Atitit.java反编译attilax 总结

# EventQueue.*invokeLater*(new Object(args) {

》》》

EventQueue.*invokeLater*(**new** Runnable( ) {

# this.val$args

Args

应该是上一级的一个变量

**this**.val$template

# 异常处理

**catch** (ParseException localParseException)

{

}

**throw** **new** BasicException(localParseException.getMessage(), localParseException);

Throw语句后退了。

》》》》》

**catch** (ParseException localParseException)

{

**throw** **new** BasicException(localParseException.getMessage(), localParseException);

}

# 增加了形式参数

ThumbNailBuilder tnb = **new** ThumbNailBuilder(32, 32, "com/openbravo/images/yast\_sysadmin.png");

/\* 76 \*/ **this**.peopleread = **new** SerializerRead(tnb) {

》》》对策：：

# 增加了null参数

**public** **static** **final** Formats ***BYTEA*** = **new** FormatsBYTEA(null);

# access$100

为了让内部类能够访问外部类的数据成员，JAVA编译器会为外部类自动生成 static Type access$iii(Outer); 方法，供内部类调用。  
一般的数据成员都是private的，因此这些方法实际上就是访问private的数据成员的后门。

编译器对这些自动生成的方法，在编译时进行检查，是不允许程序员直接来调用的，也就是不能直接调用这些后门。

但也有方法类绕过这些检查。

# Classname.**this**

**多出了此参数**

JPrincipalApp.ExitAction(JPrincipalApp.**this**, "/com/openbravo/images/gohome.png", "Menu.Exit"))

》》直接去掉次参数

# 泛型回归

Collections.*sort*(localArrayList, **new** Comparator<TaxInfo>()

{

**public** **int** compare(TaxInfo paramTaxInfo1, TaxInfo paramTaxInfo2)

{

// TaxInfo

**if** (paramTaxInfo1.getApplicationOrder().intValue() < paramTaxInfo2.getApplicationOrder().intValue())

**return** -1;

**if** (paramTaxInfo1.getApplicationOrder() == paramTaxInfo2.getApplicationOrder())

**return** 0;

**return** 1;

}

});

# 参考

access$000方法和JAVA内部类的反编译-davidyu720-ITPUB博客.htm

反编译出现access$分析 - andylao62的专栏 - 博客频道 - CSDN.NET.htm