The Complete Guide

All you need to know about Joone

http://www.joone.org
Paolo Marrone (pmarrone@users.sourceforge.net)
# Table of Contents

1 Introduction .................................................................................................................. 6
   1.1 Intended Audience .................................................................................................. 6
   1.2 What is Joone ........................................................................................................ 6
      1.2.1 Custom systems ............................................................................................. 6
      1.2.2 Embedded systems ....................................................................................... 7
      1.2.3 Portable systems ........................................................................................... 7
   1.3 About this Guide ...................................................................................................... 7
   1.4 Acknowledgements ................................................................................................. 9

2 Getting and Installing Joone .......................................................................................... 10
   2.1 Platform and requirements ................................................................................... 10
   2.2 Installing the binary distribution ........................................................................... 10
      2.2.1 The Core Engine ............................................................................................ 10
      2.2.2 The GUI Editor ............................................................................................. 12
   2.3 Building from the source distribution .................................................................... 18
      2.3.1 Prerequisites .................................................................................................. 18
      2.3.2 Getting the last released source code ............................................................ 19
      2.3.3 Getting the CVS sources .............................................................................. 19
      2.3.4 Compiling ...................................................................................................... 20

3 Inside the Core Engine .................................................................................................. 21
   3.1 Basic Concepts ........................................................................................................ 21
   3.2 The Transport Mechanism ..................................................................................... 21
   3.3 The Processing Elements ....................................................................................... 24
      3.3.1 The Layers ...................................................................................................... 24
         3.3.1.1 The Linear Layer .................................................................................... 24
         3.3.1.2 The Sigmoid Layer ............................................................................... 25
         3.3.1.3 The Tanh Layer .................................................................................... 25
         3.3.1.4 The Logarithmic Layer ......................................................................... 25
         3.3.1.5 The Delay Layer ................................................................................... 25
         3.3.1.6 The Context Layer ............................................................................... 26
         3.3.1.7 The WinnerTakeAll Layer .................................................................... 27
         3.3.1.8 The Gaussian Layer .............................................................................. 27
      3.3.2 The Synapses .................................................................................................... 29
         3.3.2.1 The Direct Synapse ............................................................................... 29
         3.3.2.2 The Full Synapse .................................................................................. 30
         3.3.2.3 The Delayed Synapse ........................................................................... 30
         3.3.2.4 The Kohonen Synapse ......................................................................... 31
         3.3.2.5 The Sanger Synapse ............................................................................ 32
   3.4 The Monitor: a central point to control the neural network .................................... 32
      3.4.1 The Monitor as a container of the NN Parameters ........................................ 33
      3.4.2 The Monitor as the NN controller ................................................................. 34
      3.4.3 Managing the events ...................................................................................... 35
      3.4.4 How the patterns and the internal weights are represented ............................ 37
3.4.4.1 The Pattern ................................................................. 37
3.4.4.2 The Matrix ................................................................. 38
3.5 Technical details ................................................................. 38
  3.5.1 The Layer abstract class .................................................. 39
    3.5.1.1 The Recall Phase ...................................................... 40
    3.5.1.2 The Learning Phase .................................................. 41
  3.5.2 Connecting a Synapse to a Layer ........................................ 41
  3.5.3 The Synapse abstract class ............................................... 42

4 I/O components: a link with the external world ......................... 45
  4.1 The Input mechanism ........................................................ 45
  4.2 The Output: using the outcome of a neural network ................. 46
  4.3 The Switching Mechanism .................................................. 47
    4.3.1 The InputSwitch ......................................................... 48
    4.3.2 The OutputSwitchSynapse ............................................. 48
  4.4 The Validation Mechanism .................................................. 48
  4.5 Technical details ............................................................. 49
    4.5.1 The StreamInputSynapse .............................................. 51
    4.5.2 The StreamOutputSynapse ............................................ 53
    4.5.3 The Switching mechanism’s object model .......................... 53
      4.5.3.1 The InputSwitchSynapse ........................................ 54
      4.5.3.2 The OutputSwitchSynapse ....................................... 54
      4.5.3.3 The LearningSwitch ............................................... 55

5 Teaching a neural network: the supervised learning .................... 56
  5.1 The Teacher component ................................................... 56
  5.2 Technical details ........................................................... 58

6 The Plugin based expansibility mechanism .................................. 59
  6.1 The Input Plugins .......................................................... 59
  6.2 The Monitor Plugins ........................................................ 59
  6.3 The Scripting Mechanism .................................................. 61
  6.4 Technical details ............................................................ 62
      6.4.1 The Input Plugins object model .................................. 62
      6.4.2 The Monitor Plugin object model ................................ 65
      6.4.3 The Scripting mechanism object model ............................ 67

7 Using the Neural Network as a Whole ...................................... 70
  7.1 The NeuralNet object ....................................................... 70
  7.2 Technical details ............................................................ 72

8 Common Architectures .......................................................... 73
  8.1 Temporal Feed Forward Neural Networks ................................ 73
      8.1.1 Managing Temporal Series ......................................... 73
        8.1.1.1 Preprocessing ..................................................... 73
  8.2 Unsupervised Neural Networks ............................................ 76
      8.2.1 Kohonen Self Organized Maps ..................................... 76
        8.2.1.1 Example: a character recognition system .................. 76

9 Applying Joone ................................................................. 83
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 A simple (but useless) neural network</td>
<td>83</td>
</tr>
<tr>
<td>9.2 A real implementation: the XOR problem</td>
<td>84</td>
</tr>
<tr>
<td>9.3 Saving and restoring a neural network</td>
<td>88</td>
</tr>
<tr>
<td>9.3.1 The simplest way</td>
<td>88</td>
</tr>
<tr>
<td>9.3.2 Using a NeuralNet object</td>
<td>89</td>
</tr>
<tr>
<td>9.4 Using the outcome of a neural network</td>
<td>92</td>
</tr>
<tr>
<td>9.4.1 Writing the results to an output file</td>
<td>92</td>
</tr>
<tr>
<td>9.4.2 Getting the results into an array</td>
<td>93</td>
</tr>
<tr>
<td>9.4.3 Using multiple input patterns</td>
<td>93</td>
</tr>
<tr>
<td>9.4.4 Using only one input pattern</td>
<td>96</td>
</tr>
<tr>
<td>10 The LGPL Licence</td>
<td>98</td>
</tr>
</tbody>
</table>
I would like to present the objectives that I had in mind when I started to write the first lines of code of Joone.
My dream was (and still is) to create a framework to implement a new approach the use of neural networks.
I felt this necessity because the biggest (and unresolved until now) problem is to find the fittest network for a given problem, without falling into local minima, thus finding the best architecture.
Okay - you'll say - this is what we can do simply by training some randomly initialised neural network (NN) with a supervised or unsupervised algorithm.
Yes, it's true, but this is just scholastic theory, because training only one neural network, especially for hard problems of the real life, is not enough.
To find the best neural network is a really hard task because we need to determine many parameters of the net such as the number of the layers, how many neurons for each layer, the transfer function, the value of the learning rate, the momentum, etc... often causing frustrating failures.
The basic idea is to have an environment to easily train many neural networks in parallel, initialised with different weights, parameters or different architectures, so the user can find the best NN simply by selecting the fittest neural network after the training process.
Not only that but this process can continue retraining the selected NNs until some final parameter is reached (i.e. a low RMSE value) like a distillation process. The best architecture is discovered by Joone, not by the user! Many programs today exist that permit selection of the fittest neural network applying a genetic algorithm. I want to go beyond this, because my goal is to build a flexible environment programmable by the end user, so any existing or newly discovered global optimisation algorithm can be implemented. This is why Joone has its own distributed training environment and why it is based on a cloneable engine.

My dreams aren't finished, because another one was to make easily usable and distributable a trained NN by the end user. For example, I'm imagining an assurance company that continuously trains many neural networks on customer’s risk evaluation¹ (using the results of historical cases), distributing the best ‘distilled’ resulting network to its sales force, that they can use it on their mobile devices.
This is why Joone is serializable and remotely transportable using any wired and wireless protocol, and it is easily runnable using a simple, small and generalized program.
Moreover, my dream can become a more solid reality thanks to the advent of handheld devices like mobile phones and PDA having inside a java virtual machine. Joone is ready to run on them, too.

Hoping you'll find Joone interesting and useful, I thank you for your interest to it.

Paolo Marrone

¹ The ethics (and the law in many countries) forbids to make racial, sexual, religious (and others) discriminations. Consequently, a decisional system based on such personal characteristics cannot be built.

http://www.joone.org
1 Introduction

1.1 Intended Audience

This paper describes the technical concepts underlying the core engine of Joone, explaining in detail the architectural design that is at its foundation. This paper is intended to provide the users - or anyone interested to use Joone - with the knowledge of the basic mechanisms of the core engine, so that anyone can understand how to use it and expand it to resolve one’s needs. A basic knowledge of the basic concepts underlying the artificial neural networks is required, consequently, who doesn’t own such a know-how should read some good introductory book on the argument.

1.2 What is Joone

Joone (http://www.joone.org/) is a Java framework to build and run AI applications based on neural networks. Joone applications can be built on a local machine, be trained on a distributed environment and run on whatever device. Joone consists of a modular architecture based on linkable components that can be extended to build new learning algorithms and neural networks architectures.

All the components have some basic specific features, like persistence, multithreading, serialization and parameterisation. These features guarantee scalability, reliability and expansibility, all mandatory features to reach the final goal to represent the future standard on the AI world.

Joone applications are built out of components. Components are pluggable, reusable, and persistent code modules. Components are written by developers. AI experts and designers can build applications by gluing together components with graphical editors, and controlling the logic with scripts.

Around the components will be based all the modules and applications written with Joone. Joone can be used to build Custom Systems, adopted in an Embedded manner to enhance an existing application, or employed to build applications on Portable Systems:

1.2.1 Custom systems

A great need of the industrial market is to have the possibility to resolve business problems suitable with neural networks (or with AI applications in general). Joone wants to
represent the optimal solution to build applications to satisfy such needs (i.e. bank loan assessment, sales forecasting, etc.).

Its characteristics are optimal to build custom applications driven from the user’s needs, where it’s important to have flexibility, scalability and portability.

Each enhancement of Joone will be compatible also with the necessity of build applications more quickly than other product on the market, so Joone can gain a large market share and become the most used neural network framework.

1.2.2 Embedded systems

Into the core engine, the components are the bricks to build whatever neural network architecture. Their purpose is to create AI applications writing Java code that uses the Joone’s API.

In the respect of the goal that aims to obtain a wide adoption of Joone from the market, the license of the core engine is the Lesser General Public License (LGPL), so everyone can freely embed the engine into existing or new applications. This will never change.

The business model of Joone contemplates the possibility of provide more components to satisfy the users needs to create several neural network architectures and algorithms, so they can embed Joone into whatever application (i.e. data mining systems, automatic categorization for search engines, customer classification for One-to-One marketing, etc.)

1.2.3 Portable systems

One long-term goal of Joone is to become the basic framework to provide a computational engine to AI applications suitable for the mobile devices (phones, PDA, etc.).

The demand for software products available for such kind of devices is growing, therefore in the future a new market of applications to satisfy these needs will be open, gaining the interest of the industrial world.

Joone wants to be present in that market and represent the main framework to distribute and run personal or corporate AI applications (i.e. handwriting and voice recognition, support to the sales force, marketing or financial forecasting, etc.).

The core engine of Joone is already suitable for small devices, having a small footprint and being runnable on Personal Java environments.

1.3 About this Guide

This complete guide is composed by the following chapters (the asterisks indicate the skill required to correctly understand the exposed concepts, as listed at the end of this paragraph):

Chapter 1 – Introduction (*)

This Chapter contains a brief description of Joone, what it is and what are its possible applications in several fields of the professional world.
Chapter 2 – Getting and installing Joone (*)
This is a starter guide to learn how to download and install all the Joone framework and how to obtain a runnable version from the source code.

Chapters 3-7 – Concepts and technical details (**)(***)
These chapters illustrate the basic concepts underlying the core engine. They explains the main features of the core engine from a functional point of view, and, for whose that are interested to the technical implementation, each chapter ends with a paragraph named ‘technical details’, where a more detailed look about how the described features have been implemented is given.

Chapter 8 – Common Architectures (**) 
This is a practical guide about how to build the most common neural network architectures, like the temporal, recurrent, unsupervised and the mixed ones. For each of them an example is built using the visual editor. This Chapter can be intended as a complement of the Editor User Guide, and its goal is to give a first look about some potential applications of Joone. / TO BE COMPLETED /

Chapter 9 – Applying Joone (***)
This Chapter explains the main features of Joone using concrete and useful examples written in java code. Applying the programming techniques described in this chapter everyone can build a custom java application that uses joone as internal neural network engine. / TO BE COMPLETED /

Legend:
* No specific skill required
** Basic knowledge about artificial neural networks
*** Good understanding of UML and Java code

http://www.joone.org
1.4 Acknowledgements

Joone was made possible thanks to the many people that have agreed my initial idea and have extended the initial code adding new ideas, suggestions and, mainly, good and often documented source code. This is the demonstration that also in a complex stuff like the Artificial Intelligence, thanks to the Open Source model it's possible to obtain the collaboration of valid and skilled programmers to build a complete, stable and powerful framework.

Paolo Marrone, the founder and project manager of Joone, wants to thank three persons that have contributed continuously for a long period of time, writing good java code, and also supporting me with very interesting proposals and suggestions (listed in alphabetical order):

Harry Glasgow  
Julien Norman  
Paul Sinclair

Thanks also to the following people that have collaborated to Joone in the past:

Mark Allen, Ka-Hing Cheung, Jan Erik Garshol, Jack Hawkins, Olivier Hussenet, Shen Linlin, Christian Ribeaud, Anat Rozenzon, Thomas Lionel Smets

Do you want to see your name listed above? Join us: any contribution is always welcome, therefore if you are interested to participate to Joone, contact me specifying what are your past experiences on artificial neural networks and java programming, and what you'd like to do for Joone - Write only if really interested to participate -

I want also to thank all the authors of the following O.S. external packages used by Joone:

- **JHotDraw**  [http://sourceforge.net/projects/jhotdraw](http://sourceforge.net/projects/jhotdraw)
- **BeanShell**  [http://sourceforge.net/projects/beanshell](http://sourceforge.net/projects/beanshell)
- **jEdit-Syntax**  [http://sourceforge.net/projects/jedit-syntax](http://sourceforge.net/projects/jedit-syntax)
- **Log4J**  [http://jakarta.apache.org/log4j](http://jakarta.apache.org/log4j)
- **HSSF-POI**  [http://jakarta.apache.org/poi](http://jakarta.apache.org/poi)
- **VisAD**  [http://www.ssec.wisc.edu/~billh/visad.html](http://www.ssec.wisc.edu/~billh/visad.html)

A particular acknowledgment to:

- [SourceForge.net](http://sourceforge.net), thanks to which all this has been possible
- **Nathan Hindley** who has designed and realized the amazing web site at [http://www.joone.org](http://www.joone.org) (you can contact him at kelticdanor@iprimus.com.au)
- **Zero G Software, Inc.** for InstallAnywhere, the multiplatform auto installer program used by Joone

http://www.joone.org
2 Getting and Installing Joone

2.1 Platform and requirements

Joone is written in 100% pure Java and can run on whatever platform for which a Java Runtime Environment v. 1.4 or later is available.

Due to his direct experience, or because he has received information from other users, the author can assure the compatibility of Joone with the following operating systems:

- Linux
- Mac OSX
- Windows 2000
- Windows XP

About the memory requirement, it depends on the complexity of the neural network used, but generally the availability of at least 128MB of RAM, even if not mandatory, is strongly recommended.

Due to its small footprint, a minimal version of the Joone’s core engine can run also on mobile devices (PDA) running J2ME Personal Profile. The author ran without problems the sample XOR neural network on a Compaq IPAQ device provided with 32MB of flash memory using successfully both Jeode and IBM J9 JVMs.

2.2 Installing the binary distribution

Joone is distributed both in source and compiled form. The compiled distribution (named also the binary distribution) is available both for the core engine and the GUI editor. We’ll see how to download and install them on your machine.

2.2.1 The Core Engine
The compiled form of the core engine can be useful to run a whatever application written in java that uses the Joone’s engine API, as deeply described in the next chapters. All the classes are contained into the library joone-engine.jar. This library cannot run stand-alone, as it doesn’t contain any main class, but it must be put into the classpath of the application that needs to use Joone.

Depending on which Joone engine’s packages are used, you need also to put in the classpath some external packages provided in a separate downloadable file.

Here are explained the steps to execute to correctly install the core engine’s libraries:

1. Download the core engine’s binary distribution file joone-engine-x.y.z.zip (where x, y and z are respectively the major/minor version and the build number of the last available distribution)
2. Download joone-ext.zip, the file containing the needed external libraries
   Unzip both the above files into a predefined directory of your file system (we’ll name it <base_dir>). At this point you should have a directory tree as below (we omitted the unessential files):

   <base_dir>
   
   Joone-engine.jar
   ...
   <ext>
   
   bsh.jar
   crimson.jar
   jakarta-poi.jar
   log4j.jar
   ...
   <samples>
   ...

3. Put the joone-engine.jar and also the <ext>*.jar files into your classpath
4. Run your own application

Depending on the engine’s packages your application uses, you need to put only the needed libraries on your classpath, as depicted in the following table:

<table>
<thead>
<tr>
<th>Library</th>
<th>Purpose</th>
<th>When used</th>
</tr>
</thead>
<tbody>
<tr>
<td>joone-engine.jar</td>
<td>The Joone’s core engine</td>
<td>Mandatory</td>
</tr>
<tr>
<td>log4j.jar</td>
<td>The configurable logger</td>
<td>Mandatory</td>
</tr>
<tr>
<td>bsh.jar</td>
<td>The BeanShell interpreter</td>
<td>Optional. Needed only if you want to use the scripting features</td>
</tr>
<tr>
<td>jakarta-poi.jar</td>
<td>The Jakarta Excel libraries</td>
<td>Optional. Needed only if you use the Excel Input/Output synapses</td>
</tr>
<tr>
<td>jh.jar</td>
<td>The Java Help libraries</td>
<td>Never. Used only in conjunction with the GUI editor contained into the joone-editor.jar file</td>
</tr>
<tr>
<td>jhotdraw.jar</td>
<td>The drawing framework</td>
<td>Never. Used only in conjunction with the GUI editor contained into the joone-editor.jar file</td>
</tr>
<tr>
<td>xalan.jar crimson.jar</td>
<td>The XML libraries</td>
<td>Never. Used only in conjunction with the GUI editor contained into the joone-editor.jar file</td>
</tr>
<tr>
<td>visad.jar</td>
<td>The external graphic library to plot graphs</td>
<td>Never. Used only in conjunction with the GUI editor contained into the joone-editor.jar file</td>
</tr>
</tbody>
</table>

http://www.joone.org
As you can see, only the first two libraries have to be present into your classpath, whereas the following two are needed only if you use some specific feature of the core engine.

The last four libraries, instead, must be used only in conjunction with the GUI editor contained into the joone-editor.jar file; but, in this case, you don’t need to install manually the editor, as you can use an auto-installer, like depicted in the following paragraph.

2.2.2 The GUI Editor

To permit to everyone to correctly install and run the GUI Editor, this is distributed in an auto-installing form. Using the ZeroG Software InstallAnywhere product we have prepared auto-installers for the following platforms:

- Linux
- Windows
- Mac OSX

You don’t need to be aware about the installation of the Java runtime environment, as all the installers are available both with and without an embedded java virtual machine (except for the Mac version, because on the OSX platform a suitable JVM is already installed).

All you need to do is to download the appropriate installer depending on your platform, and run it as described below:

**Linux Instructions:**
After downloading open a shell and, cd to the directory where you downloaded the installer.
At the prompt type: `sh ./JooneEditorX_Y_Z.bin`
If you do not have a Java virtual machine installed, be sure to download the package which includes one. Otherwise you may need to download one from Sun's Java web site or contact your OS manufacturer.

**Windows Instructions:**
After downloading, double-click JooneEditorX.Y.Z.exe
If you do not have a Java virtual machine installed, be sure to download the package which includes one.

**Mac OS X Instructions:**

---

1 InstallAnywhere is a registered trademark of Zero G Software, Inc.
Mac OS is a registered trademark of Apple Computer, Inc.
Solaris and Java are trademarks of Sun Microsystems, Inc.
Windows is a registered trademark of Microsoft Corporation.
All other marks are properties of their respective owners.

http://www.joone.org 12
After downloading, double-click JooneEditorX.Y.Z.zip (Requires Mac OS X 10.0 or later).
The compressed installer should be recognized by Stuffit Expander and should automatically be expanded after downloading. If it is not expanded, you can expand it manually using Stuffit Expander 6.0 or later.
If you have any problems launching the installer once it has been expanded, make sure that the compressed installer was expanded using Stuffit Expander.

After the launch of the installer, you should see the following panel:

By clicking on the Next button you can advance in the installation process. In any moment, pressing the Cancel button, you can abort and exit from the installation.
In this panel you must specify the directory where you want to install Joone. The Choose button will open an explorer window, where you can make the choice, whereas using the ‘Restore Default Folder’ button you can reset the directory to its initial value.
Here you can choose where to put the Joone launcher’s icon.

This panel can contain several available choices depending on the platform where you’re installing on.
By checking the ‘Create Icon for All Users’ box – if not greyed – will give the visibility of the icon to all the users of the system.
At this point a panel showing the summary of the made choices will appears. If it's all ok, press the Next button, otherwise, pressing the Previous button, you can go back to the previous panels to review and change some parameter.
Now the panel showing the GNU LESSER GENERAL PUBLIC LICENSE, the license under which Joone is released.

Be aware: Open Source **doesn’t mean** ‘no license’, hence, before to continue, you must carefully read the license agreement, and press the ‘Install’ button only if **you agree to the terms of the license**. A copy of the LGPL license is contained in one of the last Chapters of this paper.

If you continue, the installation process starts and a panel indicating the progress will appear.
At the end, the following panel indicating the success of the operation will be shown.
Press Done to exit.

After the installation, you should found a file named Joone (or Joone.bat for the Windows platforms) into the chosen installation directory. You must execute it (a double click from within the file explorer should work on all the platforms) to run the editor. If you have chosen to add a shortcut to the Start Menu or to the Desktop, you can press it to start the application.

2.3 Building from the source distribution

In this paragraph we'll show how to build joone starting from the source distribution, but first of all you need to install on your system some useful tool.

2.3.1 Prerequisites

You need to have installed on your system:

1. a Java Development Kit version 1.4 or above (http://java.sun.com)
2. the ANT build tool v. 1.5.1 or above (http://ant.apache.org)
3. the sources of joone, and to do it, you can either get the last released version, or download the last (unstable) code from the CVS repository.
The instructions to get Java JDK and ANT installed and running on your system go over the scope of this document, but you can read a lot of documentation available on Internet. Now we'll see how to get the joone's source code.

### 2.3.2 Getting the last released source code

The released version is preferable if you need to use a stable and tested version of joone, without be worried about possible unknown or not fixed bugs.

To do it, open your preferred browser and simply go to the download page of joone at [http://sourceforge.net/project/showfiles.php?group_id=22635](http://sourceforge.net/project/showfiles.php?group_id=22635) and get the files joone-engine-x.y.z.zip (the core engine), joone-editor-x.y.z.zip (the GUI editor) and joone-ext.xip (the external libraries).

Note: x, y and z are respectively the major/minor version and the build number of the last available distribution.

Unzip them on a directory of your file system (say `c:\joone` for Windows or `/home/joone` for Linux).

### 2.3.3 Getting the CVS sources

If you need to use some new feature of joone still not released, you can get the last developed source code from the CVS repository.

To do it, you need to have a cvs client installed on your system. Unix/Linux systems normally have it already installed, whereas for the Windows system go to [http://www.cvshome.org/](http://www.cvshome.org/) and download a suitable version for your OS.

The CVS repository of Joone is hosted at SourceForge, so here is an extract from the instructions gave from SF cvs page:

```
"...This project's SourceForge.net CVS repository can be checked out through anonymous (pserver) CVS with the following instruction set. The module you wish to check out must be specified as the modulename. When prompted for a password for anonymous, simply press the Enter key.

    cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/joone login
    cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/joone co joone

Information about accessing this CVS repository may be found in our document titled, "Basic Introduction to CVS and SourceForge.net (SF.net) Project CVS Services" (http://sourceforge.net/docman/display_doc.php?docid=14033&group_id=1).

Updates from within the module's directory do not need the -d parameter.

**NOTE:** UNIX file and directory names are case sensitive. The path to the project CVSROOT must be specified using lowercase characters (i.e. /cvsroot/joone)"
```

Anyway you need to download the file containing the external libraries (joone-ext.zip) and unzip it into the same directory where you have checked out from cvs (read at the previous chapter how to download it).

http://www.joone.org
2.3.4 Compiling
Regardless of which repository you have decided to download from, you should have on your file system the following directory tree:

```xml
<base_dir>
  <joone>
    <lib>
      <org>
        <joone>
          <data>
          <edit>
          <engine>
          <exception>
          <images>
          <inspection>
          <io>
          <net>
          <samples>
          <script>
          <util>
```

Before to start the build process, you need to edit the build.xml file found in the root installation directory. Open it with a text editor and search the following line:

```xml
<property name="base" value="/usr/SourceForge"/>
```

change the path into the quotes with your previous chosen installation directory (e.g. c:\joone or /home/joone) and save the file.

Assuming you have the Java JDK and ANT correctly installed and running (to verify, try to launch in a console the commands 'javac' and 'ant'; you need to cd into the installation directory and launch at the prompt the command 'ant').

At the end of the operation, under the installation directory, if no error occurs, you should have a subdirectory named 'build' containing all the compiled classes.

At this point, to run the GUI editor, you need to:
1. Put the <base_dir>/build directory and all the <base_dir>/lib/*.jar files on your classpath
2. Open a console and launch the following command: java org.joone.edit.JoonEdit

The main window of the editor should appear.