

Compiling and Running Programs Using JNI

LEGENDS:

- Shared library is <slibrary>
- Filename is <fn>. Remember: The .java and .c files should have the same name.

STEPS:

1) Create <fn>.java and <fn>.c files. The files are given below for reference.

<fn>.java:

```
public class <fn>
{
    static
    {
        System.loadLibrary("<slibrary>");
    }

    public native void <func-name>();

    public static void main(String[] args)
    {
        try
        {
            <fn> test=new <fn>();
            test.<func-name>();
        }
        catch(Exception e)
        {
            System.out.println("Alert:"+e.getMessage());
        }
    }
}
```

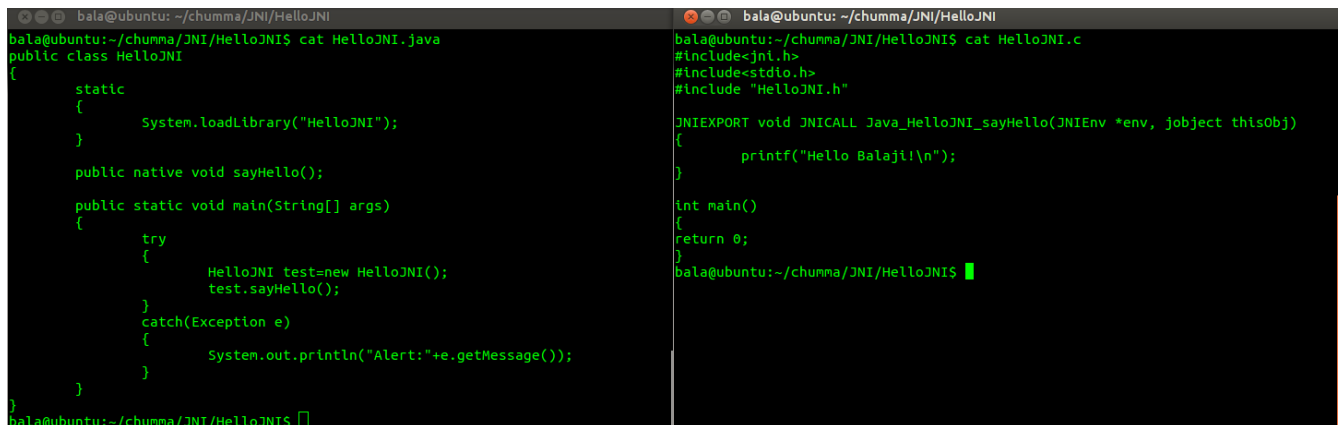
```
    }  
    }  
}
```

<fn>.c:

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

```
JNIEXPORT void JNICALL Java_<fn>_<function-name>(JNIEnv *env, jobject thisObj)  
{  
    printf("Hello world!\n");  
}
```

```
int main()  
{  
    return 0;  
}
```

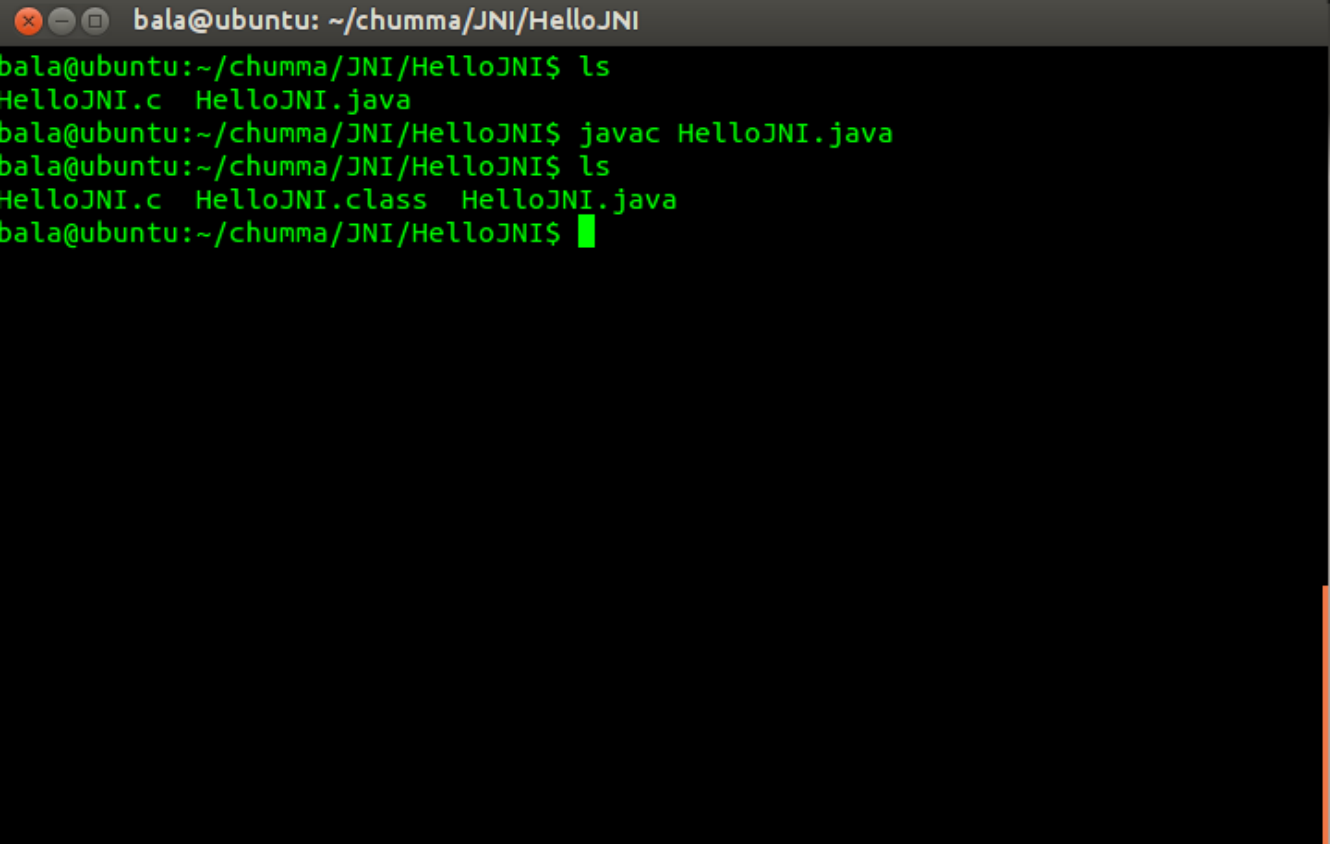


```
bala@ubuntu:~/chumma/JNI/HelloJNI  
bala@ubuntu:~/chumma/JNI/HelloJNI$ cat HelloJNI.java  
public class HelloJNI  
{  
    static  
    {  
        System.loadLibrary("HelloJNI");  
    }  
    public native void sayHello();  
    public static void main(String[] args)  
    {  
        try  
        {  
            HelloJNI test=new HelloJNI();  
            test.sayHello();  
        }  
        catch(Exception e)  
        {  
            System.out.println("Alert:"+e.getMessage());  
        }  
    }  
}  
bala@ubuntu:~/chumma/JNI/HelloJNI$  
  
bala@ubuntu:~/chumma/JNI/HelloJNI$ cat HelloJNI.c  
#include<jni.h>  
#include<stdio.h>  
#include "HelloJNI.h"  
  
JNIEXPORT void JNICALL Java_HelloJNI_sayHello(JNIEnv *env, jobject thisObj)  
{  
    printf("Hello Balaji!\n");  
}  
  
int main()  
{  
    return 0;  
}  
bala@ubuntu:~/chumma/JNI/HelloJNI$
```

2) Open the Terminal. Compile the java file by typing the following command in Terminal:

```
javac <fn>.java
```

Executing this command creates a <fn>.class file in the directory.

A terminal window titled 'bala@ubuntu: ~/chumma/JNI/HelloJNI' with a black background and green text. The terminal shows the following sequence of commands and outputs:

```
bala@ubuntu:~/chumma/JNI/HelloJNI$ ls
HelloJNI.c  HelloJNI.java
bala@ubuntu:~/chumma/JNI/HelloJNI$ javac HelloJNI.java
bala@ubuntu:~/chumma/JNI/HelloJNI$ ls
HelloJNI.c  HelloJNI.class  HelloJNI.java
bala@ubuntu:~/chumma/JNI/HelloJNI$ █
```

3) Create the header file from the class file generated in the above step by typing the following command in the Terminal:

```
javah <fn>
```

```
bala@ubuntu: ~/chumma/JNI/HelloJNI
bala@ubuntu:~/chumma/JNI/HelloJNI$ javah HelloJNI
bala@ubuntu:~/chumma/JNI/HelloJNI$ ls
HelloJNI.c HelloJNI.class HelloJNI.h HelloJNI.java
bala@ubuntu:~/chumma/JNI/HelloJNI$ █
```

4) Compile only by typing in the following command:

```
gcc -c -fPIC -I"/usr/local/java/jdk1.8.0_05/include" -
I"/usr/local/java/jdk1.8.0_05/include/linux" <fn>.c
```

-c option : only compile the source file to object file

-fPIC option : Generate position-independent code (PIC) suitable for use in a shared library, if supported for the target machine.

-I option : Instructs the compiler to search the directories following it for the header files included in the C program

```
bala@ubuntu: ~/chumma/JNI/HelloJNI
bala@ubuntu:~/chumma/JNI/HelloJNI$ gcc -c -fPIC -I"/usr/local/java/jdk1.8.0_05/i
nclude" -I"/usr/local/java/jdk1.8.0_05/include/linux" HelloJNI.c
bala@ubuntu:~/chumma/JNI/HelloJNI$ █
```

5) Create shared library:

```
gcc -I"/usr/local/java/jdk1.8.0_05/include" -I"/usr/local/java/jdk1.8.0_05/include/linux" -
shared -o lib<slibrary>.so <fn>.o
```

-shared option : Instructs the compiler to create a shared library file

-o option : Instructs the compiler to name the shared library by the parameter following it, in this case <slibrary>.so

The file lib<slibrary>.so is the shared library file created by the above command.

```
bala@ubuntu: ~/chumma/JNI/HelloJNI
bala@ubuntu:~/chumma/JNI/HelloJNI$ gcc -I"/usr/local/java/jdk1.8.0_05/include" -I"/usr/local/java/jdk1.8.0_05/include/linux" -shared -o libHelloJNI.so HelloJNI.o
bala@ubuntu:~/chumma/JNI/HelloJNI$ ls
HelloJNI.c      HelloJNI.h      HelloJNI.o
HelloJNI.class HelloJNI.java   libHelloJNI.so
bala@ubuntu:~/chumma/JNI/HelloJNI$ █
```

6) Run the java program:

```
java -Djava.library.path=. <slibrary>
```

bala@ubuntu: ~/chumma/JNI/HelloJNI

```
bala@ubuntu:~/chumma/JNI/HelloJNI$ java -Djava.library.path=. HelloJNI
```

```
Hello Balaji!
```

```
bala@ubuntu:~/chumma/JNI/HelloJNI$ █
```