Assignment No. 5 (Now called as 14): Write a program to create Dynamic Link Library for mathematical Operations in C or C++ and write a test program in Java to test those operations

Step 1: Writing the Java code for JNI.

We start from the Java code. Please refer to TestJNI.java.

I have defined the following native methods in my Java code:

**public** **native** **int** add(**int** n1,**int** n2) ;

**public** **native** **int** sub(**int** n1, **int** n2);

**public** **native** **int** mul(**int** n1, **int** n2);

**public** **native** **int** div(**int** n1, **int** n2);

**public** **native** **double** sin(**double** n1);

**public** **native** **double** cos(**double** n1);

**public** **native** **double** log10(**double** n1);

The main method is menu-driven to accept any of the options of the mathematical operations.

E.g. 1 = add, 2 = subtract etc.

The switch-case statement takes care of all 7 operations.

The most important call is: **static** {

System.loadLibrary ( "TestJNI" ) ;

}

This loads the native library TestJNI.dll in memory “statically”. i.e. only one instance of the library will be loaded per class (in this case “TestJNI” class).

Step 2:

Next, Compile the Java code from command line. Make sure you have JDK 1.8 installed and you have set the PATH and JAVA\_HOME environment variables correctly.

In a cmd session on Windows, run the following command:

javac TestJNI.java

This will create a class file “TestJNI.class” in the local folder.

Now, we use the “javah” command to create the header file for native library.

javah –jni TestJNI

The resulting file (TestJNI.h) contains declarations of all the mathematical functions in specific format required by JNI.

DO NOT EDIT THIS FILE BY HAND.

Step 3:

Now, we will compile the TestJNI.c file using gcc. Take a look at TestJNI.c and understand how each function is written.

First and foremost, we include the following header files in this file:

#include <jni.h>

#include "TestJNI.h”

#include <stdio.h>

#include <Math.h>

Also, each function signature contains the following: JNIEXPORT jint JNICALL (or JNIEXPORT jdouble JNICALL in cases where the function returns a “double” value.) Also, each function has “JNIEnv \*env” and “ jobject object” as first two mandatory parameters. Please read more about JNI data types at the following location:

<https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/types.html>

We compile this C file using the following command on windows:

gcc -I%JAVA\_HOME%\include\ -I%JAVA\_HOME%\include\win32 -o TestJNI.dll -shared -fPIC TestJNI.c

We compile the C file using the following command on Ubuntu:

gcc -I$JAVA\_HOME/include/ -I$JAVA\_HOME/include/linux -o libTestJNI.so -shared -fPIC TestJNI.c

Note:

1. You can download the 32 and 64 bit version of gcc for Windows from the following location:

<https://sourceforge.net/projects/tdm-gcc/>

1. The gcc command can be used as it is on Windows or Ubuntu, provided your JAVA\_HOME is set correctly.

Successful execution of this command results in the creation of TestJNI.dll on Windows or libTestJNI.so on Ubuntu.

Check your local folder for the same.

So, at the end of the above three steps, you should see the following files in your folder:

TestJNI.class

TestJNI.h

TestJNI.dll or libTestJNI.so

These are apart from the source files:

TestJNI.java

TestJNI.c

Step 4:

To test the Java program, run the class file as follows:

java TestJNI

You will see the following menu options:

Menu:

1.Add

2. sub

3.Mul

4.Div

5.sin

6.cos

7.Log10

Try each option and provide appropriate input parameters from command line.

You should see the results of each mathematical operation.

That’s it!

Now you are a pro in writing JNI code! ☺