public java.lang.String METHOD_1 (final java.lang.String VAR_1) { return STRING_1 . METHOD_2 (VAR_2 . METHOD_3 () . METHOD_4 ()) ; }

public java.lang.String METHOD 1 (final java.lang.String VAR 1) { return VAR 2 . METHOD 3 () . METHOD 4 () ; }

Generation of assert statements

```
// Test method
shouldNeverValidateNullUserIV ( ) { final uk . gov . gchq . gaffer . federatedstore . FederatedAccess access = new uk . gov . gchq . gaffer .
federatedstore . FederatedAccess . Builder ( ) . addingUserId ( null ) . build ( ) ; "<AssertPlaceHolder>" ; }

/// Focal method
isvalidToExecute ( uk . gov . gchq . gaffer . user . User ) { return ( isPublic ) || ( ( null != user ) && ( ( isAddingUser ( user ) )
|| ( ( ! ( isAuthsNullOrEmpty ( ) ) ) && ( isUserHasASharedAuth ( user ) ) ) ) ) ; }

lorg . junit . Assert . assertFalse ( access . isValidToExecute ( null ) )

/// Test method
testClone ( ) { org . apache . flink . api . common . accumulators . DoubleMinimum min = new org . apache . flink . api . common . accumulators .
DoubleMinimum ( ) ; double value = 3.14159265359 ; min . add ( value ) ; org . apache . flink . api . common . accumulators . DoubleMinimum clone = min . clone ( ) ; "<AssertPlaceHolder>" ; }

/// Focal method
getLocalValue ( ) { return null ; }

org . junit . Assert . assertEquals ( value , clone . getLocalValue ( ) , 0.0 )
```

Code summarization public void update() { check Widget (); Utils . paintComponentImmediately (handle); update (false); }

```
forces all outstanding paint requests for the widget

public void setWordWrap(int row, int column, boolean wrap) { prepareCell ( row , column ) ; String wrapValue = wrap ? "" : "nowrap" ; DOM .

setStyleAttribute ( getElement ( row, column ) , "whiteSpace" , wrapValue ) ; }

sets whether the specified cell will allow word wrapping of its contents
```