Module jdk.incubator.foreign
Package jdk.incubator.foreign

Interface NativeSymbol

All Superinterfaces:

Addressable

```
public sealed interface NativeSymbol
extends Addressable
```

A native symbol models a reference to a location (typically the entry point of a function) in a native library. A native symbol has a name, and is associated with a scope, which governs the native symbol's lifecycle. This is useful, since the library a native symbol refers to can be *unloaded*, thus invalidating the native symbol. While native symbols are typically obtained using a `symbol lookup`, it is also possible to obtain an *anonymous* native symbol, in the form of an upcall stub, that is, a reference to a dynamically-generated native symbol which can be used to call back into Java code.

Method Summary

All Methods | Static Methods | Instance Methods | Abstract Methods

Modifier and Type	Method	Description
MemoryAddress	address()	Returns the memory address associated with this symbol.
String	name()	Returns the name of this symbol.
static NativeSymbol	ofAddress(String name, MemoryAddress address, ResourceScope scope)	Creates a new symbol from given name, address and scope.
ResourceScope	scope()	Returns the resource scope associated with this symbol.

Method Details

name

```
String name()
```

Returns the name of this symbol.

Returns:
the name of this symbol

scope

```
ResourceScope scope()
```

Returns the resource scope associated with this symbol.

Returns:
the resource scope associated with this symbol

address

```
MemoryAddress address()
```

Returns the memory address associated with this symbol.

Specified by:
address in interface Addressable

Returns:
the memory address associated with this symbol

Throws:
IllegalStateException - if the scope associated with this symbol has been closed, or if access occurs from a thread other than the thread owning that scope.

ofAddress

```
static NativeSymbol ofAddress(String name,
                              MemoryAddress address,
                              ResourceScope scope)
```

Creates a new symbol from given name, address and scope.

This method is *restricted*. Restricted methods are unsafe, and, if used incorrectly, their use might crash the JVM or, worse, silently result in memory corruption. Thus, clients should refrain from depending on restricted methods, and use safe and supported functionalities, where possible.

Parameters:

`name` - the symbol name.

`address` - the symbol address.

`scope` - the symbol scope.

Returns:

A new symbol from given name, address and scope.

Throws:

`IllegalCallerException` - if access to this method occurs from a module `M` and the command line option `--enable-native-access` is either absent, or does not mention the module name `M`, or `ALL-UNNAMED` in case `M` is an unnamed module.
