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 with the following software environments:

- Windows 10 with JDK 1.8.0_211 and Visual Studio Community 2019
- MacOS 10.14 with JDK 1.8.0_211 and Xcode 10.2
- Linux: Ubuntu 16.04 LTS with OpenJDK 1.8.0_212, Debian 9.1 with OpenJDK 1.8.0_141, Fedora 26 with OpenJDK 1.8.0_141, CentOS 7 with OpenJDK 1.8.0_141, and Mint 18 with OpenJDK 1.8.0_131

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). Once downloaded, you must then extract the contents of the file by right clicking on it and selecting the “Extract All…” menu item.

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 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 10, to run the precompiled Java applications, launch the Command Prompt application (select Start > Windows System > Command Prompt). To recompile the native library or use the provided makefiles to rebuild the Java source code, you must have Visual Studio installed. In this case, launch the Command Prompt application by selecting Start > Visual Studio 2019 > Developer Command Prompt for VS 2019. Using the *Developer Command Prompt for VS 2019* application sets the appropriate paths to use the Visual Studio compiler and make tools. Alternately *x86 Native Tools Command Prompt for VS 2019* or *x64 Native Tools Command Prompt for VS 2019* can be used to compile a specific processor architecture.

On macOS, click the Spotlight icon in the menu bar, enter ‘Terminal’ in the search field, and then double click on Terminal.app in the search results to launch the application.

On Ubuntu, click on the “Search your computer” icon, enter ‘Terminal’ in the search field, and then click on Terminal in the search results to launch the application.

On Fedora and Debian, click on Activities in the menu bar, click the Show Applications icon, enter ‘Terminal’ in the search field, and then click on Terminal in the search results to launch the application.

On CentOS, click on Applications in the menu bar, click on Activities Overview, click the Show Applications icon, enter ‘Terminal’ in the search field, and then click on Terminal in the search results to launch the application.

On Mint, click on Menu in the lower toolbar, enter ‘Terminal’ in the search field, and then click Terminal in the search results to launch the 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 macOS, enter the following command from the CLIPSJNI directory:

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

On Linux, 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 6/12/19)
CLIPS>
```

10.4 Running the Swing Demo Programs

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

10.4.1 Sudoku Demo

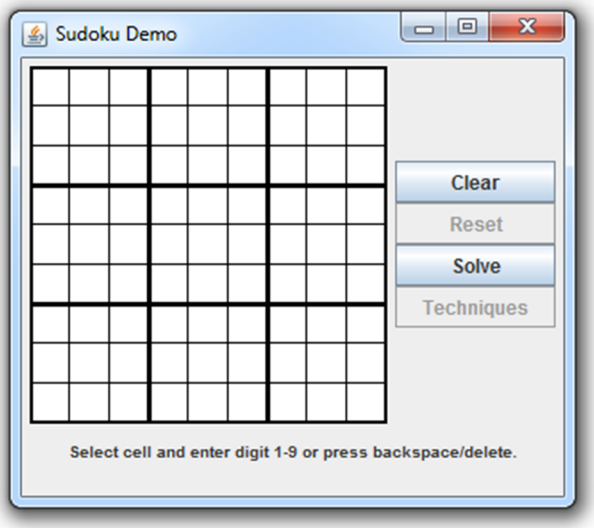
To run the Sudoku demo on Windows or macOS, enter the following command:

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

To run the Sudoku demo on Linux, enter the following command:

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

The Sudoku Demo window should appear (Windows 10 pictured):



10.4.2 Wine Demo

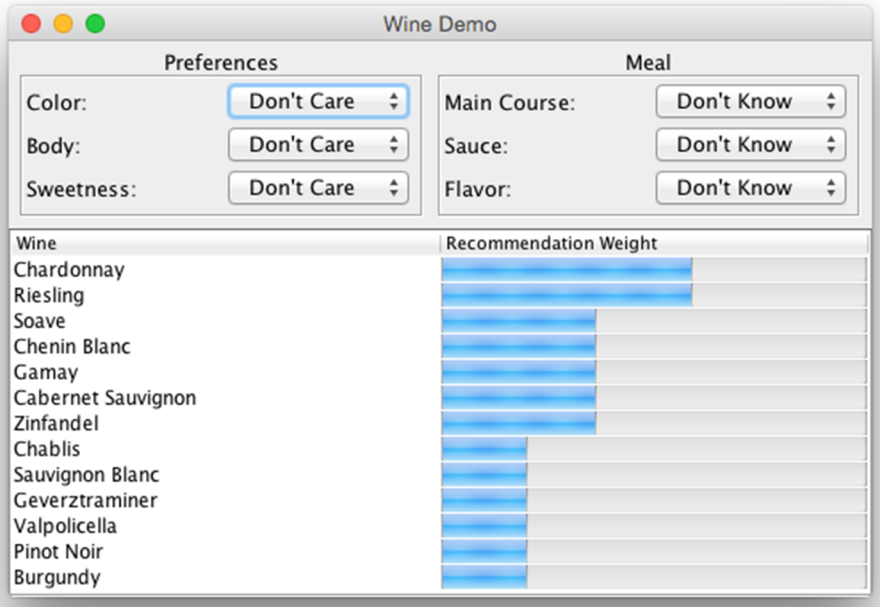
To run the Wine demo on Windows or macOS, enter the following command:

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

To run the Wine demo on Linux, enter the following command:

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

The Wine Demo window should appear (macOS pictured):



10.4.3 Auto Demo

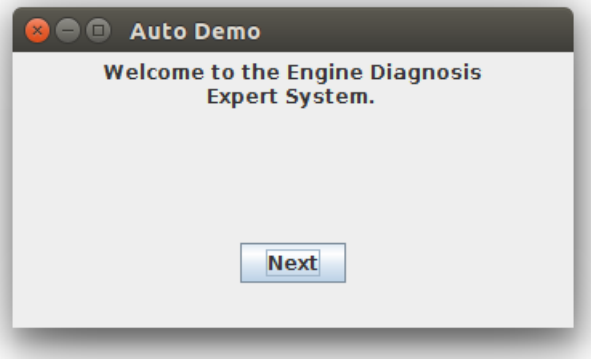
To run the Auto demo on Windows or macOS, enter the following command:

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

To run the Auto demo on Linux, enter the following command:

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

The Auto Demo window should appear (Ubuntu pictured):



10.4.4 Animal Demo

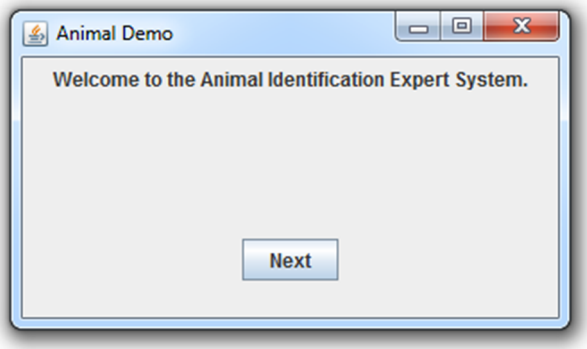
To run the Animal demo on Windows or macOS, enter the following command:

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

To run the Animal demo on Linux, enter the following command:

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

The Animal Demo window should appear (Windows 10 pictured):



10.5 Creating the CLIPSJNI JAR File

If you wish to add new functionality to the CLIPSJNI package, 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 that is 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.

On macOS, enter the following command to compile the CLIPSJNI java source and generate the JAR file:

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

On Windows 10, enter the following command to compile the CLIPSJNI java source and generate the JAR file:

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

On Linux, enter the following command to compile the CLIPSJNI java source and generate the JAR file:

```
make -f makefile.lnx clipsjni
```

10.6 Creating the CLIPSJNI Native Library

The CLIPSJNI distribution already contains a precompiled universal library for macOS, 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 macOS 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 macOS

Launch the Terminal application (as described in section 10.2). 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.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 10

Launch the Terminal application (as described in section 10.2). 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 Linux

Launch the Terminal application (as described in section 10.2). Set the directory to the CLIPSJNI/lbrary-src directory (using the cd command).

To create a native library, enter the following command (where <distribution> is either ubuntu, fedora, debian, mint, or centos):

```
make -f makefile.lnx <distribution>
```

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 macOS

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 Linux

Use these commands to recompile the examples:

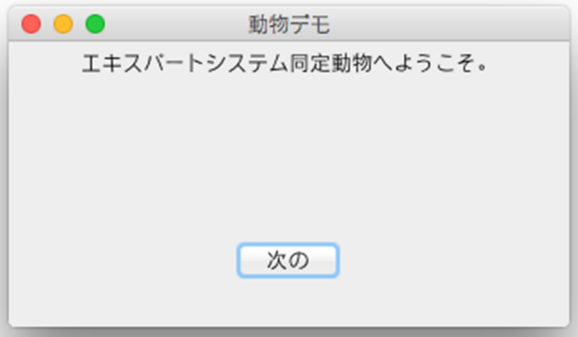
```
make –f makefile.lnx sudoku
make –f makefile.lnx wine
make –f makefile.lnx auto
make –f makefile.lnx animal
```

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 -Duser.language=ja -jar 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 macOS, you can see which languages are supported by launching 'System Preferences' and clicking the 'Language & Region' icon. On Windows 10, you can see which languages are supported by launching Settings, selecting 'Time and language,' and then selecting 'Region and language.'

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.