定义：放在一个类内部的类就叫做内部类

作用：

1. 内部类可以很好的实现隐藏

   一般的非内部类，是不允许有private 与 protected 权限的，但内部类可以

2. 内部类拥有外围类的所有元素的访问权限

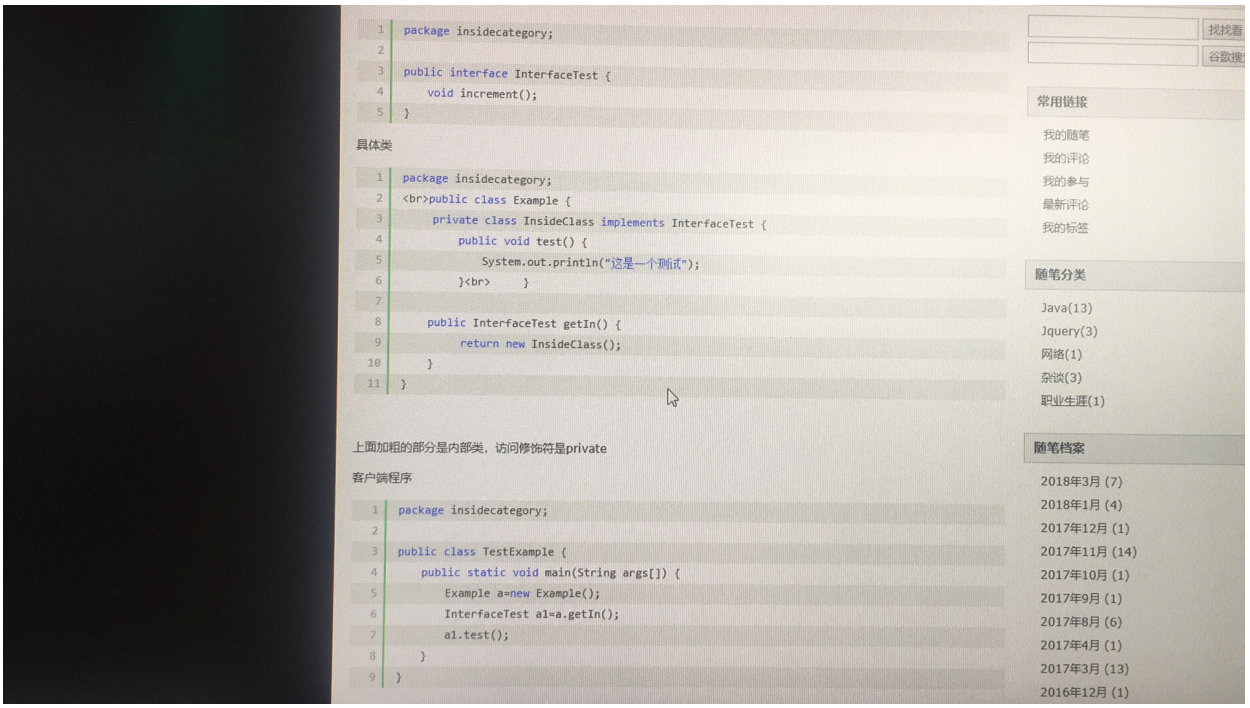
3. 可以实现多重继承

4. 可以避免修改接口而实现同一个类中两种同名方法的调用

例子：

1. 实现隐藏

   平时我们对类的访问权限，都是通过类前面的访问修饰符来限制的，一般的非内部类，是不允许有private与protected权限的，但内部类可以，所以我们能通过内部类来隐藏我们的信息。



```
package insidecategory;

public interface InterfaceTest {
    void increment();
}
```

具体类

```
package insidecategory;
<br>public class Example {
    private class InsideClass implements InterfaceTest {
        public void test() {
            System.out.println("这是一个测试");
        }<br>    }

    public InterfaceTest getIn() {
        return new InsideClass();
    }
}
```

上面加粗的部分是内部类，访问修饰符是private

客户端程序

```
package insidecategory;

public class TestExample {
    public static void main(String args[]) {
        Example a=new Example();
        InterfaceTest a1=a.getIn();
        a1.test();
    }
}
```


getIn()方法能返回一个InterfaceTest实例但是我们并不知道这个实例是怎么实现的，而且由于InsideClass是private的，所以我们如果不看代码的话根本看不到这个具体类的名字，所以说很好的实现了隐藏。

2. 可以无条件地访问外围类的所有元素

```java
package insidecategory;

public class TagBean {
    private String name="liutao";

    private class InTest {
        public InTest() {
            System.out.println(name);
        }
    }

    public void test() {<br>        new InTest();<br>    }

    public static void main(String args[]) {
        TagBean bb=new TagBean();
        bb.test();
    }
}
```

name这个变量是在TagBean里面定义的私有变量，这个变量在内部类中可以无条件地访问。

3.可以实现多重继承

这个特点非常重要，正是由于他的存在使得java继承机制更加完善。java只能继承一个类，它的多重继承在没有学内部类之前是用接口来实现的。但使用接口时候有很多不便的地方。比如我们实现一个接口就必须实现它里面的所有方法，而有了内部类就不一样了。它可以使我们的类继承多个具体类或抽象类。

```java
package insidecategory;

public class Example1 {
    public String name() {
        return "liutao";
    }
}
```

类二

```java
package insidecategory;

public class Example2 {
    public int age() {
        return 25;
    }
}
```

类三

```java
package insidecategory;

public class MainExample {
    private class test1 extends Example1 {
        public String name() {
            return super.name();
        }
     }

    private class test2 extends Example2 {
         public int age() {<br>            return super.age(
    }
```

```
13
14      public String name() {
15          return new test1().name();<br>   }
16
17      public int age() {
18          return new test2().age();<br>   }
19
20      public static void main(String args[]) {
21
22          MainExample mi=new MainExample();
23
24          System.out.println("姓名:"+mi.name());
25
26          System.out.println("年龄:"+mi.age());
27
28      }
29
30  }
```

类三中，里面分别实现了两个内部类test1和test2，test1类又继承了Example1，test2继承了Example2，这样我们的类三MainExample就拥有了Example1和Example2的方法和属性，也就间接地实现了多继承。

4.避免修改接口而实现同一个类中两种同名方法的调用

如果类要继承一个类，还要实现一个接口，当继承的类和接口里面有两个同名的方法怎么办，这时候就需要内部类了。

```
package insidecategory;

public interface Incrementable {

 void increment();<br><br>}
```

类 MyIncrement

```
package insidecategory;

public class MyIncrement {
    public void increment() {

        System.out.println("Other increment()");

    }

    static void f(MyIncrement f)  {

        f.increment();

    }
}
```

```java
package insidecategory;

public class Callee2 extends MyIncrement {

 private int i=0;

 private void incr() {

       i++;

       System.out.println(i);

 }

 private class Closure implements Incrementable {

      public void increment() {

        incr();

      }

 }

 Incrementable getCallbackReference() {

      return new Closure();
 }

}
```

我们可以用内部类来实现接口，这样就不会与外围类的方法冲突了。