Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

Project CASA: Running C code from JAVA.

One of the approaches we have considered while rewriting CASA is running parts of the c++ and c code from java. Essentially calling methods and initializing objects defined in c, but from a java program. The reasons for this are numerous. First of all, it would help to test the rewritten code. Say, we rewrite three classes out of o module containing ten and we want to test them, this way we can. Furthermore, there are parts of CASA that we don't necessarily have to rewrite, but that would prove useful for purposes of development. Here the possibility of integrating C code into Java would also prove useful and save time. Lastly as we are after all trying to maintain the CASA programs speed and effectiveness, it may prove difficult to accomplish this with merely java, and so it could be useful to have some alternative options, such as running some highly CPU unfriendly algorithms in C instead.

There are basically 3 ways to accomplish this: Java Native Interface (from now only JNI), Java Native Access (JNA) or run parts of the C program by accessing the command line via java, entering input a reading the output. The third option we outright reject, seeing as is would require us to make large adjustments to the CASA code so that individual fragments could be executed separately. It is also generally impractical. The choice really came down to JNI and JNA. Unfortunately JNA does not support mapping of c++ classes and according to our research JNI also offers a better performance. So in the end we agreed to use JNI.

Here we include an example of how it should work (this is more for our benefit, than the stakeholders). We create a java class:

```
public class HelloJNI {
  static {
    System.loadLibrary("hello");  }
  private native void sayHello();
  public static void main(String[] args) {
    new HelloJNI().sayHello();
  }
}
```

Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

Here we load the native library mapped to hello.dll (libhello.so in Unix) containing the sayHello method. Furthermore we would have such a header:

```
#include <jni.h>
#ifndef _Included_HelloJNI
#define _Included_HelloJNI
#ifdef __cplusplus
extern "C" {
#endif

JNIEXPORT void JNICALL Java_HelloJNI_sayHello(JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif
```

The header declares the C function witch we than implement in the HelloJNI.c file. like so:

```
#include <jni.h>
#include <stdio.h>
#include "HelloJNI.h"

// Implementation of native method sayHello() of HelloJNI class
JNIEXPORT void JNICALL Java_HelloJNI_sayHello(JNIEnv *env, jobject thisObj) {
  printf("Hello World!\n");
  return;
}
```

We complile the C program (for example using: gcc -Wl,--add-stdcall-alias -I"C:\Program Files\Java\jdk1.8.0_73\include" -I"C:\Program Files\Java\jdk1.8.0_73\include\win32" -shared -o hello.dll HelloJNI.c) To create the hello.dll, that can be then included as a library and used in java.
It should be noted, that this is a simplification and more comprehensive documentation can be found at https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaNativeInterface.html.
We are currently trying to use this approach and run parts of CASA from java. We will report regularly on our progress.

Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

There are a few questions for the stakeholder:

- *Is this approach worth pursuing? Do you think it could prove helpful when putting the java version of the CASA program together?*
- *Is it possible to merely run parts of the CASA code in C or will you require the entire code rewritten and usable with java alone.*
- *If we should use such an approach to supplement some parts of the CASA program, witch? Is there for example a particular module witch you don't need rewritten at all, but could use in the java implementation?*