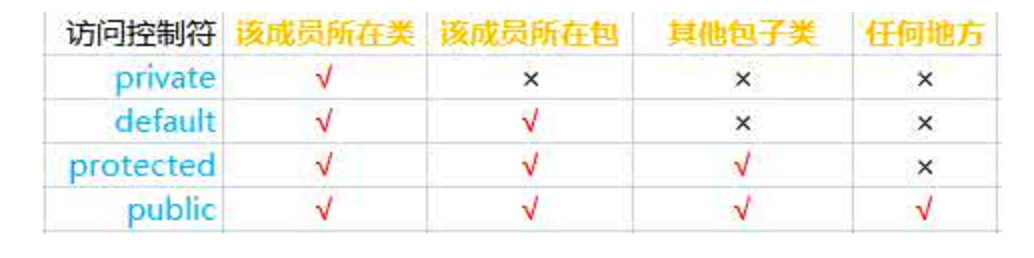
Day10.1—面向对象语言特性（续）

**——访问控制**

面向对象的基本思想之一是封装实现细节并且公开方法。Java语言采用访问控制修饰符来控制类及类的方法和变量的访问权限，从而只向使用者暴露方法，但隐藏实现细节。**访问控制分4种级别。**

成员变量、成员方法和构造方法可以处于4个访问级别中的一个；

顶层类只可以处于公开或默认访问级别；

**——内部类**

说起内部类这个词，想必很多同学都听说过，但是让你具体说它到底是个什么玩意儿你又说不上来…

**在Java中，可以将一个类定义在另一个类里面或者一个方法里面，这样的类称为内部类。**

内部类一般来说包括这四种：**成员内部类、局部内部类、匿名内部类和静态内部类。**

——成员内部类（**相当于类中的一个成员**）

成员内部类是最普通的内部类，它的定义为位于另一个类的内部，形如下面的形式:

**成员内部类可以无条件访问外部类的所有成员属性和成员方法（包括private成员和静态成员）。**

不过要注意的是，当成员内部类拥有和外部类同名的成员变量或者方法时，会发生隐藏现象，即默认情况下访问的是成员内部类的成员。如果要访问外部类的同名成员，需要以下面的形式进行访问：

**外部类.this.成员变量**

**外部类.this.成员方法**

虽然成员内部类可以无条件地访问外部类的成员，而外部类想访问成员内部类的成员却不是这么随心所欲了。**在外部类中如果要访问成员内部类的成员，必须先创建一个成员内部类的对象，再通过指向这个对象的引用来访问。**

成员内部类是依附外部类而存在的，也就是说，如果要创建成员内部类的对象，前提是必须存在一个外部类的对象。创建成员内部类对象的一般方式如下：

**Outer.Inner inner = new Outer().new Inner();**

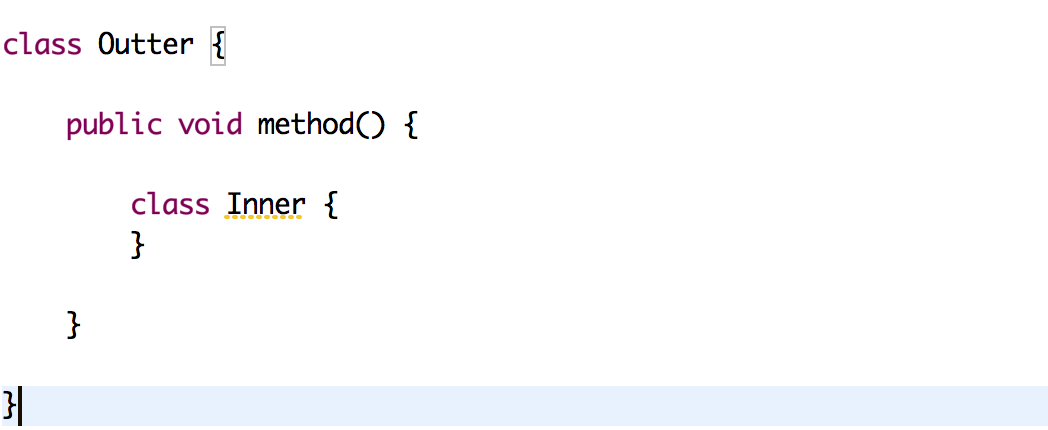
static一般用来修饰成员变量或函数也修饰代码块，一般不能修饰类，但是可以修饰内部类，被修饰的内部类可以直接作为一个普通类来用，不需要创建一个外部类的实例.

**——成员内部类中不能定义静态成员，而只能定义实例成员。**

**——外部类实例与内部类实例之间是一对多的关系，一个内部类实例只会引用一个外部类实例，而一个外部类实例对应零个或多个内部类实例。**

**内部类可以拥有private访问权限、protected访问权限、public访问权限及包访问权限。如果成员内部类Inner用private修饰，则只能在外部类的内部访问，如果用public修饰，则任何地方都能访问；如果用protected修饰，则只能在同一个包下或者继承外部类的情况下访问；如果是默认访问权限，则只能在同一个包下访问。这一点和外部类有一点不一样，外部类只能被public和包访问两种权限修饰。**

——局部内部类（相当于一个方法中的局部变量）

——局部内部类是定义在一个方法或者一个作用域里面的类

——局部内部类就像是方法里面的一个局部变量一样 ，是不能有public、protected、private以及static修饰符的。

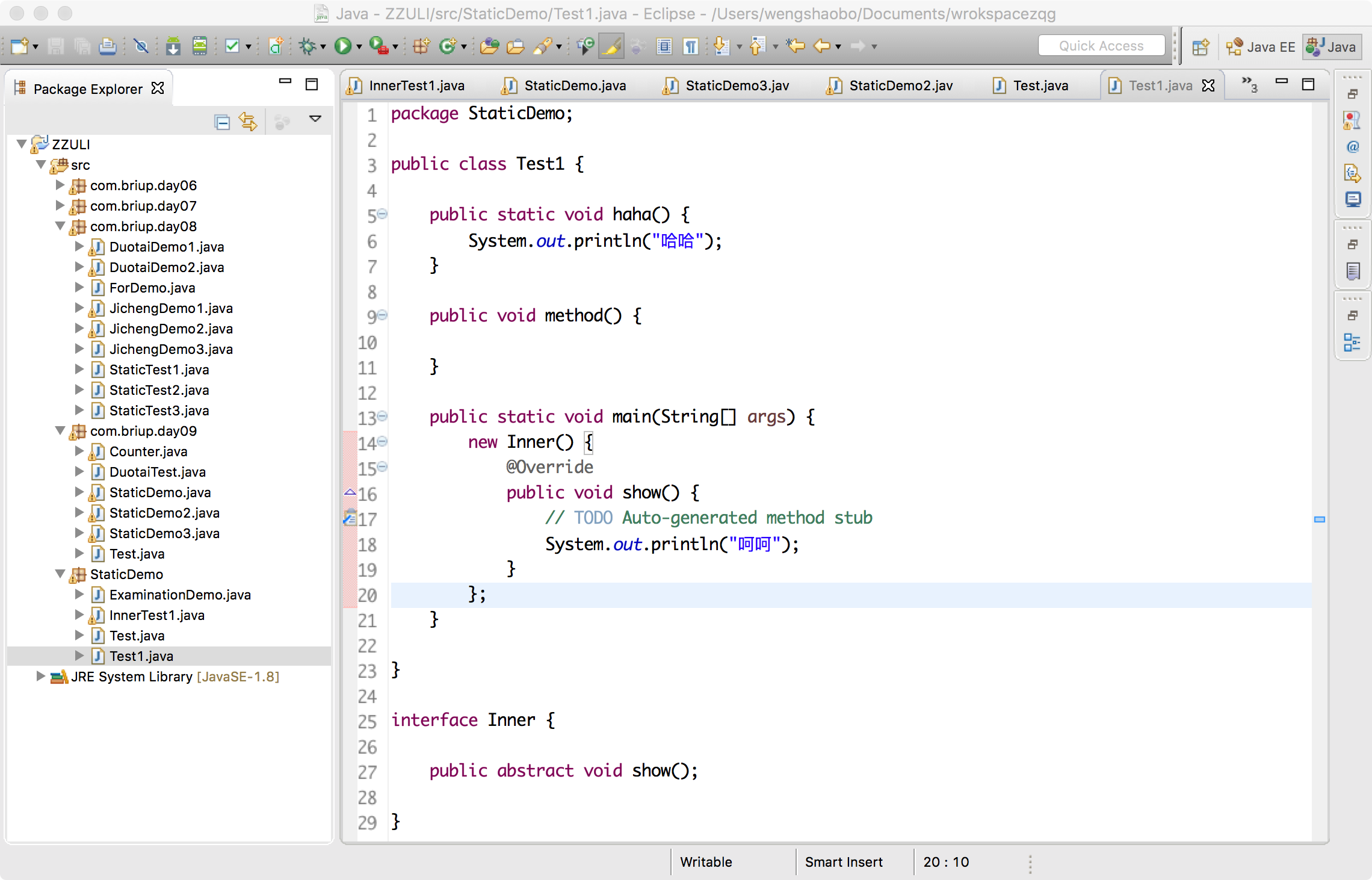
——局部内部类的访问仅限于方法内或者该作用域内。

——可以访问外层类的普通成员变量和静态成员变量以及普通方法和静态方法,也可以访问该内部类所在方法当中的局部变量,但是这个局部变量必须是final修饰。

——局部内部类和成员内部类一样，不能包含静态成员。

——匿名内部类（和局部内部类很相似）

匿名内部类也是用的最多的内部类。

**本质上其实是一个继承了类或者实现了接口的子类匿名对象。**

——可以写成成员变量的形式,也可以写在方法当中,一般写在方法当中较多；

——匿名内部类里可以访问外部类的普通属性和方法,以及静态属性和方法,如果要访问这个内部类所在方法中的局部变量,那么要求这个局部变量必须是final修饰的；

——匿名内部类里面没有构造函数,因为这个类没有名字,所以在其他地方不能用;

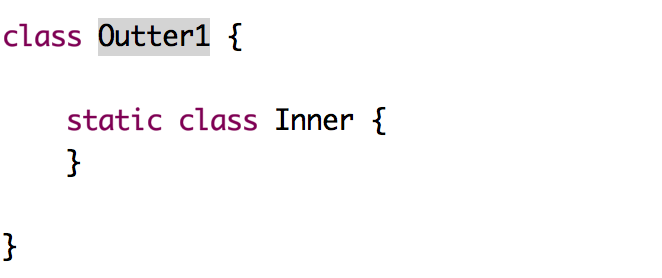
——匿名内部类也是不能有访问修饰符和static修饰符的。

Q：匿名内部类生成的.class文件是怎么起的名字?

匿名内部类是唯一一种没有构造器的类。正因为其没有构造器，所以匿名内部类的使用范围非常有限，大部分匿名内部类用于接口回调。匿名内部类在编译的时候由系统自动起名为Outter$1.class。一般来说，匿名内部类用于继承其他类或是实现接口，并不需要增加额外的方法，只是对继承方法的实现或是重写。

new Student().method();

——静态内部类

定义在另一个类里面的类，只不过在类的前面多了一个关键字static。

——静态内部类是不需要依赖于外部类的，这点和类的静态成员属性有点类似，并且静态内部类的对象不需要依赖于外部类的对象。

**内部类 变量名字 = new 内部类();**

——它不能使用外部类的非static成员变量或者方法，这点很好理解，因为在没有外部类的对象的情况下，可以创建静态内部类的对象，如果允许访问外部类的非static成员就会产生矛盾，因为外部类的非static成员必须依附于具体的对象。