Java SE 17 & JDK 17

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

# Interface SymbolLookup

#### **Functional Interface:**

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

# @FunctionalInterface

#### public interface SymbolLookup

A symbol lookup. Exposes a lookup operation for searching symbol addresses by name, see lookup(String). A symbol lookup can be used to lookup a symbol in a loaded library. Clients can obtain a loader lookup, which can be used to search symbols in libraries loaded by the current classloader (e.g. using System.load(String), or System.loadLibrary(String)). Alternatively, clients can obtain a platform-dependent lookup, to search symbols in the standard C library.

Unless otherwise specified, passing a null argument, or an array argument containing one or more null elements to a method in this class causes a NullPointerException to be thrown.

# **Method Summary**

	All Methods	Static Me	thods	Instance Me	thods	Abstract Methods	
Modifier and Type		Method		Description			
	static <b>Symboll</b>	c <b>SymbolLookup</b>		loaderLookup()		Obtains a symbol lookup suitable to find symbols in native libraries associated with the caller's classloader (that is, libraries loaded using System.loadLibrary(java.lang.String) or System.load(java.lang.String)).	
	Optional <memor< th=""><th>ryAddress&gt;</th><th>lookup(</th><th>String name)</th><th>Looks u</th><th>s up a symbol with given name in this lookup.</th></memor<>	ryAddress>	lookup(	String name)	Looks u	s up a symbol with given name in this lookup.	

#### **Method Details**

#### lookup

Optional<MemoryAddress> lookup(String name)

Looks up a symbol with given name in this lookup.

# Parameters:

name - the symbol name.

# Returns:

the memory address associated with the symbol (if any).

# loaderLookup

static SymbolLookup loaderLookup()

Obtains a symbol lookup suitable to find symbols in native libraries associated with the caller's classloader (that is, libraries loaded using System.loadLibrary(java.lang.String) or System.load(java.lang.String)).

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.

### **Returns:**

a symbol lookup suitable to find symbols in libraries loaded by the caller's classloader.

### 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.

## Report a bug or suggest an enhancement

For further API reference and developer documentation see the Java SE Documentation, which contains more detailed, developer-targeted descriptions with conceptual overviews, definitions of terms, workarounds, and working code examples. Other versions.

Java is a trademark or registered trademark of Oracle and/or its affiliates in the US and other countries.

Copyright © 1993, 2023, Oracle and/or its affiliates, 500 Oracle Parkway, Redwood Shores, CA 94065 USA.

All rights reserved. Use is subject to license terms and the documentation redistribution policy. Modify Cookie Preferences. Modify Ad Choices.