果：输出为 8,9,9,10



如果类中有静态成员，则这个类必须是静态的。

1. **匿名对象**
2. **匿名对象（没有名字的对象）：new Car(); 这就是个匿名对象。如果需要调用某个成员方法，直接利用 new Car().run();**
3. **当对象对成员方法仅仅一次的调用时，可以简化不给此对象起名字，使用匿名对象。因为匿名对象一旦使用一次后，无法再次找到它了。**匿名对象对属性操作没有什么意义，因为如果赋值一次后，没法调用出来。
4. **匿名对象可以作为实际参数进行传递，此时匿名对象对象就会传进去，被此参数所在方法所接收了。如show（Car c）{ c.red = ….; c.run(); ………}；是一个方法，而调用语句show (new Car()); 此匿名对象就会被方法中的c所接收了。**
5. **匿名内部类**
6. 匿名对象就是没有名字的对象，也就是对象的简写；同样，匿名内部类就是没有名字的内部类，是内部类的简写。注意匿名修饰的是内部类。
7. **匿名内部类的使用必须有个前提：内部类必须继承或者实现一个外部类或者接口。**
8. **抽象类之所以不能new（创建对象，实例化），是因为其中有抽象方法（当然抽象类中也可以没有抽象方法，但是那种情况大多数是其中的方法体是空的），如果把其中的抽象方法都实现了，就可以new（实例化）了。**
9. **匿名内部类其实就是一个匿名子类对象，只不过把对父类（可能是抽象类）的继承以及对接口的实现和创建一个对象综合在一起了，简化了一下而已。其中对父类的继承和对接口的实现与一般情况下相同。**
10. **为什么是匿名的，是因为这种定义格式不可能是独立于类之外，因为此类没有名字，不可能独立成为一个单独的类，必须在另外一个类内部创建，匿名就是说创建的类没有名字，并且此类没有名字就必须立刻创建对象，存在的意义就是创建一个对象。**
11. **格式： new 父类名（）or 接口名（）{子类内容}**

**在子类内容中如同一般继承或实现一样，可以重写也可以新建新的成员。**

**这种产生的是一个对象，可以另外给这个对象指定一个引用，所以匿名修饰的是内部类，不是说匿名对象。下面这种情况是先创建匿名内部类，后又创建一个匿名对象。**

**父类名 or 接口名 引用名 = new 父类名（）or接口名（）{子类内容} ；//要有分号，虽然可以以分号或大括号两种方式结尾，但是此处大括号不能代表结尾，必须加上分号。**

**如果直接调用方法：new 父类名or 接口名{子类内容}.xX();**

**实例：interface** USB{

**void** open();

**void** close();

}

**public** **class** NiMingNeiBuLei {

**public** **static** **void** main(String[] args) {

USB usb = **new** USB() {

**public** **void** open() {

System.***out***.println("open");

}

**public** **void** close() {

System.***out***.println("close");

}

**public** **void** download() {

System.***out***.println("download");

}

};

usb.open();

usb.close();

//usb.download(); //注意：在这里利用了多态，已经 但是无法向下转型，所以无法通过多态调用匿名内部类的特有方法，使用匿名内部类的特有方法只有一个方法，就是直接在匿名内部类后面调用特有方法，new 父类名or 接口名{子类内容}.xX(); 这样是因为 new 父类名or 接口名{子类内容} 相当于是个子类的对象，所以可以直接调用自己的特有方法。

}

}

1. **匿名内部类：**
2. **使用场景之一：当方法的形参是接口类型时，而且接口中的方法不超过三个时（太多了会使可阅读性降低），可以利用匿名内部类作为实参进行传递。**
3. **D**
4. **D**
5. **面试题目：**

**public** **class** NiMing {

**static** **class** Inter{

}//一个内部类

**public** **static** **void** main(String[] args) {

**new** Inter();//考点：这里内部类是成员级别的，只有把内部类（所有成员级别的内容）定义为static，才可以在静态方法（这里是main静态方法）中new对象（调用），否则出错，因为在一般方法里相当于this.new Inter（）;在静态方法中不能出现this，故在静态方法中所访问的成员必须是静态的（即静态变量，静态方法，以及静态内部类）。

}

**public** **void** method () {

**new** Test();//然而在非静态方法中不需要内部类是静态的，当然静态的和非静态的都是正确的

}

}