## Apply JPF

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

## Apply 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)
```

## Apply 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?

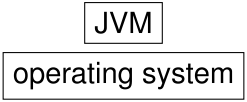
**Answer**

- Allows programmers to use code that has been already 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

EECS 4315

www.cse.yorku.ca/course/4315/

## Handling Native Methods

JPF provides several ways to handle native methods.

- Using model classes.
- Using native peers.
- Using a combination of model classes and native peers.
- Using the extension jpf-nhandler.

## Model Classes

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

**Question**

How can we capture the behaviour of the sin method?

## Model Classes

A model 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:

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

## Model 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));
  }
}
```

## Model Class

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

## Model Classes

To ensure that JPF verifies the model class, rather than the original class, we need to add the model class to JPF's class path.

```
target=Sine
classpath=C:/Users/franck/workspace/examples/bin
```

In this case, the directory `C:\Users\franck\workspace\examples\bin\java\lang` should contains the file `StrictMath.class`.