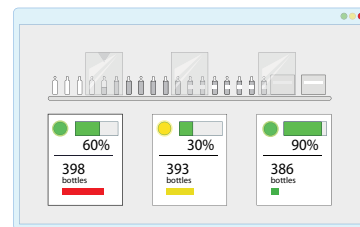


HOW TO  
**SAVE** MONEY & TIME  
WITH



**JNIWrapper**




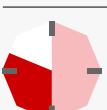
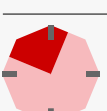
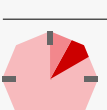
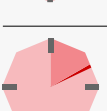
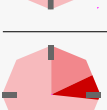


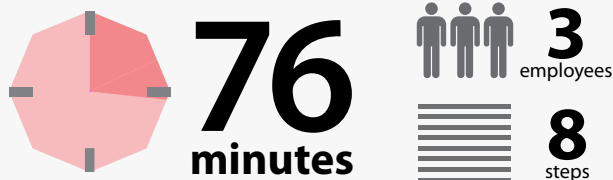
Suppose that there is an existing application, written in C, that monitors conditions on a factory floor, and you need to incorporate this functionality into a Java application.

The traditional approach using JNI involves Java and C developers as well as management coordination to keep the functionality synchronized.




By contrast, using JNIWrapper is trivial and saves time and efforts dramatically.

# Traditional Solution

	<b>Java and C developers</b> Declare Java class with required native methods	<pre>class ReadFile {     native byte[] loadFile(String name);     /* Native method declaration*/ } </pre>
	<b>Java developer</b> Compile this class using Javac compiler (from JDK)	<pre>javac ReadFile.java </pre>
	<b>C developer</b> Make a C header file from this class (from JDK)	<pre>javah ReadFile </pre>
	<b>C developer</b> Implementation of this C header by a C library file	<pre>JNIEXPORT jbyteArray JNICALL Java_ ReadFile_loadFile (JNIEnv* env, jobject jobj, jstring name) { /*implementation of native part*/} </pre>
	<b>C developer and manager</b> Document the interfaces and create a process to make sure they stay in synch	
	<b>C developer</b> Compilation of the library project	
	<b>Java developer</b> Load this native library in a Java application	<pre>System.loadLibrary("nativeLib"); </pre>
	<b>Java developer</b> Invoke native method in a Java application	<pre>loadFile("FileName"); </pre>



# Using JNIWrapper

	<b>Java developer</b> Load native library	<pre>Library nativeLib = new Library("libraryName"); </pre>
	<b>Java developer</b> Get the required function from a native library	<pre>Function function = nativeLib.getFunction("functionName"); </pre>
	<b>Java developer</b> Invoke the function using the required input parameters	<pre>function.invoke(...); </pre>

