Section 10:
CLIPS Java Native Interface

This section describes the CLIPS Java Native Inteface (CLIPSJNI) and the examples demonstrating the integration of CLIPS with a Swing interface. The examples have been tested using Java version 1.8.0_31 running on Mac OS X 10.10.2 and Windows 7 and Java version 1.8.0_45 running on Ubuntu 14.04 LTS.

10.1 CLIPSJNI Directory Structure

In order to use CLIPSJNI, you must obtain the source code by downloading the CLIPSJNI zip file from the Files page on the CLIPS SourceForge web page (see appendix A for the SourceForge URL). When unzipped the CLIPSJNI project file contains the following directory structure:

```
CLIPSJNI
  bin
    animal
    auto
    clipsjni
    sudoku
    wine
  java-src
    net
      sf
        clipsrules
          jni
            examples
              animal
                resources
              auto
                resources
              sudoku
                resources
              wine
                resources
  library-src
```

If you are using the CLIPSJNI on Mac OS X, then the native CLIPS library is already contained in the top level CLIPSJNI directory.

On Windows, it is necessary to verify that the correct DLL is installed. By default, the DLL for 64-bit Windows is used as the CLIPSJNI.dll file in the top level of the CLIPSJNI directory. If running CLIPSJNI with 32-bit Windows, delete the existing CLIPSJNI.dll file, then make a copy of the CLIPSJNI32.dll file and rename it to CLIPSJNI.dll.

On other systems, you must created 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 Issuing Commands from the Terminal

As packaged, invoking and compiling various CLIPSJNI components requires that you enter commands from a terminal application.

On Windows 7, launch the Command Prompt application (select Start > All Programs > Accessories > Command Prompt).

On Mac OS X, launch the Terminal application (located in the Applications/Utilities directory).

On Ubuntu, launch the Terminal application (either by double clicking gnome-terminal in /usr/bin or clicking the Home button, searching for Terminal, then clicking the Terminal application).

Once the terminal has been launched, set the directory to the CLIPSJNI top level directory (using the cd command). Unless otherwise noted, all commands should be entered while in the CLIPSJNI directory.

10.3 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 and Mac OS X, enter the following command from the CLIPSJNI directory:

```
java -jar CLIPSJNI.jar
```

On Ubuntu, you must first create the CLIPSJNI native library (see section 10.6.3). Once created, enter the following command from the CLIPSJNI directory:

```
java -Djava.library.path=. -jar CLIPSJNI.jar
```

The CLIPS banner and command prompt should appear:

```
         CLIPS (6.31 5/19/15)
CLIPS>
```

10.4 Running the Swing Demo Programs

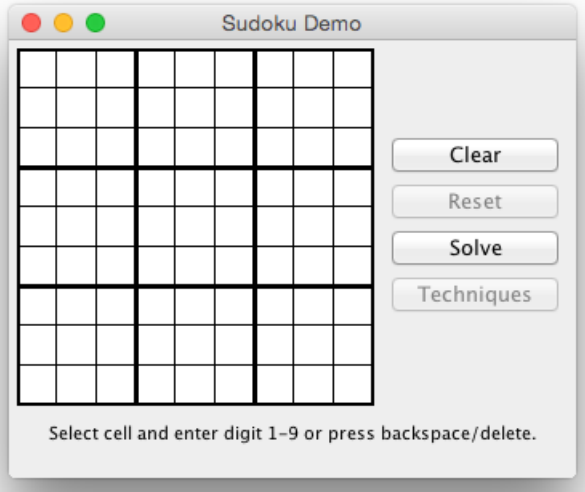
The Swing CLIPSJNI demonstration programs can be run on Windows 7 or Mac OS X using the precompiled native libraries in the CLIPSJNI top level directory. On Ubuntu and other systems, a native library must first be created before the programs can be run.

10.4.1 Running the Demo Programs on Mac OS X

To run the Sudoku demo, enter the following command:

```
java -jar SudokuDemo.jar
```

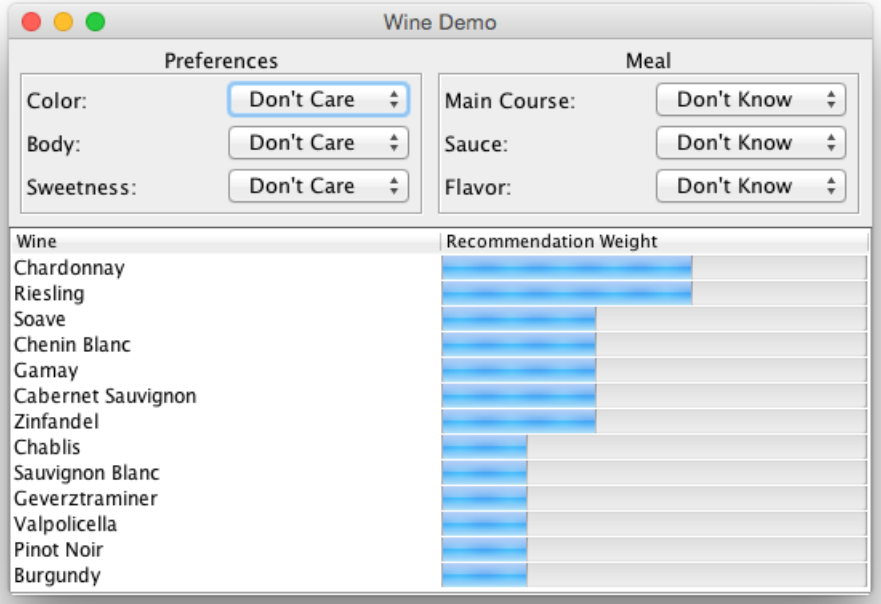
The Sudoku Demo window should appear:



To run the Wine demo, enter the following command:

```
java -jar WineDemo.jar
```

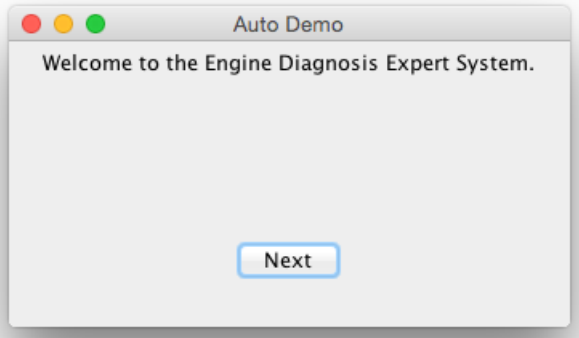
The Wine Demo window should appear:



To run the Auto demo, enter the following command:

```
java -jar AutoDemo.jar
```

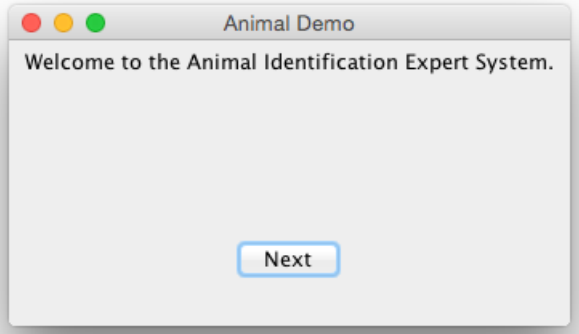
The Auto Demo window should appear:



To run the Animal demo, enter the following command:

```
java -jar AnimalDemo.jar
```

The Animal Demo window should appear:

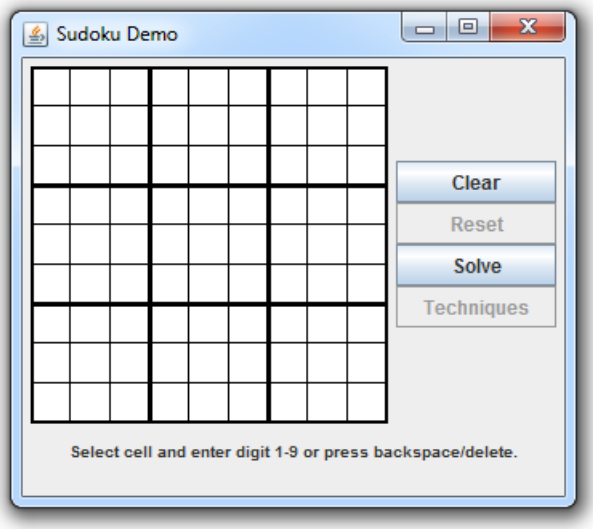


10.4.2 Running the Demo Programs on Windows 7

To run the Sudoku demo, enter the following command:

```
java –jar SudokuDemo.jar
```

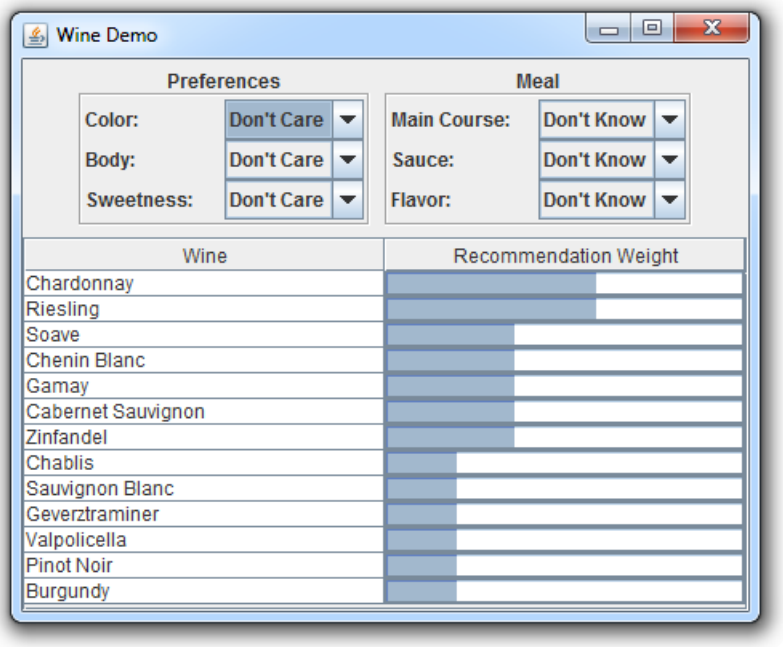
The Sudoku Demo window should appear:



To run the Wine demo, enter the following command:

```
java -jar WineDemo.jar
```

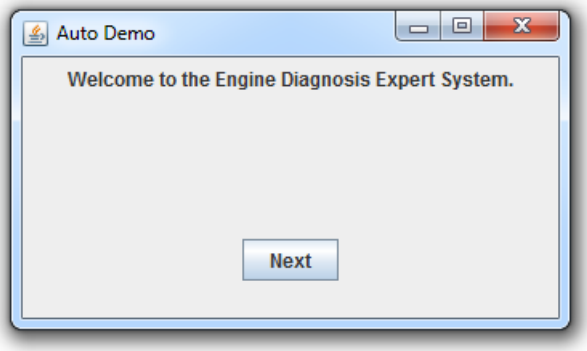
The Wine Demo window should appear:



To run the Auto demo, enter the following command:

```
java -jar AutoDemo.jar
```

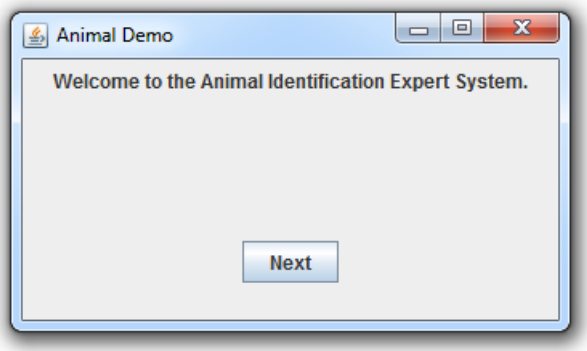
The Auto Demo window should appear:



To run the Animal demo, enter the following command:

```
java -jar AnimalDemo.jar
```

The Animal Demo window should appear:



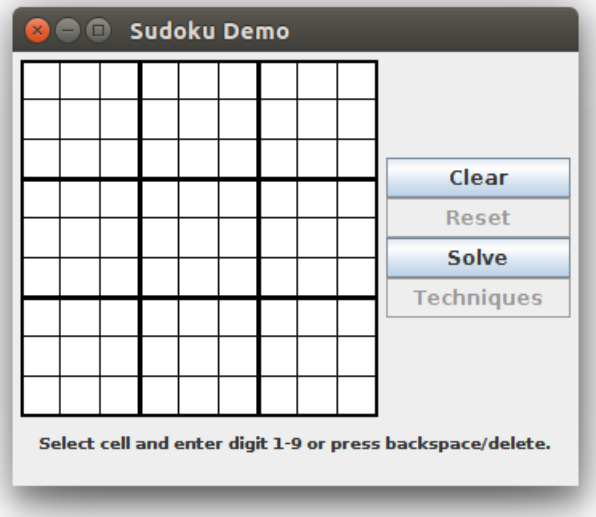
10.4.3 Running the Demo Programs on Ubuntu

To run the demos on Ubuntu, you must first create the CLIPSJNI native library (see section 10.6.3).

To run the Sudoku demo, enter the following command:

```
java -Djava.library.path=. -jar SudokuDemo.jar
```

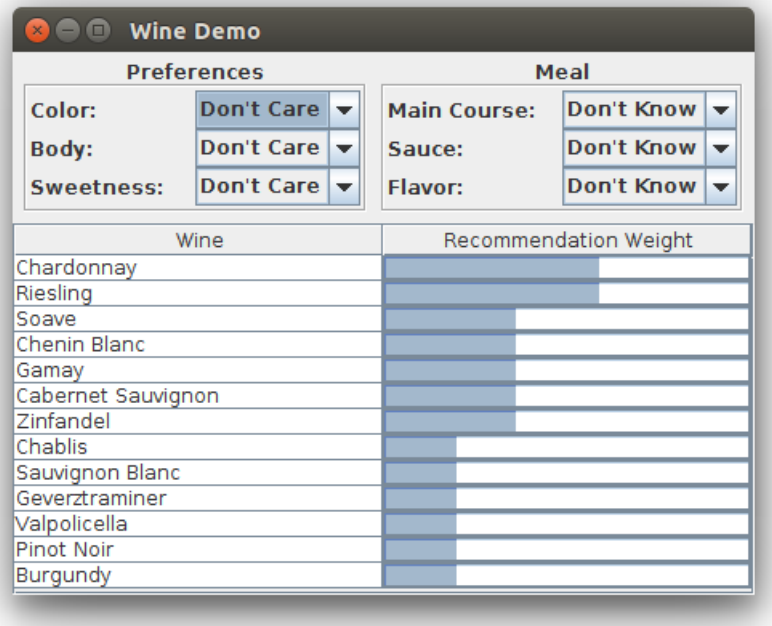
The Sudoku Demo window should appear:



To run the Wine demo, enter the following command:

```
java -Djava.library.path=. -jar WineDemo.jar
```

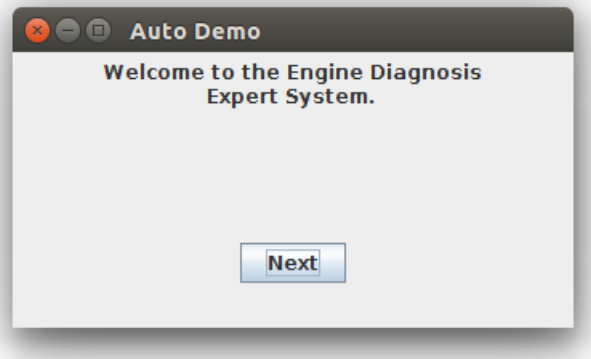
The Wine Demo window should appear:



To run the Auto demo, enter the following command:

```
java -Djava.library.path=. -jar AutoDemo.jar
```

The Auto Demo window should appear:



To run the Animal demo, enter the following command:

```
java -Djava.library.path=. -jar AnimalDemo.jar
```

The Animal Demo window should appear:

10.5 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 1.8.0).

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 in the CLIPSJNI directory, enter the following command:

```
javah -d library-src -classpath java-src -jni net.sf.clipsrules.jni.Environment
```

This command creates a file named net_sf_clipsrules_jni_Environment.h and places it in the CLIPSJNI/library-src directory.

10.5.1 Creating the CLIPSJNI JAR on Mac OS X

Enter the following command to compile the CLIPSJNI java source and generate the JAR file:

```
make -f makefile.mac clipsjni
```

10.5.2 Creating the CLIPSJNI JAR on Windows

Enter the following command to compile the CLIPSJNI java source and generate the JAR file:

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

10.5.3 Creating the CLIPSJNI JAR on Ubuntu

Enter the following command to compile the CLIPSJNI java source and generate the JAR file:

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

10.6 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.

10.6.1 Creating the Native Library on Mac OS X

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

To create a universal native library that can run on both Intel 32 and 64 bit architectures, enter the following command:

```
make -f makefile.mac
```

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

10.6.2 Creating the Native Library on Windows 7

The following steps assume you have Microsoft Visual Studio 2013 installed. First, launch the Command Prompt application (select Start > All Programs > Accessories > Command Prompt).

Next, execute the script that sets up the environment variables for the appropriate target machine. For example, the vcvars64.bat batch file in the directory "Program Files (x86)/Microsoft Visual Studio 12.0/VC/bin/amd64".

Set the directory to the CLIPSJNI/lbrary-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.

10.6.3 Creating the Native Library On Ubuntu

Launch the Terminal application (either by double clicking gnome-terminal in /usr/bin or clicking the Home button, searching for Terminal, then clicking the Terminal application). Set the directory to the CLIPSJNI/lbrary-src directory (using the cd command).

To create a native library, enter the following command:

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

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

10.7 Recompiling the Swing Demo Programs

If you want to make modification to the Swing Demo programs, you can recompile them using the makefiles in the CLIPSJNI directory.

10.7.1 Recompiling the Swing Demo Programs on Mac OS X

Use these commands to recompile the examples:

```
make –f makefile.mac sudoku

make –f makefile.mac wine

make –f makefile.mac auto

make –f makefile.mac animal
```

10.7.2 Recompiling the Swing Demo Programs on Windows

Use these commands to recompile the examples:

```
nmake –f makefile.win sudoku

nmake –f makefile.win wine

nmake –f makefile.win auto

nmake –f makefile.win animal
```

10.7.3 Recompiling the Swing Demo Programs on Ubuntu

Use these commands to recompile the examples:

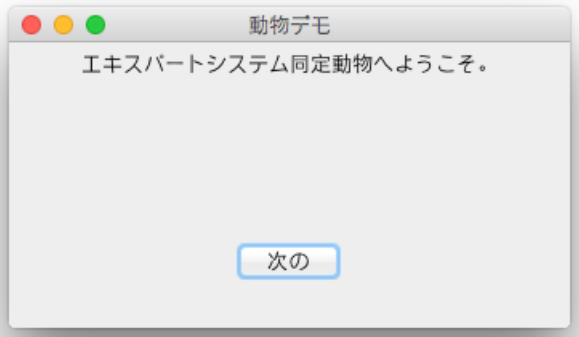
```
make –f makefile.linux sudoku

make –f makefile.linux wine

make –f makefile.linux auto

make –f makefile.linux animal
```

10.8 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). The Sudoku and Wine demos make use of translations just for the Swing Interface. The Auto and Animal demos also demonstrate the use of translation text from within CLIPS. 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 -jar -Duser.language=ja AnimalDemo.jar
```

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 clicking the Language & Region icon. On Windows 7, you can see which languages are supported by launching Control Panel and selecting the Keyboards and Languages tab from Region 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://www.mathguide.de/info/tools/languagecode.html. 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.

The CLIPS translation files stored in the resource directory (such as animal_es.clp) can be duplicated and edited to support new languages. The base name of each new file should end with the appropriate two letter language code. There is no need to convert these UTF-8 files to another format as CLIPS can read these directly.