As you see in the first capture the initialize() method is in line number 34.

Mac80211.cc | Mac80211.ned | BaseLayer.cc | BasicModule.cc

```
1  /* -*- mode:c++ -*- ********************************************************
2   * file:        BasicModule.cc
3   *
4   * author:      Steffen Sroka
5   *              Andreas Koepke
6   *
7   * copyright:   (C) 2004 Telecommunication Networks Group (TKN) at
8   *              Technische Universitaet Berlin, Germany.
9   *
10  *              This program is free software; you can redistribute it
11  *              and/or modify it under the terms of the GNU General Public
12  *              License as published by the Free Software Foundation; either
13  *              version 2 of the License, or (at your option) any later
14  *              version.
15  *              For further information see file COPYING
16  *              in the top level directory
17  ***************************************************************************
18  * part of:     framework implementation developed by tkn
19  **************************************************************************/
20
21
22 #include <iostream>
23 #include "BasicModule.h"
24
25 #define coreEV (ev.isDisabled()||!coreDebug) ? (std::ostream&)ev : EV << loggingName << "::BasicModule: "
26
27 /**
28  * Subscription to NotificationBoard should be in stage==0, and firing
29  * notifications in stage==1 or later.
30  *
31  * NOTE: You have to call this in the initialize() function of the
32  * inherited class!
33  */
34 void BasicModule::initialize(int stage)
35 {
36     cModule *host = findHost(false);
37
38     if (stage == 0)
39     {
40
41         if (hasPar("coreDebug"))
42             coreDebug = par("coreDebug").boolValue();
```

Scrolling a bit down the method is referenced to be in line number 2, so the line numbers have not moved accordingly to the text when scrolling down.

Mac80211.cc | Mac80211.ned | BaseLayer.cc | BasicModule.cc

```
1  */
2  void BasicModule::initialize(int stage)
3  {
4      cModule *host = findHost(false);
5
6      if (stage == 0)
7      {
8
9          if (hasPar("coreDebug"))
10             coreDebug = par("coreDebug").boolValue();
11         else
12             coreDebug = false;
13         if (hasPar("debug"))
14             debug = par("debug").boolValue();
15         else
16             debug = false;
17
18         if (host) {
19             // get the logging name of the host
20             if (host->hasPar("logName"))
21                 loggingName = host->par("logName").stringValue();
22             else
23                 loggingName = host->getName();
24             char tmp[8];
25             sprintf(&tmp[0], "[%d]", host->getIndex());
26             loggingName += tmp;
27         }
28     }
29 }
30
31 cModule *BasicModule::findHost(bool errorIfNotFound) const
32 {
33     cModule *mod;
34     for (mod = getParentModule(); mod != 0; mod = mod->getParentModule()) {
35         cProperties *properties = mod->getProperties();
36         if (properties && properties->getAsBool("node"))
37             break;
38     }
39     if (errorIfNotFound && !mod)
40         error("findHost(): host module not found (it should have a property named node)");
41
42
```