| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ClassFileTransformer.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/ClassDefinition.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/instrument/IllegalClassFormatException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/instrument/ClassFileTransformer.html)    [**NO FRAMES**](http://docs.google.com/ClassFileTransformer.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 ClassFileTransformer

public interface **ClassFileTransformer**

An agent provides an implementation of this interface in order to transform class files. The transformation occurs before the class is defined by the JVM.

Note the term *class file* is used as defined in the chapter [The class File Format](http://java.sun.com/docs/books/vmspec/2nd-edition/html/ClassFile.doc.html#80959) of *The Java Virtual Machine Specification*, to mean a sequence of bytes in class file format, whether or not they reside in a file.

**Since:** 1.5 **See Also:**[Instrumentation](http://docs.google.com/java/lang/instrument/Instrumentation.html), [Instrumentation.addTransformer(java.lang.instrument.ClassFileTransformer, boolean)](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean)), [Instrumentation.removeTransformer(java.lang.instrument.ClassFileTransformer)](http://docs.google.com/java/lang/instrument/Instrumentation.html#removeTransformer(java.lang.instrument.ClassFileTransformer))

| **Method Summary** | |
| --- | --- |
| byte[] | [**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))([ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) loader, [String](http://docs.google.com/java/lang/String.html) className, [Class](http://docs.google.com/java/lang/Class.html)<?> classBeingRedefined, [ProtectionDomain](http://docs.google.com/java/security/ProtectionDomain.html) protectionDomain, byte[] classfileBuffer)            The implementation of this method may transform the supplied class file and return a new replacement class file. |

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

### transform

byte[] **transform**([ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) loader,  
 [String](http://docs.google.com/java/lang/String.html) className,  
 [Class](http://docs.google.com/java/lang/Class.html)<?> classBeingRedefined,  
 [ProtectionDomain](http://docs.google.com/java/security/ProtectionDomain.html) protectionDomain,  
 byte[] classfileBuffer)  
 throws [IllegalClassFormatException](http://docs.google.com/java/lang/instrument/IllegalClassFormatException.html)

The implementation of this method may transform the supplied class file and return a new replacement class file.

There are two kinds of transformers, determined by the canRetransform parameter of [Instrumentation.addTransformer(ClassFileTransformer,boolean)](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean)):

* *retransformation capable* transformers that were added with canRetransform as true
* *retransformation incapable* transformers that were added with canRetransform as false or where added with [Instrumentation.addTransformer(ClassFileTransformer)](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer))

Once a transformer has been registered with [addTransformer](http://docs.google.com/java/lang/instrument/Instrumentation.html#addTransformer(java.lang.instrument.ClassFileTransformer,%20boolean)), the transformer will be called for every new class definition and every class redefinition. Retransformation capable transformers will also be called on every class retransformation. The request for a new class definition is made with [ClassLoader.defineClass](http://docs.google.com/java/lang/ClassLoader.html#defineClass(byte%5B%5D,%20int,%20int)) or its native equivalents. The request for a class redefinition is made with [Instrumentation.redefineClasses](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...)) or its native equivalents. The request for a class retransformation is made with [Instrumentation.retransformClasses](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...)) or its native equivalents. The transformer is called during the processing of the request, before the class file bytes have been verified or applied. When there are multiple transformers, transformations are composed by chaining the transform calls. That is, the byte array returned by one call to transform becomes the input (via the classfileBuffer parameter) to the next call.

Transformations are applied in the following order:

* Retransformation incapable transformers
* Retransformation incapable native transformers
* Retransformation capable transformers
* Retransformation capable native transformers

For retransformations, the retransformation incapable transformers are not called, instead the result of the previous transformation is reused. In all other cases, this method is called. Within each of these groupings, transformers are called in the order registered. Native transformers are provided by the ClassFileLoadHook event in the Java Virtual Machine Tool Interface).

The input (via the classfileBuffer parameter) to the first transformer is:

* for new class definition, the bytes passed to ClassLoader.defineClass
* for class redefinition, definitions.getDefinitionClassFile() where definitions is the parameter to [Instrumentation.redefineClasses](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...))
* for class retransformation, the bytes passed to the new class definition or, if redefined, the last redefinition, with all transformations made by retransformation incapable transformers reapplied automatically and unaltered; for details see [Instrumentation.retransformClasses](http://docs.google.com/java/lang/instrument/Instrumentation.html#retransformClasses(java.lang.Class...))

If the implementing method determines that no transformations are needed, it should return null. Otherwise, it should create a new byte[] array, copy the input classfileBuffer into it, along with all desired transformations, and return the new array. The input classfileBuffer must not be modified.

In the retransform and redefine cases, the transformer must support the redefinition semantics: if a class that the transformer changed during initial definition is later retransformed or redefined, the transformer must insure that the second class output class file is a legal redefinition of the first output class file.

If the transformer throws an exception (which it doesn't catch), subsequent transformers will still be called and the load, redefine or retransform will still be attempted. Thus, throwing an exception has the same effect as returning null. To prevent unexpected behavior when unchecked exceptions are generated in transformer code, a transformer can catch Throwable. If the transformer believes the classFileBuffer does not represent a validly formatted class file, it should throw an IllegalClassFormatException; while this has the same effect as returning null. it facilitates the logging or debugging of format corruptions.

**Parameters:**loader - the defining loader of the class to be transformed, may be null if the bootstrap loaderclassName - the name of the class in the internal form of fully qualified class and interface names as defined in *The Java Virtual Machine Specification*. For example, "java/util/List".classBeingRedefined - if this is triggered by a redefine or retransform, the class being redefined or retransformed; if this is a class load, nullprotectionDomain - the protection domain of the class being defined or redefinedclassfileBuffer - the input byte buffer in class file format - must not be modified **Returns:**a well-formed class file buffer (the result of the transform), or null if no transform is performed. **Throws:** [IllegalClassFormatException](http://docs.google.com/java/lang/instrument/IllegalClassFormatException.html) - if the input does not represent a well-formed class file**See Also:**[Instrumentation.redefineClasses(java.lang.instrument.ClassDefinition...)](http://docs.google.com/java/lang/instrument/Instrumentation.html#redefineClasses(java.lang.instrument.ClassDefinition...))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ClassFileTransformer.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/ClassDefinition.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/instrument/IllegalClassFormatException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/instrument/ClassFileTransformer.html)    [**NO FRAMES**](http://docs.google.com/ClassFileTransformer.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).