为什么匿名内部类和局部内部类只能访问final变量？

1、因为局部内部类或匿名内部类的对象可以被外部方法作为返回值，返回到方法外使用，那么如果这样的话，就会出现矛盾。

|  |
| --- |
| public class Test {  public static void main(String[] args) {  **Change c = method();**  //这里method()已经结束，按理说局部a早就失效，那么下面调用change()，打印出a的值就矛盾了，即change()访问了一个不存在的变量，这是不允许的。那么怎么解决呢？  c.change();  }    public static **Change** method(){  int a = 10;  **class Inner implements Change**{  public void change(){  System.out.println("a="+a);  }  }  Change in = new Inner();  **return in;**  }  }  interface Change{  void change();  } |

1. 局部内部类和匿名内部类访问局部变量的机制

在java中，类是封装的，内部类也不例外。我们知道，非静态内部类能够访问外部类成员是因为它持有外部类对象的引用 Outer.this， 就像子类对像能够访问父类成员是持有父类对象引用super一样。局部内部类也和一般内部类一样，只持有了Outer.this，能够访问外部类成员，但是它又是如何访问到局部变量的呢？

实际上java是将局部变量作为参数传给了局部内部类的构造函数，而将其作为内部类的成员属性封装在了类中。我们看到的内部类访问局部变量实际上只是访问了自己的成员属性而已，这和类的封装性是一致的。那么上面的代码**实际上**是这样：

|  |
| --- |
| public class TestInner {  public static void main(String[] args) {  Change c = method();  c.change();  //change()访问的就不是method()的局部变量a，  //而是Inner内部类的成员变量a，只是它的值是method()的局部变量a的副本  }    public static Change method(){  int a = 10;  class Inner implements Change{  **private int a;**  **Inner(int a){**  **this.a =a;**  **}**  public void change(){  System.out.println("a=" + **this.a**);  }  }  Change in = new Inner(**a**);//a=10  return in;  }  }  interface Change{  void change();  } |

method()方法的局部变量a失效了，但其实change()方法已经访问的不是method()方法的局部变量a了，而是Inner内部类的成员变量a。

1. 如果在Inner类中，可以修改a的值的话，那么代码就是这样的

|  |
| --- |
| public class TestInner {  public static void main(String[] args) {  Change c = method();  }    public static Change method(){  int a = 10;  class Inner implements Change{  public void change(){  **a = 20; //如果这里可以修改a的值**  **System.out.println("a=" + a);//20**  }  }  Change in = new Inner();  in.change();  **System.out.println(“a=”+a);//从阅读角度，这里预期打印a=20，然而只会打印出a=10，会让人误解，因为在Inner的change()中修改的是Inner内部类对象的this.a，它是method的局部变量a的副本**  return in;  }  }  interface Change{  void change();  } |

1. 结论

为了保证阅读与运行的一致性，只能规定局部变量a（和副本a）不能修改，这样阅读和运行就可保持一致。那么就只能限定a为final。

|  |
| --- |
| public class TestInner {  public static void main(String[] args) {  Change c = method();  c.change();  }    public static Change method(){  **final int a = 10;**  class Inner implements Change{  public void change(){  System.out.println("a=" + **a**);  }  }  Change in = new Inner();  return in;  }  }  interface Change{  void change();  } |

效果如下：

|  |
| --- |
| public class TestInner {  public static void main(String[] args) {  Change c = method();  c.change();  }    public static Change method(){  **final int a = 10;**  class Inner implements Change{  **private final int a;**  **Inner(int a){**  **this.a =a;**  **}**  public void change(){  System.out.println("a=" + **this.a**);  }  }  Change in = new Inner(**a**);//a=10  return in;  }  }  interface Change{  void change();  } |