

Section 10 – CLIPS Java Native Interface

This section describes the CLIPS Java Native Interface (CLIPSJNI) and the examples demonstrating the integration of CLIPS with a Swing interface. The examples have been tested using Java version 1.5.0_13 running on Mac OS X 10.5.1 and Java version 1.6.0_02 running on Windows XP SP2.

10.1 CLIPSJNI DIRECTORY STRUCTURE

When unzipped the CLIPSJNI project file contains the following directory structure:

```
CLIPSJNI
  examples
    AnimalDemo
      resources
    AutoDemo
      resources
    SudokuDemo
      resources
    WineDemo
      resources
  java-src
  CLIPSJNI
  library-src
```

If you are using the CLIPSJNI with either Windows XP or Mac OS X, then the native CLIPS library is already contained in the top level CLIPSJNI directory. On other systems, you must create a native library using the source files contained in the library-src directory before you can utilize the CLIPSJNI. The CLIPSJNI jar file is also contained in the top level CLIPSJNI directory. The source files used to create the jar file are contained in the java-src directory.

10.2 RUNNING CLIPSJNI IN COMMAND LINE MODE

You can invoke the command line mode of CLIPS through CLIPSJNI to interactively enter commands while running within a Java environment.

On Windows XP, launch the Command Prompt application (select Start > All Programs > Accessories > Command Prompt). Set the directory to the CLIPSJNI top level directory (using the cd command).

On Mac OS X, launch the Terminal application (located in the Applications/Utilities directory). Set the directory to the CLIPSJNI top level directory (using the cd command).

From the CLIPSJNI directory, enter the following command:

```
java -cp CLIPSJNI.jar CLIPSJNI.Environment
```

The CLIPS banner and command prompt should appear:

```
CLIPS (Quicksilver Beta 1/30/08)
CLIPS>
```

10.3 RUNNING THE SWING DEMO PROGRAMS

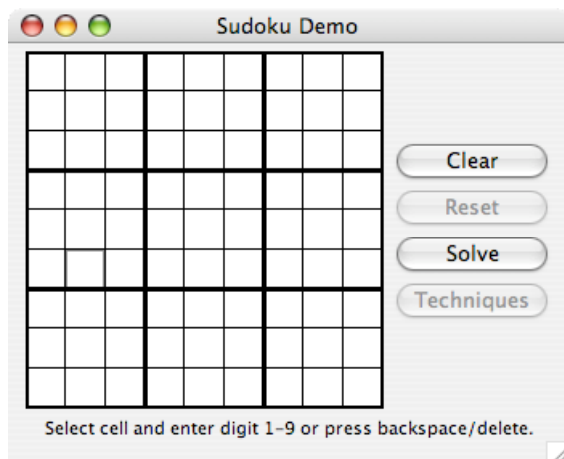
The Swing CLIPSJNI demonstration programs can be run on Windows XP or Mac OS X using the precompiled native libraries in the CLIPSJNI top level directory. On other systems a native library must first be created before the programs can be run.

10.3.1 Running the Demo Programs on Mac OS X

Launch the Terminal application (located in the Applications/Utilities directory). Set the directory to the CLIPSJNI/examples/SudokuDemo directory (using the cd command). To run the Sudoku demo, enter the following command:

```
java -cp ../../CLIPSJNI.jar -Djava.library.path=../../ SudokuDemo
```

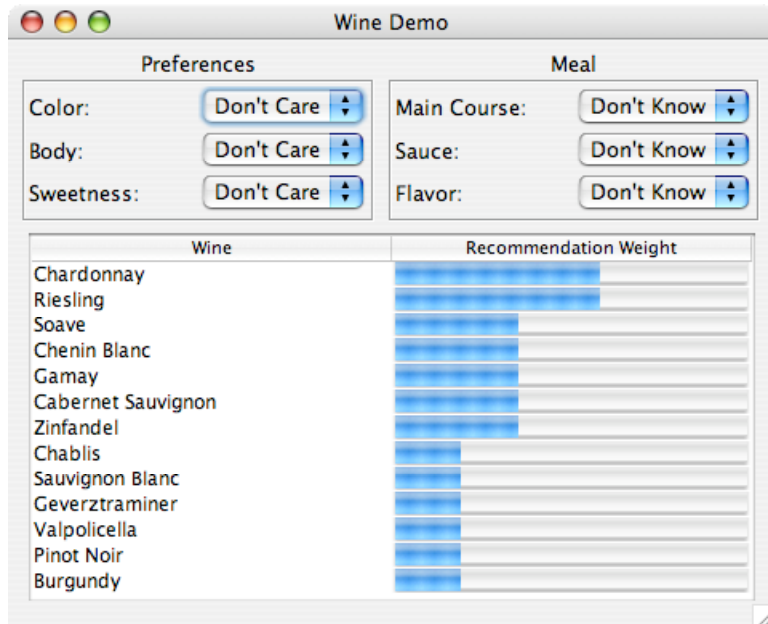
The Sudoku Demo window should appear:



To run the Wine demo, set the directory to the CLIPSJNI/examples/WineDemo directory and enter the following command:

```
java -cp ../../CLIPSJNI.jar -Djava.library.path=../../ WineDemo
```

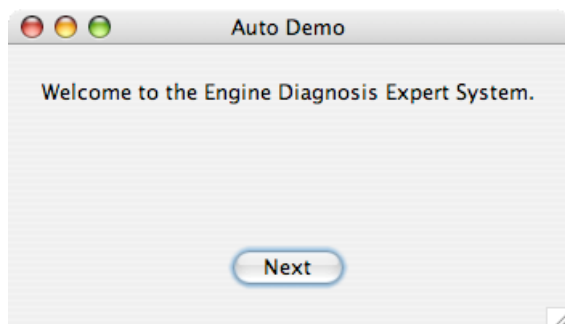
The Wine Demo window should appear:



To run the Auto demo, set the directory to the CLIPSJNI/examples/AutoDemo directory and enter the following command:

```
java -cp ../../CLIPSJNI.jar -Djava.library.path=../../ AutoDemo
```

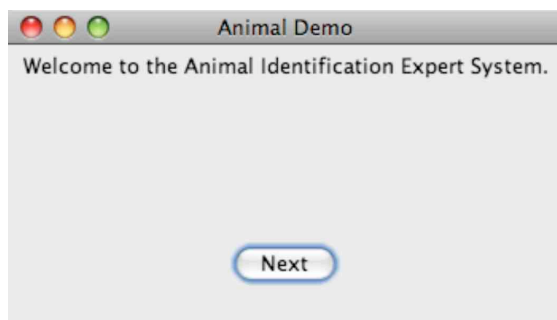
The Auto Demo window should appear:



To run the Animal demo, set the directory to the CLIPSJNI/examples/AnimalDemo directory and enter the following command:

```
java -cp ../../CLIPSJNI.jar -Djava.library.path=../../ AnimalDemo
```

The Animal Demo window should appear:

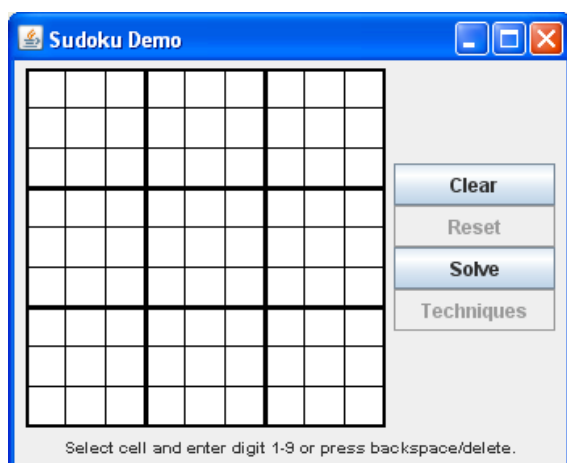


10.3.2 Running the Demo Programs on Windows XP

Launch the Command Prompt application (select Start > All Programs > Accessories > Command Prompt). Set the directory to the CLIPSJNI/examples/SudokuDemo directory (using the cd command). To run the Sudoku demo, enter the following command:

```
java -cp .;../../CLIPSJNI.jar -Djava.library.path=../../ SudokuDemo
```

The Sudoku Demo window should appear:



To run the Wine demo, set the directory to the CLIPSJNI/examples/WineDemo directory and enter the following command:

```
java -cp .;../../CLIPSJNI.jar -Djava.library.path=../../ WineDemo
```

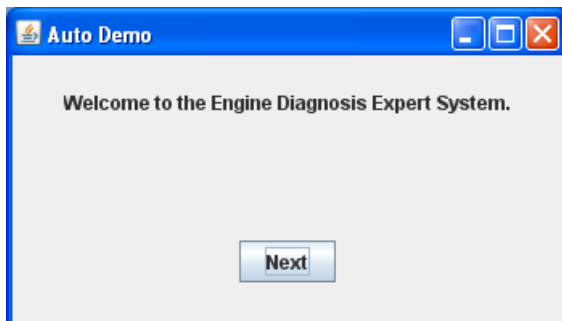
The Wine Demo window should appear:

Wine	Recommendation Weight
Chardonnay	0.8
Riesling	0.7
Soave	0.6
Chenin Blanc	0.5
Gamay	0.4
Cabernet Sauvignon	0.3
Zinfandel	0.2
Chablis	0.1
Sauvignon Blanc	0.0
Geverztraminer	0.0
Valpolicella	0.0
Pinot Noir	0.0
Burgundy	0.0

To run the Auto demo, set the directory to the CLIPSJNI/examples/WineDemo directory and enter the following command:

```
java -cp .;../../CLIPSJNI.jar -Djava.library.path=../../ AutoDemo
```

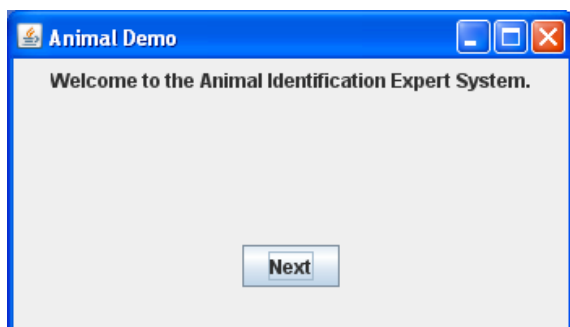
The Auto Demo window should appear:



To run the Animal demo, set the directory to the CLIPSJNI/examples/AnimalDemo directory and enter the following command:

```
java -cp .;../../CLIPSJNI.jar -Djava.library.path=../../ AnimalDemo
```

The Animal Demo window should appear:



10.4 CREATING THE CLIPSJNI JAR FILE

If you wish to add new functionality to the CLIPSJNI package, such as new Java methods which may call existing or new native functions, it is necessary to recreate the CLIPSJNI jar file. The CLIPSJNI distribution already contains the precompiled CLIPSJNI jar file in the top level CLIPSJNI directory, so if you are not adding new functionality to the CLIPSJNI package, you do not need to recreate the jar file (unless you want to create a jar file using a Java version prior to version 1.5.0_13).

To create the jar file, first open a terminal window where you can enter Java tool commands. On Mac OS X, launch the Terminal application (located in the Applications/Utilities directory). On Windows XP, launch the Command Prompt application (select Start > All Programs > Accessories > Command Prompt).

Using the appropriate commands (cd on Mac OS X and Windows XP), set the current directory to CLIPSJNI/java-src , then enter the following command to compile the CLIPSJNI java source:

```
javac CLIPSJNI/*.java
```

Once compiled, enter the following command to place the class files in a jar file:

```
jar -cf CLIPSJNI.jar CLIPSJNI/*.class
```

Once the CLIPSJNI.jar file is created, move it from the CLIPSJNI/java-src directory to the top level CLIPSJNI directory.

If you are adding new native functions to the CLIPSJNI package, it is also necessary to create the JNI header file which will be used to compile the native library. While you are still in the CLIPSJNI/java-src directory, enter the following command:

```
javah -jni CLIPSJNI.Environment
```

This command creates a file named `CLIPSJNI_Environment.h` which must be moved from the `CLIPSJNI/java-src` directory to the `CLIPSJNI/library-src` directory.

10.5 CREATING THE CLIPSJNI NATIVE LIBRARY

The CLIPSJNI distribution already contains a precompiled universal library for Mac OS X, `libCLIPSJNI.jnilib`, and for Windows, `CLIPSJNI.dll`, in the top level CLIPSJNI directory. It is necessary to create a native library only if you are using the CLIPSJNI with an operating system other than Mac OS X or Windows. You must also create the native library if you want to add new functionality to the CLIPSJNI package by adding additional native functions. The steps for creating a native library varies between operating systems, so some research may be necessary to determine how to create one for your operating system.

10.5.1 Creating the Native Library on Mac OS X

Launch the Terminal application (located in the Applications/Utilities directory). Set the directory to the `CLIPSJNI/library-src` directory (using the `cd` command).

To create a universal native library that can run on both Intel and Power PC architectures, enter the following command:

```
make -f makefile.macunv
```

You can ignore the following warnings generated when the `CLIPSJNI_Environment.c` file is compiled:

```
CLIPSJNI_Environment.c:###: warning: cast to pointer from integer of different size
```

To create a native library that runs only on the Intel architecture, enter the following command:

```
make -f makefile.macint
```

To create a native library that runs only on the Power PC architecture, enter the following command:

```
make -f makefile.macppc
```

Once you have create the native library, copy the `libCLIPSJNI.jnilib` file from the `CLIPSJNI/library-src` to the top level CLIPSJNI directory.

10.5.2 Creating the Native Library on Windows XP

The following steps assume you have Microsoft Visual C++ 2008 Express installed. First, launch the Command Prompt application (select Start > All Programs > Accessories > Command Prompt). Set the directory to the CLIPSJNI/library-src directory (using the cd command).

To create the native library DLL, enter the following command:

```
nmake -f makefile.win
```

Once you have create the native library, copy the CLIPSJNI.dll file from the CLIPSJNI/library-src to the top level CLIPSJNI directory.

10.5.3 Creating the Native Library On Other Systems

The file makefile.linux is intended to generate a native library for Linux systems using Java version 1.4.2. It can be invoked using the following command:

```
make -f makefile.linux
```

The shared library generated by this makefile is libCLIPSJNI.so. You will likely need to change the directory paths in the makefile to the appropriate location for your Java installation.

10.6 RECOMPILING THE SWING DEMO PROGRAMS

If you want to make modification to the Swing Demo programs and recompile them, you can use one of the following commands to do so (assuming you are in the appropriate directory for the example and the CLIPSJNI.jar file is present in the top level CLIPSJNI directory):

```
javac -classpath ../../CLIPSJNI.jar SudokuDemo.java
```

```
javac -classpath ../../CLIPSJNI.jar WineDemo.java
```

```
javac -classpath ../../CLIPSJNI.jar AutoDemo.java
```

```
javac -classpath ../../CLIPSJNI.jar AnimalDemo.java
```

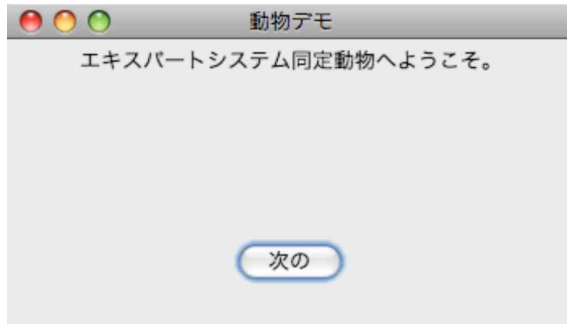
10.7 INTERNATIONALIZING THE SWING DEMO PROGRAMS

The Swing Demo Programs have been designed for internationalization. Several software generated example translations have been provided including Japanese (language code ja), Russian (language code ru), Spanish (language code es), and Arabic (language code ar). To make use of one of the translations, specify the language code when starting the demonstration

program. For example, to run the Animal Demo in Japanese on Mac OS X, use the following command:

```
java -cp ../../CLIPSJNI.jar -Djava.library.path=../../ -Duser.language=ja AnimalDemo
```

The welcome screen for the program should appear in Japanese rather than English:



It may be necessary to install additional fonts to view some languages. On Mac OS X, you can see which languages are supported by launching System Preferences and selecting the Language tab from the International preference category. On Windows XP, you can see which languages are supported by launching Control Panel and selecting the Languages tab from Regional and Language Options.

To create translations for other languages, first determine the two character language code for the target language. Make a copy in the resources directory of the ASCII English properties file for the demo program and save it as a UTF-8 encoded file including the language code in the name and using the .source extension. A list of language code is available at <http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt>. For example, to create a Greek translation file for the Wine Demo, create the UTF-8 encoded WineResources_el.source file from the ASCII WineResources.properties file. Note that this step requires that you to do more than just duplicate the property file and rename it. You need to use a text editor that allows you to change the encoding from ASCII to UTF-8.

Once you've created the translation source file, edit the values for the properties keys and replaced the English text following each = symbol with the appropriate translation. When you have completed the translation, use the Java native2ascii utility to create an ASCII text file from the source file. For example, to create a Greek translation for the Wine Demo program, you'd use the following command:

```
native2ascii -encoding UTF-8 WineResources_el.source WineResources_el.properties
```

Note that the properties file for languages containing non-ASCII characters will contain Unicode escape sequences and is therefore more difficult to read (assuming of course that you can read the language in the original source file). This is the reason that two files are used for creating the translation. The UTF-8 source file is encoded so that you can read and edit the translation and the

ASCII properties file is encoded in the format expected for use with Java internationalization features.