Draft project proposal

Please submit a draft of your project proposal (worth 2%) before Wednesday February 21 by

- transferring the file to `red.eecs.yorku.ca` and
- submitting the file using

```
submit 4315 draft <name of file>
```

Peers and Native Peers

EECS 4315

www.eecs.yorku.ca/course/4315/

JPF

```
public class Sine {
  public static void main(String[] args) {
    System.out.println(StrictMath.sin(0.3));
  }
}
```

Applying JPF

Question

Why does JPF report the following error?

```
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
java.lang.UnsatisfiedLinkError: cannot find
native java.lang.StrictMath.sin
at java.lang.StrictMath.sin(no peer)
at Sinus.main(Sine.java:3)
```

Applying JPF

Question

Why does JPF report the following error?

```
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
java.lang.UnsatisfiedLinkError: cannot find
native java.lang.StrictMath.sin
at java.lang.StrictMath.sin(no peer)
at Sinus.main(Sine.java:3)
```

Answer

Because the sin method is native.

```
public static native double sin(double a);
```

Native methods

Question

What is a native method?

Native methods

Question

What is a native method?

Answer

A method that is implemented in a language other than Java but that is invoked from a Java app.

Native methods

Question

Why are there native methods?

Native methods

Question

Why are there native methods?

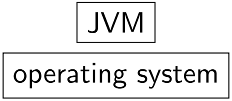
Answer

- Allows programmers to use code that has already been implemented in other languages.
- May increase the performance.
- May support certain platform-dependent features.

Many of the classes of the Java standard library include native methods.

Java native interface (JNI)

JNI provides the infrastructure for Java code to use libraries written in other languages such as C, C++ and assembly.



Invoking a native method can be viewed as transferring the execution from the JVM to the operating system, since the native code will be executed outside the JVM and will run on the operating system.

Sheng Liang. *Java Native Interface: Programmer's Guide and Specification*. Prentice Hall. 1999.

Handling native methods

JPF provides several ways to handle native methods.

- Using peers (also known as model classes).
- Using native peers.
- Using a combination of peers and native peers.
- Using the extension jpf-nhandler.

Peers

A peer class captures the behaviour of a native method in pure Java.

Question

How can we capture the behaviour of the sin method?

Peers

A peer class captures the behaviour of a native method in pure Java.

Question

How can we capture the behaviour of the sin method?

Answer

For example, we approximate the sine function with the Bhaskara I's sine approximation formula:

$$\sin(a) = \frac{16a(\pi - a)}{5\pi^2 - 4a(\pi - a)}$$

Peers

A peer class captures the behaviour of a native method in pure Java.

Question

How can we capture the behaviour of the sin method?

Peers

A peer class captures the behaviour of a native method in pure Java.

Question

How can we capture the behaviour of the sin method?

Answer

For example, we can use `Math.sin`.

Peer class

```
package java.lang;

public class StrictMath {
  public static double sin(double a) {
    return 16 * a * (Math.PI - a) /
      (5 * Math.PI * Math.PI - 4 * a * (Math.PI - a));
  }
}
```

Peer class

```
package java.lang;

public class StrictMath {
  public static double sin(double a) {
    return Math.sin(a);
  }
}
```

Peer class

- The peer class `StrictMath` is part of the package `java.lang`.
- The peer class only contains one method, whereas the original `StrictMath` class contains many more.

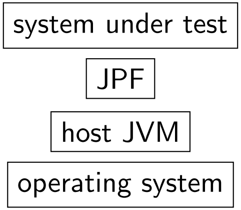
Peer classes

To ensure that JPF verifies the peer class, rather than the original class, we need to add the peer class to JPFs classpath.

```
target=Sine
classpath=<folder containing StrictMath.class>
```

Native peers

Native peers are the JPF analogue of native code.



Native peers are executed by the host JVM (recall that peers are executed by JPF).

Model Java interface (MJI)

MJI is the JPF analogue of JNI. It provides the infrastructure for interaction between JPF and the host JVM when executing a native peer.

Naming

MJI uses a specific name pattern to establish the correspondence between the original class (containing the native method) and its native peer, similar to JNI.

```
package java.lang;

public class StrictMath
```

corresponds to

```
public class JPF_java_lang_StrictMath
  extends NativePeer
```

Naming

Question

What is the MJI counterpart of

```
package java.lang;

public class Boolean
```

Naming

Question

What is the MJI counterpart of

```
package java.lang;

public class Boolean
```

Answer

```
public class JPF_java_lang_Boolean
  extends NativePeer
```

Naming

MJI uses a specific name pattern to establish the correspondence between the original class (containing the native method) and its native peer, similar to JNI.

https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/design.html

```
public static native double sin(double a);
```

corresponds to

```
@MJI
public double sin__D__D(MJIEnv env,
  int clsObjRef, double a)
```

Naming

boolean	Z
byte	B
char	C
short	S
int	I
long	L
float	F
double	D

Naming

Question

What is the MJI counterpart of

```
public static Boolean valueOf(boolean b)
```

Naming

Question

What is the MJI counterpart of

```
public static Boolean valueOf(boolean b)
```

Answer

```
public int valueOf__Z__Ljava_lang_Boolean_2
  (MJIEnv env, int clsObjRef, boolean b)
```

Naming

Question

What is the MJI counterpart of

```
public abstract boolean compareAndSet(T obj,
  int expect, int update)
```

Naming

Question

What is the MJI counterpart of

```
public abstract boolean compareAndSet(T obj,
  int expect, int update)
```

Answer

```
public boolean compareAndSet__Ljava_lang_Object_2II__Z
  (MJIEnv env, int objRef, int obj, int expect, int update
```

GenPeer

The app GenPeer, which is part of the package gov.nasa.jpf.tool, generates the framework of a native peer MJI class from a class.

The command

```
java -cp /cs/fac/packages/jpf/jpf-core/build/jpf.jar \
gov.nasa.jpf.tool.GenPeer
```

generates the output

```
usage:  'GenPeer [<option>..] <className> [<method>..]'
options: -s  : system peer class (gov.nasa.jpf.vm)
         -ci : create <clinit> MJI method
         -m  : create mangled method names
         -a  : create MJI methods for all target class meth
```

GenPeer

The command

```
java -cp /cs/fac/packages/jpf/jpf-core/build/jpf.jar \
gov.nasa.jpf.tool.GenPeer -m -a java.lang.StrictMath
```

generates the output

```
import gov.nasa.jpf.vm.MJIEnv;
import gov.nasa.jpf.vm.NativePeer;

public class JPF_java_lang_StrictMath extends NativePeer {

  @MJI
  public double floorOrCeil__DDDD__D (MJIEnv env,
      int clsObjRef, double v0, double v1, double v2,
      double v3) {
    ...
  }
  ...
}
```

Native peers

To ensure that JPF uses the native peer, rather than the original class, we need to add the native peer to the host JVMs classpath.

```
target=Sine
classpath=<folder containing Sine.class>
native_classpath=<folder containing \
  JPF_java_lang_StrictMath.class>
```

jpf-nhandler

jpf-nhandler is an extension of JPF. It automatically delegates the execution of methods from JPF to the host JVM.

`https://bitbucket.org/nastaran/jpf-nhandler`

jpf-nhandler

jpf-nhandler can be applied to the `Sine` app with the following properties file.

```
@using=jpf-nhandler
target=Sine
classpath=<folder containing Sine.class>
nhandler.delegateUnhandledNative=true
```