| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Instrumentation.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/instrument/IllegalClassFormatException.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/instrument/UnmodifiableClassException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/instrument/Instrumentation.html)    [**NO FRAMES**](http://docs.google.com/Instrumentation.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **java.lang.instrument**

Interface Instrumentation

public interface **Instrumentation**

This class provides services needed to instrument Java programming language code. Instrumentation is the addition of byte-codes to methods for the purpose of gathering data to be utilized by tools. Since the changes are purely additive, these tools do not modify application state or behavior. Examples of such benign tools include monitoring agents, profilers, coverage analyzers, and event loggers.

There are two ways to obtain an instance of the Instrumentation interface:

1. When a JVM is launched in a way that indicates an agent class. In that case an Instrumentation instance is passed to the premain method of the agent class.
2. When a JVM provides a mechanism to start agents sometime after the JVM is launched. In that case an Instrumentation instance is passed to the agentmain method of the agent code.

These mechanisms are described in the [package specification](http://docs.google.com/java/lang/instrument/package-summary.html).

Once an agent acquires an Instrumentation instance, the agent may call methods on the instance at any time.

**Since:** 1.5

| **Method Summary** | |
| --- | --- |
| void | [**addTransformer**](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer))([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer)            Registers the supplied transformer. |
| void | [**addTransformer**](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean))([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer, boolean canRetransform)            Registers the supplied transformer. |
| void | [**appendToBootstrapClassLoaderSearch**](http://docs.google.com/java/lang/instrument/Instrumentation.html#appendToBootstrapClassLoaderSearch(java.util.jar.JarFile))([JarFile](http://docs.google.com/java/util/jar/JarFile.html) jarfile)            Specifies a JAR file with instrumentation classes to be defined by the bootstrap class loader. |
| void | [**appendToSystemClassLoaderSearch**](http://docs.google.com/java/lang/instrument/Instrumentation.html#appendToSystemClassLoaderSearch(java.util.jar.JarFile))([JarFile](http://docs.google.com/java/util/jar/JarFile.html) jarfile)            Specifies a JAR file with instrumentation classes to be defined by the system class loader. |
| [Class](http://docs.google.com/java/lang/Class.html)[] | [**getAllLoadedClasses**](http://docs.google.com/java/lang/instrument/Instrumentation.html#getAllLoadedClasses())()            Returns an array of all classes currently loaded by the JVM. |
| [Class](http://docs.google.com/java/lang/Class.html)[] | [**getInitiatedClasses**](http://docs.google.com/java/lang/instrument/Instrumentation.html#getInitiatedClasses(java.lang.ClassLoader))([ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) loader)            Returns an array of all classes for which loader is an initiating loader. |
| long | [**getObjectSize**](http://docs.google.com/java/lang/instrument/Instrumentation.html#getObjectSize(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) objectToSize)            Returns an implementation-specific approximation of the amount of storage consumed by the specified object. |
| boolean | [**isModifiableClass**](http://docs.google.com/java/lang/instrument/Instrumentation.html#isModifiableClass(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<?> theClass)            Determines whether a class is modifiable by [retransformation](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...)) or [redefinition](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)). |
| boolean | [**isNativeMethodPrefixSupported**](http://docs.google.com/java/lang/instrument/Instrumentation.html#isNativeMethodPrefixSupported())()            Returns whether the current JVM configuration supports [setting a native method prefix](http://docs.google.com/java/lang/instrument/Instrumentation.html#setNativeMethodPrefix(java.lang.instrument.ClassFileTransformer,%20java.lang.String)). |
| boolean | [**isRedefineClassesSupported**](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRedefineClassesSupported())()            Returns whether or not the current JVM configuration supports redefinition of classes. |
| boolean | [**isRetransformClassesSupported**](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRetransformClassesSupported())()            Returns whether or not the current JVM configuration supports retransformation of classes. |
| void | [**redefineClasses**](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...))([ClassDefinition](http://docs.google.com/java/lang/instrument/ClassDefinition.html)... definitions)            Redefine the supplied set of classes using the supplied class files. |
| boolean | [**removeTransformer**](http://docs.google.com/java/lang/instrument/Instrumentation.html#removeTransformer(java.lang.instrument.ClassFileTransformer))([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer)            Unregisters the supplied transformer. |
| void | [**retransformClasses**](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...))([Class](http://docs.google.com/java/lang/Class.html)<?>... classes)            Retransform the supplied set of classes. |
| void | [**setNativeMethodPrefix**](http://docs.google.com/java/lang/instrument/Instrumentation.html#setNativeMethodPrefix(java.lang.instrument.ClassFileTransformer,%20java.lang.String))([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer, [String](http://docs.google.com/java/lang/String.html) prefix)            This method modifies the failure handling of native method resolution by allowing retry with a prefix applied to the name. |

| **Method Detail** |
| --- |

### addTransformer

void **addTransformer**([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer,  
 boolean canRetransform)

Registers the supplied transformer. All future class definitions will be seen by the transformer, except definitions of classes upon which any registered transformer is dependent. The transformer is called when classes are loaded, when they are [redefined](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)). and if canRetransform is true, when they are [retransformed](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...)). See [ClassFileTransformer.transform](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html#transform(java.lang.ClassLoader,%20java.lang.String,%20java.lang.Class,%20java.security.ProtectionDomain,%20byte%5B%5D)) for the order of transform calls. If a transformer throws an exception during execution, the JVM will still call the other registered transformers in order. The same transformer may be added more than once, but it is strongly discouraged -- avoid this by creating a new instance of tranformer class.

This method is intended for use in instrumentation, as described in the [class specification](http://docs.google.com/java/lang/instrument/Instrumentation.html).

**Parameters:**transformer - the transformer to registercanRetransform - can this transformer's transformations be retransformed **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if passed a null transformer [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if canRetransform is true and the current configuration of the JVM does not allow retransformation ([isRetransformClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRetransformClassesSupported()) is false)**Since:** 1.6

### addTransformer

void **addTransformer**([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer)

Registers the supplied transformer.

Same as addTransformer(transformer, false).

**Parameters:**transformer - the transformer to register **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if passed a null transformer**See Also:**[addTransformer(ClassFileTransformer,boolean)](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean))

### removeTransformer

boolean **removeTransformer**([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer)

Unregisters the supplied transformer. Future class definitions will not be shown to the transformer. Removes the most-recently-added matching instance of the transformer. Due to the multi-threaded nature of class loading, it is possible for a transformer to receive calls after it has been removed. Transformers should be written defensively to expect this situation.

**Parameters:**transformer - the transformer to unregister **Returns:**true if the transformer was found and removed, false if the transformer was not found **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if passed a null transformer

### isRetransformClassesSupported

boolean **isRetransformClassesSupported**()

Returns whether or not the current JVM configuration supports retransformation of classes. The ability to retransform an already loaded class is an optional capability of a JVM. Retransformation will only be supported if the Can-Retransform-Classes manifest attribute is set to true in the agent JAR file (as described in the [package specification](http://docs.google.com/java/lang/instrument/package-summary.html)) and the JVM supports this capability. During a single instantiation of a single JVM, multiple calls to this method will always return the same answer.

**Returns:**true if the current JVM configuration supports retransformation of classes, false if not.**Since:** 1.6 **See Also:**[retransformClasses(java.lang.Class...)](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...))

### retransformClasses

void **retransformClasses**([Class](http://docs.google.com/java/lang/Class.html)<?>... classes)  
 throws [UnmodifiableClassException](http://docs.google.com/java/lang/instrument/UnmodifiableClassException.html)

Retransform the supplied set of classes.

This function facilitates the instrumentation of already loaded classes. When classes are initially loaded or when they are [redefined](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)), the initial class file bytes can be transformed with the [ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html). This function reruns the transformation process (whether or not a transformation has previously occurred). This retransformation follows these steps:

* starting from the initial class file bytes
* for each transformer that was added with canRetransform false, the bytes returned by [transform](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html#transform(java.lang.ClassLoader,%20java.lang.String,%20java.lang.Class,%20java.security.ProtectionDomain,%20byte%5B%5D)) during the last class load or redefine are reused as the output of the transformation; note that this is equivalent to reapplying the previous transformation, unaltered; except that [transform](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html#transform(java.lang.ClassLoader,%20java.lang.String,%20java.lang.Class,%20java.security.ProtectionDomain,%20byte%5B%5D)) is not called
* for each transformer that was added with canRetransform true, the [transform](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html#transform(java.lang.ClassLoader,%20java.lang.String,%20java.lang.Class,%20java.security.ProtectionDomain,%20byte%5B%5D)) method is called in these transformers
* the transformed class file bytes are installed as the new definition of the class

The order of transformation is described in the ([transform](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html#transform(java.lang.ClassLoader,%20java.lang.String,%20java.lang.Class,%20java.security.ProtectionDomain,%20byte%5B%5D)) method. This same order is used in the automatic reapplication of retransformation incapable transforms.

The initial class file bytes represent the bytes passed to [ClassLoader.defineClass](http://docs.google.com/java/lang/ClassLoader.html#defineClass(byte%5B%5D,%20int,%20int)) or [redefineClasses](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)) (before any transformations were applied), however they might not exactly match them. The constant pool might not have the same layout or contents. The constant pool may have more or fewer entries. Constant pool entries may be in a different order; however, constant pool indices in the bytecodes of methods will correspond. Some attributes may not be present. Where order is not meaningful, for example the order of methods, order might not be preserved.

This method operates on a set in order to allow interdependent changes to more than one class at the same time (a retransformation of class A can require a retransformation of class B).

If a retransformed method has active stack frames, those active frames continue to run the bytecodes of the original method. The retransformed method will be used on new invokes.

This method does not cause any initialization except that which would occur under the customary JVM semantics. In other words, redefining a class does not cause its initializers to be run. The values of static variables will remain as they were prior to the call.

Instances of the retransformed class are not affected.

The retransformation may change method bodies, the constant pool and attributes. The retransformation must not add, remove or rename fields or methods, change the signatures of methods, or change inheritance. These restrictions maybe be lifted in future versions. The class file bytes are not checked, verified and installed until after the transformations have been applied, if the resultant bytes are in error this method will throw an exception.

If this method throws an exception, no classes have been retransformed.

This method is intended for use in instrumentation, as described in the [class specification](http://docs.google.com/java/lang/instrument/Instrumentation.html).

**Parameters:**classes - array of classes to retransform; a zero-length array is allowed, in this case, this method does nothing **Throws:** [UnmodifiableClassException](http://docs.google.com/java/lang/instrument/UnmodifiableClassException.html) - if a specified class cannot be modified ([isModifiableClass(java.lang.Class)](http://docs.google.com/java/lang/instrument/Instrumentation.html#isModifiableClass(java.lang.Class)) would return false) [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the current configuration of the JVM does not allow retransformation ([isRetransformClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRetransformClassesSupported()) is false) or the retransformation attempted to make unsupported changes [ClassFormatError](http://docs.google.com/java/lang/ClassFormatError.html) - if the data did not contain a valid class [NoClassDefFoundError](http://docs.google.com/java/lang/NoClassDefFoundError.html) - if the name in the class file is not equal to the name of the class [UnsupportedClassVersionError](http://docs.google.com/java/lang/UnsupportedClassVersionError.html) - if the class file version numbers are not supported [ClassCircularityError](http://docs.google.com/java/lang/ClassCircularityError.html) - if the new classes contain a circularity [LinkageError](http://docs.google.com/java/lang/LinkageError.html) - if a linkage error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the supplied classes array or any of its components is null.**Since:** 1.6 **See Also:**[isRetransformClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRetransformClassesSupported()), [addTransformer(java.lang.instrument.ClassFileTransformer, boolean)](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean)), [ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html)

### isRedefineClassesSupported

boolean **isRedefineClassesSupported**()

Returns whether or not the current JVM configuration supports redefinition of classes. The ability to redefine an already loaded class is an optional capability of a JVM. Redefinition will only be supported if the Can-Redefine-Classes manifest attribute is set to true in the agent JAR file (as described in the [package specification](http://docs.google.com/java/lang/instrument/package-summary.html)) and the JVM supports this capability. During a single instantiation of a single JVM, multiple calls to this method will always return the same answer.

**Returns:**true if the current JVM configuration supports redefinition of classes, false if not.**See Also:**[redefineClasses(java.lang.instrument.ClassDefinition...)](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...))

### redefineClasses

void **redefineClasses**([ClassDefinition](http://docs.google.com/java/lang/instrument/ClassDefinition.html)... definitions)  
 throws [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html),  
 [UnmodifiableClassException](http://docs.google.com/java/lang/instrument/UnmodifiableClassException.html)

Redefine the supplied set of classes using the supplied class files.

This method is used to replace the definition of a class without reference to the existing class file bytes, as one might do when recompiling from source for fix-and-continue debugging. Where the existing class file bytes are to be transformed (for example in bytecode instrumentation) [retransformClasses](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...)) should be used.

This method operates on a set in order to allow interdependent changes to more than one class at the same time (a redefinition of class A can require a redefinition of class B).

If a redefined method has active stack frames, those active frames continue to run the bytecodes of the original method. The redefined method will be used on new invokes.

This method does not cause any initialization except that which would occur under the customary JVM semantics. In other words, redefining a class does not cause its initializers to be run. The values of static variables will remain as they were prior to the call.

Instances of the redefined class are not affected.

The redefinition may change method bodies, the constant pool and attributes. The redefinition must not add, remove or rename fields or methods, change the signatures of methods, or change inheritance. These restrictions maybe be lifted in future versions. The class file bytes are not checked, verified and installed until after the transformations have been applied, if the resultant bytes are in error this method will throw an exception.

If this method throws an exception, no classes have been redefined.

This method is intended for use in instrumentation, as described in the [class specification](http://docs.google.com/java/lang/instrument/Instrumentation.html).

**Parameters:**definitions - array of classes to redefine with corresponding definitions; a zero-length array is allowed, in this case, this method does nothing **Throws:** [UnmodifiableClassException](http://docs.google.com/java/lang/instrument/UnmodifiableClassException.html) - if a specified class cannot be modified ([isModifiableClass(java.lang.Class)](http://docs.google.com/java/lang/instrument/Instrumentation.html#isModifiableClass(java.lang.Class)) would return false) [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the current configuration of the JVM does not allow redefinition ([isRedefineClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRedefineClassesSupported()) is false) or the redefinition attempted to make unsupported changes [ClassFormatError](http://docs.google.com/java/lang/ClassFormatError.html) - if the data did not contain a valid class [NoClassDefFoundError](http://docs.google.com/java/lang/NoClassDefFoundError.html) - if the name in the class file is not equal to the name of the class [UnsupportedClassVersionError](http://docs.google.com/java/lang/UnsupportedClassVersionError.html) - if the class file version numbers are not supported [ClassCircularityError](http://docs.google.com/java/lang/ClassCircularityError.html) - if the new classes contain a circularity [LinkageError](http://docs.google.com/java/lang/LinkageError.html) - if a linkage error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the supplied definitions array or any of its components is null [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - Can never be thrown (present for compatibility reasons only)**See Also:**[isRedefineClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRedefineClassesSupported()), [addTransformer(java.lang.instrument.ClassFileTransformer, boolean)](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean)), [ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html)

### isModifiableClass

boolean **isModifiableClass**([Class](http://docs.google.com/java/lang/Class.html)<?> theClass)

Determines whether a class is modifiable by [retransformation](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...)) or [redefinition](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)). If a class is modifiable then this method returns true. If a class is not modifiable then this method returns false.

For a class to be retransformed, [isRetransformClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRetransformClassesSupported()) must also be true. But the value of isRetransformClassesSupported() does not influence the value returned by this function. For a class to be redefined, [isRedefineClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRedefineClassesSupported()) must also be true. But the value of isRedefineClassesSupported() does not influence the value returned by this function.

Primitive classes (for example, java.lang.Integer.TYPE) and array classes are never modifiable.

**Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified class is null.**Since:** 1.6 **See Also:**[retransformClasses(java.lang.Class...)](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...)), [isRetransformClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRetransformClassesSupported()), [redefineClasses(java.lang.instrument.ClassDefinition...)](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)), [isRedefineClassesSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isRedefineClassesSupported())

### getAllLoadedClasses

[Class](http://docs.google.com/java/lang/Class.html)[] **getAllLoadedClasses**()

Returns an array of all classes currently loaded by the JVM.

**Returns:**an array containing all the classes loaded by the JVM, zero-length if there are none

### getInitiatedClasses

[Class](http://docs.google.com/java/lang/Class.html)[] **getInitiatedClasses**([ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) loader)

Returns an array of all classes for which loader is an initiating loader. If the supplied loader is null, classes initiated by the bootstrap class loader are returned.

**Parameters:**loader - the loader whose initiated class list will be returned **Returns:**an array containing all the classes for which loader is an initiating loader, zero-length if there are none

### getObjectSize

long **getObjectSize**([Object](http://docs.google.com/java/lang/Object.html) objectToSize)

Returns an implementation-specific approximation of the amount of storage consumed by the specified object. The result may include some or all of the object's overhead, and thus is useful for comparison within an implementation but not between implementations. The estimate may change during a single invocation of the JVM.

**Parameters:**objectToSize - the object to size **Returns:**an implementation-specific approximation of the amount of storage consumed by the specified object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the supplied Object is null.

### appendToBootstrapClassLoaderSearch

void **appendToBootstrapClassLoaderSearch**([JarFile](http://docs.google.com/java/util/jar/JarFile.html) jarfile)

Specifies a JAR file with instrumentation classes to be defined by the bootstrap class loader.

When the virtual machine's built-in class loader, known as the "bootstrap class loader", unsuccessfully searches for a class, the entries in the [JAR file](http://docs.google.com/java/util/jar/JarFile.html) will be searched as well.

This method may be used multiple times to add multiple JAR files to be searched in the order that this method was invoked.

The agent should take care to ensure that the JAR does not contain any classes or resources other than those to be defined by the bootstrap class loader for the purpose of instrumentation. Failure to observe this warning could result in unexpected behaviour that is difficult to diagnose. For example, suppose there is a loader L, and L's parent for delegation is the bootstrap class loader. Furthermore, a method in class C, a class defined by L, makes reference to a non-public accessor class C$1. If the JAR file contains a class C$1 then the delegation to the bootstrap class loader will cause C$1 to be defined by the bootstrap class loader. In this example an IllegalAccessError will be thrown that may cause the application to fail. One approach to avoiding these types of issues, is to use a unique package name for the instrumentation classes.

The [Java Virtual Machine Specification](http://java.sun.com/docs/books/vmspec/) specifies that a subsequent attempt to resolve a symbolic reference that the Java virtual machine has previously unsuccessfully attempted to resolve always fails with the same error that was thrown as a result of the initial resolution attempt. Consequently, if the JAR file contains an entry that corresponds to a class for which the Java virtual machine has unsuccessfully attempted to resolve a reference, then subsequent attempts to resolve that reference will fail with the same error as the initial attempt.

**Parameters:**jarfile - The JAR file to be searched when the bootstrap class loader unsuccessfully searches for a class. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If jarfile is null.**Since:** 1.6 **See Also:**[appendToSystemClassLoaderSearch(java.util.jar.JarFile)](http://docs.google.com/java/lang/instrument/Instrumentation.html#appendToSystemClassLoaderSearch(java.util.jar.JarFile)), [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html), [JarFile](http://docs.google.com/java/util/jar/JarFile.html)

### appendToSystemClassLoaderSearch

void **appendToSystemClassLoaderSearch**([JarFile](http://docs.google.com/java/util/jar/JarFile.html) jarfile)

Specifies a JAR file with instrumentation classes to be defined by the system class loader. When the system class loader for delegation (see [getSystemClassLoader()](http://docs.google.com/java/lang/ClassLoader.html#getSystemClassLoader())) unsuccessfully searches for a class, the entries in the [JarFile](http://docs.google.com/java/util/jar/JarFile.html) will be searched as well.

This method may be used multiple times to add multiple JAR files to be searched in the order that this method was invoked.

The agent should take care to ensure that the JAR does not contain any classes or resources other than those to be defined by the system class loader for the purpose of instrumentation. Failure to observe this warning could result in unexpected behaviour that is difficult to diagnose (see [appendToBootstrapClassLoaderSearch](http://docs.google.com/java/lang/instrument/Instrumentation.html#appendToBootstrapClassLoaderSearch(java.util.jar.JarFile)).

The system class loader supports adding a JAR file to be searched if it implements a method named appendToClassPathForInstrumentation which takes a single parameter of type java.lang.String. The method is not required to have public access. The name of the JAR file is obtained by invoking the [getName()](http://docs.google.com/java/util/zip/ZipFile.html#getName()) method on the jarfile and this is provided as the parameter to the appendtoClassPathForInstrumentation method.

The [Java Virtual Machine Specification](http://java.sun.com/docs/books/vmspec/) specifies that a subsequent attempt to resolve a symbolic reference that the Java virtual machine has previously unsuccessfully attempted to resolve always fails with the same error that was thrown as a result of the initial resolution attempt. Consequently, if the JAR file contains an entry that corresponds to a class for which the Java virtual machine has unsuccessfully attempted to resolve a reference, then subsequent attempts to resolve that reference will fail with the same error as the initial attempt.

This method does not change the value of java.class.path [system property](http://docs.google.com/java/lang/System.html#getProperties()).

**Parameters:**jarfile - The JAR file to be searched when the system class loader unsuccessfully searches for a class. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - If the system class loader does not support appending a a JAR file to be searched. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If jarfile is null.**Since:** 1.6 **See Also:**[appendToBootstrapClassLoaderSearch(java.util.jar.JarFile)](http://docs.google.com/java/lang/instrument/Instrumentation.html#appendToBootstrapClassLoaderSearch(java.util.jar.JarFile)), [ClassLoader.getSystemClassLoader()](http://docs.google.com/java/lang/ClassLoader.html#getSystemClassLoader()), [JarFile](http://docs.google.com/java/util/jar/JarFile.html)

### isNativeMethodPrefixSupported

boolean **isNativeMethodPrefixSupported**()

Returns whether the current JVM configuration supports [setting a native method prefix](http://docs.google.com/java/lang/instrument/Instrumentation.html#setNativeMethodPrefix(java.lang.instrument.ClassFileTransformer,%20java.lang.String)). The ability to set a native method prefix is an optional capability of a JVM. Setting a native method prefix will only be supported if the Can-Set-Native-Method-Prefix manifest attribute is set to true in the agent JAR file (as described in the [package specification](http://docs.google.com/java/lang/instrument/package-summary.html)) and the JVM supports this capability. During a single instantiation of a single JVM, multiple calls to this method will always return the same answer.

**Returns:**true if the current JVM configuration supports setting a native method prefix, false if not.**Since:** 1.6 **See Also:**[setNativeMethodPrefix(java.lang.instrument.ClassFileTransformer, java.lang.String)](http://docs.google.com/java/lang/instrument/Instrumentation.html#setNativeMethodPrefix(java.lang.instrument.ClassFileTransformer,%20java.lang.String))

### setNativeMethodPrefix

void **setNativeMethodPrefix**([ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html) transformer,  
 [String](http://docs.google.com/java/lang/String.html) prefix)

This method modifies the failure handling of native method resolution by allowing retry with a prefix applied to the name. When used with the [ClassFileTransformer](http://docs.google.com/java/lang/instrument/ClassFileTransformer.html), it enables native methods to be instrumented.

Since native methods cannot be directly instrumented (they have no bytecodes), they must be wrapped with a non-native method which can be instrumented. For example, if we had:

native boolean foo(int x);

We could transform the class file (with the ClassFileTransformer during the initial definition of the class) so that this becomes:

boolean foo(int x) {  
 *... record entry to foo ...*  
 return wrapped\_foo(x);  
 }  
   
 native boolean wrapped\_foo(int x);

Where foo becomes a wrapper for the actual native method with the appended prefix "wrapped\_". Note that "wrapped\_" would be a poor choice of prefix since it might conceivably form the name of an existing method thus something like "$$$MyAgentWrapped$$$\_" would be better but would make these examples less readable.

The wrapper will allow data to be collected on the native method call, but now the problem becomes linking up the wrapped method with the native implementation. That is, the method wrapped\_foo needs to be resolved to the native implementation of foo, which might be:

Java\_somePackage\_someClass\_foo(JNIEnv\* env, jint x)

This function allows the prefix to be specified and the proper resolution to occur. Specifically, when the standard resolution fails, the resolution is retried taking the prefix into consideration. There are two ways that resolution occurs, explicit resolution with the JNI function RegisterNatives and the normal automatic resolution. For RegisterNatives, the JVM will attempt this association:

method(foo) -> nativeImplementation(foo)

When this fails, the resolution will be retried with the specified prefix prepended to the method name, yielding the correct resolution:

method(wrapped\_foo) -> nativeImplementation(foo)

For automatic resolution, the JVM will attempt:

method(wrapped\_foo) -> nativeImplementation(wrapped\_foo)

When this fails, the resolution will be retried with the specified prefix deleted from the implementation name, yielding the correct resolution:

method(wrapped\_foo) -> nativeImplementation(foo)

Note that since the prefix is only used when standard resolution fails, native methods can be wrapped selectively.

Since each ClassFileTransformer can do its own transformation of the bytecodes, more than one layer of wrappers may be applied. Thus each transformer needs its own prefix. Since transformations are applied in order, the prefixes, if applied, will be applied in the same order (see [addTransformer](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean))). Thus if three transformers applied wrappers, foo might become $trans3\_$trans2\_$trans1\_foo. But if, say, the second transformer did not apply a wrapper to foo it would be just $trans3\_$trans1\_foo. To be able to efficiently determine the sequence of prefixes, an intermediate prefix is only applied if its non-native wrapper exists. Thus, in the last example, even though $trans1\_foo is not a native method, the $trans1\_ prefix is applied since $trans1\_foo exists.

**Parameters:**transformer - The ClassFileTransformer which wraps using this prefix.prefix - The prefix which has been applied to wrapped native methods. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if passed a null transformer. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the current configuration of the JVM does not allow setting a native method prefix ([isNativeMethodPrefixSupported()](http://docs.google.com/java/lang/instrument/Instrumentation.html#isNativeMethodPrefixSupported()) is false). [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the transformer is not registered (see [addTransformer](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean))).**Since:** 1.6

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Instrumentation.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/instrument/IllegalClassFormatException.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/instrument/UnmodifiableClassException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/instrument/Instrumentation.html)    [**NO FRAMES**](http://docs.google.com/Instrumentation.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).