## 基本内部类结构

class Outer{  
 private String msg = "hello,world";  
 class Inner{ //内部类,这里的this表示内部类的实例,外部类.this表示外部类的实例  
 private String info = "世界,你好";  
 public void print(){  
 System.*out*.print(msg); //内部类直接访问外部属性和方法(不受防控属性影响)  
 //System.out.print(Outer.this.msg); 正规写法  
 }  
 }  
 public void fun(){  
 Inner in = new Inner();

//Inner in = new Outer.Inner();也可以这么写  
 System.*out*.print(in.info); //外部类通过实例化内部类对象直接访问属性和方法(不受防控属性的影响)  
 }  
}

**注意点:**

1. **内部类里的this表示内部类的实例,内部类能访问到**

**外部类的实例:外部类.this**

**内部类能通过 外部类.this 能访问所有外部类的方法和属性(不受防控属性的影响),就一个类的this一样,可以访问这个类的所有方法和属性**

1. **外部类的方法里能直接实例化内部类对象,并且这个内部类对象能访问内部类的所有方法和属性(不受防控属性的影响)**

## 访问内部类

1. public class test{  
    public static void main(String args[]){  
    //由于内部类可能会用到外部类的方法和属性,所以必须先实例化外部  
    //类,再实例化内部类  
    Outer.Inner in = new Outer().new Inner();  
    in.print();  
    //Outer out = new Outer();  
    //Outer.Inner in = out.new Inner();  
    }  
   }  
   class Outer{  
    private String msg = "hello,world";  
    class Inner{  
    public void print(){  
    System.*out*.print(msg);  
    }  
    }  
   }

外部的程序访问内部类受到防控属性的影响,内部类的防控属性可以为public,private,default,protecetd,和方法,属性的防控属性一样

**如果是内部类是非static,则外部程序实例化内部类，需要先实例化外部类,所以格式为:**

外部类名称.内部类名称 实例 = new 外部类名称().new 内部类名称()

**如果是内部类是static,则不受外部实例影响,并且成了一个外部类，则格式为:**

外部类名称.内部类名称 实例 = new 外部类名称.内部类名称()

范例如下:

public class test{  
 public static void main(String args[]){  
 Outer.Inner in = new Outer.Inner();  
 in.print();  
 }  
}  
class Outer{  
 private String msg = "hello,world";  
 static class Inner{  
 public void print(){  
 System.*out*.print("hahahha");  
 }  
 }  
}

## 3.在方法中定义内部类

内部类可以在任意位置定义:类中,代码块,方法里(常用)

|  |
| --- |
| public class Main{  public static void main(String args[]) throws Exception {  new Outer().fun();  }  }  //访问方法中的参数和定义的变量  class Outer{  private String msg = "hello,world";  public void fun(){  class Inner{  public void print(int num){//方法参数  double score = 99.7; //方法变量  System.out.println("属性" + Outer.this.msg);  System.out.println("方法参数" + num);  System.out.println("方法变量" + score);  }  }  new Inner().print(100);  }  } |

内部类能直接访问方法中的参数和在方法内定义的变量,JDK1.8才有

在JDK1.7以及之前的版本有严格的要求,方法中的内部类要访问方法的参数或者在方法中定义的变量,必须在参数和方法前添加final

## 匿名内部类

前提:必须要基于接口和抽象类

好处:有时候只需要调用一次某个接口的方法,为了调用某个接口或者抽象方法而定义一个子类很麻烦,所以定义一个匿名内部类

|  |
| --- |
| public class Main{  public static void main(String args[]) throws Exception {  fun(new Message() {  @Override  public void print() {  System.out.println("Message");  }  });  fun(new A() {  @Override  public void print() {  System.out.println("A");  }  });  }  public static void fun(Message msg){  msg.print();  }  public static void fun(A a){  a.print();  }  }  interface Message{  public abstract void print();  }  abstract class A{  abstract public void print();  } |

## 5.内部类继承问题

|  |
| --- |
| class Outer{ //这里为普通,抽象,接口都是一样的  static class A{  }  static interface B{ //这里可以省略static  }  public static abstract class C{  }  }  class X extends Outer{  class E extends A{}  class F implements B{}  class G extends C{}  }  class A1 extends Outer.A{  class E extends Outer.A{} //下面也一样,就不写了  }  class A2 implements Outer.B{}  class A3 extends Outer.C{} |

**当内部类为接口的时候:**接口默认是static,可以省略static