| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Class.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/CharSequence.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/ClassCastException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Class.html)    [**NO FRAMES**](http://docs.google.com/Class.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **java.lang**

Class Class<T>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.lang.Class<T>**

**Type Parameters:**T - the type of the class modeled by this Class object. For example, the type of String.class is Class<String>. Use Class<?> if the class being modeled is unknown. **All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [AnnotatedElement](http://docs.google.com/java/lang/reflect/AnnotatedElement.html), [GenericDeclaration](http://docs.google.com/java/lang/reflect/GenericDeclaration.html), [Type](http://docs.google.com/java/lang/reflect/Type.html)

public final class **Class<T>**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html), [GenericDeclaration](http://docs.google.com/java/lang/reflect/GenericDeclaration.html), [Type](http://docs.google.com/java/lang/reflect/Type.html), [AnnotatedElement](http://docs.google.com/java/lang/reflect/AnnotatedElement.html)

Instances of the class Class represent classes and interfaces in a running Java application. An enum is a kind of class and an annotation is a kind of interface. Every array also belongs to a class that is reflected as a Class object that is shared by all arrays with the same element type and number of dimensions. The primitive Java types (boolean, byte, char, short, int, long, float, and double), and the keyword void are also represented as Class objects.

Class has no public constructor. Instead Class objects are constructed automatically by the Java Virtual Machine as classes are loaded and by calls to the defineClass method in the class loader.

The following example uses a Class object to print the class name of an object:

void printClassName(Object obj) {  
 System.out.println("The class of " + obj +  
 " is " + obj.getClass().getName());  
 }

It is also possible to get the Class object for a named type (or for void) using a class literal (JLS Section [15.8.2](http://java.sun.com/docs/books/jls/second_edition/html/expressions.doc.html#251530)). For example:

System.out.println("The name of class Foo is: "+Foo.class.getName());

**Since:** JDK1.0 **See Also:**[ClassLoader.defineClass(byte[], int, int)](http://docs.google.com/java/lang/ClassLoader.html#defineClass(byte%5B%5D,%20int,%20int)), [Serialized Form](http://docs.google.com/serialized-form.html#java.lang.Class)

| **Method Summary** | |
| --- | --- |
| | <U> [Class](http://docs.google.com/java/lang/Class.html)<? extends U> | | --- | | [**asSubclass**](http://docs.google.com/java/lang/Class.html#asSubclass(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<U> clazz)            Casts this Class object to represent a subclass of the class represented by the specified class object. |
| [T](http://docs.google.com/java/lang/Class.html) | [**cast**](http://docs.google.com/java/lang/Class.html#cast(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Casts an object to the class or interface represented by this Class object. |
| boolean | [**desiredAssertionStatus**](http://docs.google.com/java/lang/Class.html#desiredAssertionStatus())()            Returns the assertion status that would be assigned to this class if it were to be initialized at the time this method is invoked. |
| static [Class](http://docs.google.com/java/lang/Class.html)<?> | [**forName**](http://docs.google.com/java/lang/Class.html#forName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) className)            Returns the Class object associated with the class or interface with the given string name. |
| static [Class](http://docs.google.com/java/lang/Class.html)<?> | [**forName**](http://docs.google.com/java/lang/Class.html#forName(java.lang.String,%20boolean,%20java.lang.ClassLoader))([String](http://docs.google.com/java/lang/String.html) name, boolean initialize, [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) loader)            Returns the Class object associated with the class or interface with the given string name, using the given class loader. |
| | <A extends [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)>  A | | --- | | [**getAnnotation**](http://docs.google.com/java/lang/Class.html#getAnnotation(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<A> annotationClass)            Returns this element's annotation for the specified type if such an annotation is present, else null. |
| [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)[] | [**getAnnotations**](http://docs.google.com/java/lang/Class.html#getAnnotations())()            Returns all annotations present on this element. |
| [String](http://docs.google.com/java/lang/String.html) | [**getCanonicalName**](http://docs.google.com/java/lang/Class.html#getCanonicalName())()            Returns the canonical name of the underlying class as defined by the Java Language Specification. |
| [Class](http://docs.google.com/java/lang/Class.html)<?>[] | [**getClasses**](http://docs.google.com/java/lang/Class.html#getClasses())()            Returns an array containing Class objects representing all the public classes and interfaces that are members of the class represented by this Class object. |
| [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) | [**getClassLoader**](http://docs.google.com/java/lang/Class.html#getClassLoader())()            Returns the class loader for the class. |
| [Class](http://docs.google.com/java/lang/Class.html)<?> | [**getComponentType**](http://docs.google.com/java/lang/Class.html#getComponentType())()            Returns the Class representing the component type of an array. |
| [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<[T](http://docs.google.com/java/lang/Class.html)> | [**getConstructor**](http://docs.google.com/java/lang/Class.html#getConstructor(java.lang.Class...))([Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)            Returns a Constructor object that reflects the specified public constructor of the class represented by this Class object. |
| [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<?>[] | [**getConstructors**](http://docs.google.com/java/lang/Class.html#getConstructors())()            Returns an array containing Constructor objects reflecting all the public constructors of the class represented by this Class object. |
| [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)[] | [**getDeclaredAnnotations**](http://docs.google.com/java/lang/Class.html#getDeclaredAnnotations())()            Returns all annotations that are directly present on this element. |
| [Class](http://docs.google.com/java/lang/Class.html)<?>[] | [**getDeclaredClasses**](http://docs.google.com/java/lang/Class.html#getDeclaredClasses())()            Returns an array of Class objects reflecting all the classes and interfaces declared as members of the class represented by this Class object. |
| [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<[T](http://docs.google.com/java/lang/Class.html)> | [**getDeclaredConstructor**](http://docs.google.com/java/lang/Class.html#getDeclaredConstructor(java.lang.Class...))([Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)            Returns a Constructor object that reflects the specified constructor of the class or interface represented by this Class object. |
| [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<?>[] | [**getDeclaredConstructors**](http://docs.google.com/java/lang/Class.html#getDeclaredConstructors())()            Returns an array of Constructor objects reflecting all the constructors declared by the class represented by this Class object. |
| [Field](http://docs.google.com/java/lang/reflect/Field.html) | [**getDeclaredField**](http://docs.google.com/java/lang/Class.html#getDeclaredField(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Returns a Field object that reflects the specified declared field of the class or interface represented by this Class object. |
| [Field](http://docs.google.com/java/lang/reflect/Field.html)[] | [**getDeclaredFields**](http://docs.google.com/java/lang/Class.html#getDeclaredFields())()            Returns an array of Field objects reflecting all the fields declared by the class or interface represented by this Class object. |
| [Method](http://docs.google.com/java/lang/reflect/Method.html) | [**getDeclaredMethod**](http://docs.google.com/java/lang/Class.html#getDeclaredMethod(java.lang.String,%20java.lang.Class...))([String](http://docs.google.com/java/lang/String.html) name, [Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)            Returns a Method object that reflects the specified declared method of the class or interface represented by this Class object. |
| [Method](http://docs.google.com/java/lang/reflect/Method.html)[] | [**getDeclaredMethods**](http://docs.google.com/java/lang/Class.html#getDeclaredMethods())()            Returns an array of Method objects reflecting all the methods declared by the class or interface represented by this Class object. |
| [Class](http://docs.google.com/java/lang/Class.html)<?> | [**getDeclaringClass**](http://docs.google.com/java/lang/Class.html#getDeclaringClass())()            If the class or interface represented by this Class object is a member of another class, returns the Class object representing the class in which it was declared. |
| [Class](http://docs.google.com/java/lang/Class.html)<?> | [**getEnclosingClass**](http://docs.google.com/java/lang/Class.html#getEnclosingClass())()            Returns the immediately enclosing class of the underlying class. |
| [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<?> | [**getEnclosingConstructor**](http://docs.google.com/java/lang/Class.html#getEnclosingConstructor())()            If this Class object represents a local or anonymous class within a constructor, returns a [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html) object representing the immediately enclosing constructor of the underlying class. |
| [Method](http://docs.google.com/java/lang/reflect/Method.html) | [**getEnclosingMethod**](http://docs.google.com/java/lang/Class.html#getEnclosingMethod())()            If this Class object represents a local or anonymous class within a method, returns a [Method](http://docs.google.com/java/lang/reflect/Method.html) object representing the immediately enclosing method of the underlying class. |
| [T](http://docs.google.com/java/lang/Class.html)[] | [**getEnumConstants**](http://docs.google.com/java/lang/Class.html#getEnumConstants())()            Returns the elements of this enum class or null if this Class object does not represent an enum type. |
| [Field](http://docs.google.com/java/lang/reflect/Field.html) | [**getField**](http://docs.google.com/java/lang/Class.html#getField(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Returns a Field object that reflects the specified public member field of the class or interface represented by this Class object. |
| [Field](http://docs.google.com/java/lang/reflect/Field.html)[] | [**getFields**](http://docs.google.com/java/lang/Class.html#getFields())()            Returns an array containing Field objects reflecting all the accessible public fields of the class or interface represented by this Class object. |
| [Type](http://docs.google.com/java/lang/reflect/Type.html)[] | [**getGenericInterfaces**](http://docs.google.com/java/lang/Class.html#getGenericInterfaces())()            Returns the Types representing the interfaces directly implemented by the class or interface represented by this object. |
| [Type](http://docs.google.com/java/lang/reflect/Type.html) | [**getGenericSuperclass**](http://docs.google.com/java/lang/Class.html#getGenericSuperclass())()            Returns the Type representing the direct superclass of the entity (class, interface, primitive type or void) represented by this Class. |
| [Class](http://docs.google.com/java/lang/Class.html)<?>[] | [**getInterfaces**](http://docs.google.com/java/lang/Class.html#getInterfaces())()            Determines the interfaces implemented by the class or interface represented by this object. |
| [Method](http://docs.google.com/java/lang/reflect/Method.html) | [**getMethod**](http://docs.google.com/java/lang/Class.html#getMethod(java.lang.String,%20java.lang.Class...))([String](http://docs.google.com/java/lang/String.html) name, [Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)            Returns a Method object that reflects the specified public member method of the class or interface represented by this Class object. |
| [Method](http://docs.google.com/java/lang/reflect/Method.html)[] | [**getMethods**](http://docs.google.com/java/lang/Class.html#getMethods())()            Returns an array containing Method objects reflecting all the public *member* methods of the class or interface represented by this Class object, including those declared by the class or interface and those inherited from superclasses and superinterfaces. |
| int | [**getModifiers**](http://docs.google.com/java/lang/Class.html#getModifiers())()            Returns the Java language modifiers for this class or interface, encoded in an integer. |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/java/lang/Class.html#getName())()            Returns the name of the entity (class, interface, array class, primitive type, or void) represented by this Class object, as a String. |
| [Package](http://docs.google.com/java/lang/Package.html) | [**getPackage**](http://docs.google.com/java/lang/Class.html#getPackage())()            Gets the package for this class. |
| [ProtectionDomain](http://docs.google.com/java/security/ProtectionDomain.html) | [**getProtectionDomain**](http://docs.google.com/java/lang/Class.html#getProtectionDomain())()            Returns the ProtectionDomain of this class. |
| [URL](http://docs.google.com/java/net/URL.html) | [**getResource**](http://docs.google.com/java/lang/Class.html#getResource(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Finds a resource with a given name. |
| [InputStream](http://docs.google.com/java/io/InputStream.html) | [**getResourceAsStream**](http://docs.google.com/java/lang/Class.html#getResourceAsStream(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Finds a resource with a given name. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**getSigners**](http://docs.google.com/java/lang/Class.html#getSigners())()            Gets the signers of this class. |
| [String](http://docs.google.com/java/lang/String.html) | [**getSimpleName**](http://docs.google.com/java/lang/Class.html#getSimpleName())()            Returns the simple name of the underlying class as given in the source code. |
| [Class](http://docs.google.com/java/lang/Class.html)<? super [T](http://docs.google.com/java/lang/Class.html)> | [**getSuperclass**](http://docs.google.com/java/lang/Class.html#getSuperclass())()            Returns the Class representing the superclass of the entity (class, interface, primitive type or void) represented by this Class. |
| [TypeVariable](http://docs.google.com/java/lang/reflect/TypeVariable.html)<[Class](http://docs.google.com/java/lang/Class.html)<[T](http://docs.google.com/java/lang/Class.html)>>[] | [**getTypeParameters**](http://docs.google.com/java/lang/Class.html#getTypeParameters())()            Returns an array of TypeVariable objects that represent the type variables declared by the generic declaration represented by this GenericDeclaration object, in declaration order. |
| boolean | [**isAnnotation**](http://docs.google.com/java/lang/Class.html#isAnnotation())()            Returns true if this Class object represents an annotation type. |
| boolean | [**isAnnotationPresent**](http://docs.google.com/java/lang/Class.html#isAnnotationPresent(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<? extends [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)> annotationClass)            Returns true if an annotation for the specified type is present on this element, else false. |
| boolean | [**isAnonymousClass**](http://docs.google.com/java/lang/Class.html#isAnonymousClass())()            Returns true if and only if the underlying class is an anonymous class. |
| boolean | [**isArray**](http://docs.google.com/java/lang/Class.html#isArray())()            Determines if this Class object represents an array class. |
| boolean | [**isAssignableFrom**](http://docs.google.com/java/lang/Class.html#isAssignableFrom(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<?> cls)            Determines if the class or interface represented by this Class object is either the same as, or is a superclass or superinterface of, the class or interface represented by the specified Class parameter. |
| boolean | [**isEnum**](http://docs.google.com/java/lang/Class.html#isEnum())()            Returns true if and only if this class was declared as an enum in the source code. |
| boolean | [**isInstance**](http://docs.google.com/java/lang/Class.html#isInstance(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Determines if the specified Object is assignment-compatible with the object represented by this Class. |
| boolean | [**isInterface**](http://docs.google.com/java/lang/Class.html#isInterface())()            Determines if the specified Class object represents an interface type. |
| boolean | [**isLocalClass**](http://docs.google.com/java/lang/Class.html#isLocalClass())()            Returns true if and only if the underlying class is a local class. |
| boolean | [**isMemberClass**](http://docs.google.com/java/lang/Class.html#isMemberClass())()            Returns true if and only if the underlying class is a member class. |
| boolean | [**isPrimitive**](http://docs.google.com/java/lang/Class.html#isPrimitive())()            Determines if the specified Class object represents a primitive type. |
| boolean | [**isSynthetic**](http://docs.google.com/java/lang/Class.html#isSynthetic())()            Returns true if this class is a synthetic class; returns false otherwise. |
| [T](http://docs.google.com/java/lang/Class.html) | [**newInstance**](http://docs.google.com/java/lang/Class.html#newInstance())()            Creates a new instance of the class represented by this Class object. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/Class.html#toString())()            Converts the object to a string. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Converts the object to a string. The string representation is the string "class" or "interface", followed by a space, and then by the fully qualified name of the class in the format returned by getName. If this Class object represents a primitive type, this method returns the name of the primitive type. If this Class object represents void this method returns "void".

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this class object.

### forName

public static [Class](http://docs.google.com/java/lang/Class.html)<?> **forName**([String](http://docs.google.com/java/lang/String.html) className)  
 throws [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Returns the Class object associated with the class or interface with the given string name. Invoking this method is equivalent to:

Class.forName(className, true, currentLoader)

where currentLoader denotes the defining class loader of the current class.

For example, the following code fragment returns the runtime Class descriptor for the class named java.lang.Thread:

Class t = Class.forName("java.lang.Thread")

A call to forName("X") causes the class named X to be initialized.

**Parameters:**className - the fully qualified name of the desired class. **Returns:**the Class object for the class with the specified name. **Throws:** [LinkageError](http://docs.google.com/java/lang/LinkageError.html) - if the linkage fails [ExceptionInInitializerError](http://docs.google.com/java/lang/ExceptionInInitializerError.html) - if the initialization provoked by this method fails [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if the class cannot be located

### forName

public static [Class](http://docs.google.com/java/lang/Class.html)<?> **forName**([String](http://docs.google.com/java/lang/String.html) name,  
 boolean initialize,  
 [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) loader)  
 throws [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Returns the Class object associated with the class or interface with the given string name, using the given class loader. Given the fully qualified name for a class or interface (in the same format returned by getName) this method attempts to locate, load, and link the class or interface. The specified class loader is used to load the class or interface. If the parameter loader is null, the class is loaded through the bootstrap class loader. The class is initialized only if the initialize parameter is true and if it has not been initialized earlier.

If name denotes a primitive type or void, an attempt will be made to locate a user-defined class in the unnamed package whose name is name. Therefore, this method cannot be used to obtain any of the Class objects representing primitive types or void.

If name denotes an array class, the component type of the array class is loaded but not initialized.

For example, in an instance method the expression:

Class.forName("Foo")

is equivalent to:

Class.forName("Foo", true, this.getClass().getClassLoader())

Note that this method throws errors related to loading, linking or initializing as specified in Sections 12.2, 12.3 and 12.4 of *The Java Language Specification*. Note that this method does not check whether the requested class is accessible to its caller.

If the loader is null, and a security manager is present, and the caller's class loader is not null, then this method calls the security manager's checkPermission method with a RuntimePermission("getClassLoader") permission to ensure it's ok to access the bootstrap class loader.

**Parameters:**name - fully qualified name of the desired classinitialize - whether the class must be initializedloader - class loader from which the class must be loaded **Returns:**class object representing the desired class **Throws:** [LinkageError](http://docs.google.com/java/lang/LinkageError.html) - if the linkage fails [ExceptionInInitializerError](http://docs.google.com/java/lang/ExceptionInInitializerError.html) - if the initialization provoked by this method fails [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if the class cannot be located by the specified class loader**Since:** 1.2 **See Also:**[forName(String)](http://docs.google.com/java/lang/Class.html#forName(java.lang.String)), [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html)

### newInstance

public [T](http://docs.google.com/java/lang/Class.html) **newInstance**()  
 throws [InstantiationException](http://docs.google.com/java/lang/InstantiationException.html),  
 [IllegalAccessException](http://docs.google.com/java/lang/IllegalAccessException.html)

Creates a new instance of the class represented by this Class object. The class is instantiated as if by a new expression with an empty argument list. The class is initialized if it has not already been initialized.

Note that this method propagates any exception thrown by the nullary constructor, including a checked exception. Use of this method effectively bypasses the compile-time exception checking that would otherwise be performed by the compiler. The [Constructor.newInstance](http://docs.google.com/java/lang/reflect/Constructor.html#newInstance(java.lang.Object...)) method avoids this problem by wrapping any exception thrown by the constructor in a (checked) [InvocationTargetException](http://docs.google.com/java/lang/reflect/InvocationTargetException.html).

**Returns:**a newly allocated instance of the class represented by this object. **Throws:** [IllegalAccessException](http://docs.google.com/java/lang/IllegalAccessException.html) - if the class or its nullary constructor is not accessible. [InstantiationException](http://docs.google.com/java/lang/InstantiationException.html) - if this Class represents an abstract class, an interface, an array class, a primitive type, or void; or if the class has no nullary constructor; or if the instantiation fails for some other reason. [ExceptionInInitializerError](http://docs.google.com/java/lang/ExceptionInInitializerError.html) - if the initialization provoked by this method fails. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies creation of new instances of this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

### isInstance

public boolean **isInstance**([Object](http://docs.google.com/java/lang/Object.html) obj)

Determines if the specified Object is assignment-compatible with the object represented by this Class. This method is the dynamic equivalent of the Java language instanceof operator. The method returns true if the specified Object argument is non-null and can be cast to the reference type represented by this Class object without raising a ClassCastException. It returns false otherwise.

Specifically, if this Class object represents a declared class, this method returns true if the specified Object argument is an instance of the represented class (or of any of its subclasses); it returns false otherwise. If this Class object represents an array class, this method returns true if the specified Object argument can be converted to an object of the array class by an identity conversion or by a widening reference conversion; it returns false otherwise. If this Class object represents an interface, this method returns true if the class or any superclass of the specified Object argument implements this interface; it returns false otherwise. If this Class object represents a primitive type, this method returns false.

**Parameters:**obj - the object to check **Returns:**true if obj is an instance of this class**Since:** JDK1.1

### isAssignableFrom

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

Determines if the class or interface represented by this Class object is either the same as, or is a superclass or superinterface of, the class or interface represented by the specified Class parameter. It returns true if so; otherwise it returns false. If this Class object represents a primitive type, this method returns true if the specified Class parameter is exactly this Class object; otherwise it returns false.

Specifically, this method tests whether the type represented by the specified Class parameter can be converted to the type represented by this Class object via an identity conversion or via a widening reference conversion. See *The Java Language Specification*, sections 5.1.1 and 5.1.4 , for details.

**Parameters:**cls - the Class object to be checked **Returns:**the boolean value indicating whether objects of the type cls can be assigned to objects of this class **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified Class parameter is null.**Since:** JDK1.1

### isInterface

public boolean **isInterface**()

Determines if the specified Class object represents an interface type.

**Returns:**true if this object represents an interface; false otherwise.

### isArray

public boolean **isArray**()

Determines if this Class object represents an array class.

**Returns:**true if this object represents an array class; false otherwise.**Since:** JDK1.1

### isPrimitive

public boolean **isPrimitive**()

Determines if the specified Class object represents a primitive type.

There are nine predefined Class objects to represent the eight primitive types and void. These are created by the Java Virtual Machine, and have the same names as the primitive types that they represent, namely boolean, byte, char, short, int, long, float, and double.

These objects may only be accessed via the following public static final variables, and are the only Class objects for which this method returns true.

**Returns:**true if and only if this class represents a primitive type**Since:** JDK1.1 **See Also:**[Boolean.TYPE](http://docs.google.com/java/lang/Boolean.html#TYPE), [Character.TYPE](http://docs.google.com/java/lang/Character.html#TYPE), [Byte.TYPE](http://docs.google.com/java/lang/Byte.html#TYPE), [Short.TYPE](http://docs.google.com/java/lang/Short.html#TYPE), [Integer.TYPE](http://docs.google.com/java/lang/Integer.html#TYPE), [Long.TYPE](http://docs.google.com/java/lang/Long.html#TYPE), [Float.TYPE](http://docs.google.com/java/lang/Float.html#TYPE), [Double.TYPE](http://docs.google.com/java/lang/Double.html#TYPE), [Void.TYPE](http://docs.google.com/java/lang/Void.html#TYPE)

### isAnnotation

public boolean **isAnnotation**()

Returns true if this Class object represents an annotation type. Note that if this method returns true, [isInterface()](http://docs.google.com/java/lang/Class.html#isInterface()) would also return true, as all annotation types are also interfaces.

**Returns:**true if this class object represents an annotation type; false otherwise**Since:** 1.5

### isSynthetic

public boolean **isSynthetic**()

Returns true if this class is a synthetic class; returns false otherwise.

**Returns:**true if and only if this class is a synthetic class as defined by the Java Language Specification.**Since:** 1.5

### getName

public [String](http://docs.google.com/java/lang/String.html) **getName**()

Returns the name of the entity (class, interface, array class, primitive type, or void) represented by this Class object, as a String.

If this class object represents a reference type that is not an array type then the binary name of the class is returned, as specified by the Java Language Specification, Second Edition.

If this class object represents a primitive type or void, then the name returned is a String equal to the Java language keyword corresponding to the primitive type or void.

If this class object represents a class of arrays, then the internal form of the name consists of the name of the element type preceded by one or more '[' characters representing the depth of the array nesting. The encoding of element type names is as follows:

| Element Type |  | Encoding |
| --- | --- | --- |
| boolean |  | Z |
| byte |  | B |
| char |  | C |
| class or interface |  | L*classname*; |
| double |  | D |
| float |  | F |
| int |  | I |
| long |  | J |
| short |  | S |

The class or interface name *classname* is the binary name of the class specified above.

Examples:

String.class.getName()  
 returns "java.lang.String"  
 byte.class.getName()  
 returns "byte"  
 (new Object[3]).getClass().getName()  
 returns "[Ljava.lang.Object;"  
 (new int[3][4][5][6][7][8][9]).getClass().getName()  
 returns "[[[[[[[I"

**Returns:**the name of the class or interface represented by this object.

### getClassLoader

public [ClassLoader](http://docs.google.com/java/lang/ClassLoader.html) **getClassLoader**()

Returns the class loader for the class. Some implementations may use null to represent the bootstrap class loader. This method will return null in such implementations if this class was loaded by the bootstrap class loader.

If a security manager is present, and the caller's class loader is not null and the caller's class loader is not the same as or an ancestor of the class loader for the class whose class loader is requested, then this method calls the security manager's checkPermission method with a RuntimePermission("getClassLoader") permission to ensure it's ok to access the class loader for the class.

If this object represents a primitive type or void, null is returned.

**Returns:**the class loader that loaded the class or interface represented by this object. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkPermission method denies access to the class loader for the class.**See Also:**[ClassLoader](http://docs.google.com/java/lang/ClassLoader.html), [SecurityManager.checkPermission(java.security.Permission)](http://docs.google.com/java/lang/SecurityManager.html#checkPermission(java.security.Permission)), [RuntimePermission](http://docs.google.com/java/lang/RuntimePermission.html)

### getTypeParameters

public [TypeVariable](http://docs.google.com/java/lang/reflect/TypeVariable.html)<[Class](http://docs.google.com/java/lang/Class.html)<[T](http://docs.google.com/java/lang/Class.html)>>[] **getTypeParameters**()

Returns an array of TypeVariable objects that represent the type variables declared by the generic declaration represented by this GenericDeclaration object, in declaration order. Returns an array of length 0 if the underlying generic declaration declares no type variables.

**Specified by:**[getTypeParameters](http://docs.google.com/java/lang/reflect/GenericDeclaration.html#getTypeParameters()) in interface [GenericDeclaration](http://docs.google.com/java/lang/reflect/GenericDeclaration.html) **Returns:**an array of TypeVariable objects that represent the type variables declared by this generic declaration **Throws:** GenericSignatureFormatError - if the generic signature of this generic declaration does not conform to the format specified in the Java Virtual Machine Specification, 3rd edition**Since:** 1.5

### getSuperclass

public [Class](http://docs.google.com/java/lang/Class.html)<? super [T](http://docs.google.com/java/lang/Class.html)> **getSuperclass**()

Returns the Class representing the superclass of the entity (class, interface, primitive type or void) represented by this Class. If this Class represents either the Object class, an interface, a primitive type, or void, then null is returned. If this object represents an array class then the Class object representing the Object class is returned.

**Returns:**the superclass of the class represented by this object.

### getGenericSuperclass

public [Type](http://docs.google.com/java/lang/reflect/Type.html) **getGenericSuperclass**()

Returns the Type representing the direct superclass of the entity (class, interface, primitive type or void) represented by this Class.

If the superclass is a parameterized type, the Type object returned must accurately reflect the actual type parameters used in the source code. The parameterized type representing the superclass is created if it had not been created before. See the declaration of [ParameterizedType](http://docs.google.com/java/lang/reflect/ParameterizedType.html) for the semantics of the creation process for parameterized types. If this Class represents either the Object class, an interface, a primitive type, or void, then null is returned. If this object represents an array class then the Class object representing the Object class is returned.

**Returns:**the superclass of the class represented by this object **Throws:** GenericSignatureFormatError - if the generic class signature does not conform to the format specified in the Java Virtual Machine Specification, 3rd edition [TypeNotPresentException](http://docs.google.com/java/lang/TypeNotPresentException.html) - if the generic superclass refers to a non-existent type declaration MalformedParameterizedTypeException - if the generic superclass refers to a parameterized type that cannot be instantiated for any reason**Since:** 1.5

### getPackage

public [Package](http://docs.google.com/java/lang/Package.html) **getPackage**()

Gets the package for this class. The class loader of this class is used to find the package. If the class was loaded by the bootstrap class loader the set of packages loaded from CLASSPATH is searched to find the package of the class. Null is returned if no package object was created by the class loader of this class.

Packages have attributes for versions and specifications only if the information was defined in the manifests that accompany the classes, and if the class loader created the package instance with the attributes from the manifest.

**Returns:**the package of the class, or null if no package information is available from the archive or codebase.

### getInterfaces

public [Class](http://docs.google.com/java/lang/Class.html)<?>[] **getInterfaces**()

Determines the interfaces implemented by the class or interface represented by this object.

If this object represents a class, the return value is an array containing objects representing all interfaces implemented by the class. The order of the interface objects in the array corresponds to the order of the interface names in the implements clause of the declaration of the class represented by this object. For example, given the declaration:

class Shimmer implements FloorWax, DessertTopping { ... }

suppose the value of s is an instance of Shimmer; the value of the expression:

s.getClass().getInterfaces()[0]

is the Class object that represents interface FloorWax; and the value of:

s.getClass().getInterfaces()[1]

is the Class object that represents interface DessertTopping.

If this object represents an interface, the array contains objects representing all interfaces extended by the interface. The order of the interface objects in the array corresponds to the order of the interface names in the extends clause of the declaration of the interface represented by this object.

If this object represents a class or interface that implements no interfaces, the method returns an array of length 0.

If this object represents a primitive type or void, the method returns an array of length 0.

**Returns:**an array of interfaces implemented by this class.

### getGenericInterfaces

public [Type](http://docs.google.com/java/lang/reflect/Type.html)[] **getGenericInterfaces**()

Returns the Types representing the interfaces directly implemented by the class or interface represented by this object.

If a superinterface is a parameterized type, the Type object returned for it must accurately reflect the actual type parameters used in the source code. The parameterized type representing each superinterface is created if it had not been created before. See the declaration of [ParameterizedType](http://docs.google.com/java/lang/reflect/ParameterizedType.html) for the semantics of the creation process for parameterized types.

If this object represents a class, the return value is an array containing objects representing all interfaces implemented by the class. The order of the interface objects in the array corresponds to the order of the interface names in the implements clause of the declaration of the class represented by this object. In the case of an array class, the interfaces Cloneable and Serializable are returned in that order.

If this object represents an interface, the array contains objects representing all interfaces directly extended by the interface. The order of the interface objects in the array corresponds to the order of the interface names in the extends clause of the declaration of the interface represented by this object.

If this object represents a class or interface that implements no interfaces, the method returns an array of length 0.

If this object represents a primitive type or void, the method returns an array of length 0.

**Returns:**an array of interfaces implemented by this class **Throws:** GenericSignatureFormatError - if the generic class signature does not conform to the format specified in the Java Virtual Machine Specification, 3rd edition [TypeNotPresentException](http://docs.google.com/java/lang/TypeNotPresentException.html) - if any of the generic superinterfaces refers to a non-existent type declaration MalformedParameterizedTypeException - if any of the generic superinterfaces refer to a parameterized type that cannot be instantiated for any reason**Since:** 1.5

### getComponentType

public [Class](http://docs.google.com/java/lang/Class.html)<?> **getComponentType**()

Returns the Class representing the component type of an array. If this class does not represent an array class this method returns null.

**Returns:**the Class representing the component type of this class if this class is an array**Since:** JDK1.1 **See Also:**[Array](http://docs.google.com/java/lang/reflect/Array.html)

### getModifiers

public int **getModifiers**()

Returns the Java language modifiers for this class or interface, encoded in an integer. The modifiers consist of the Java Virtual Machine's constants for public, protected, private, final, static, abstract and interface; they should be decoded using the methods of class Modifier.

If the underlying class is an array class, then its public, private and protected modifiers are the same as those of its component type. If this Class represents a primitive type or void, its public modifier is always true, and its protected and private modifiers are always false. If this object represents an array class, a primitive type or void, then its final modifier is always true and its interface modifier is always false. The values of its other modifiers are not determined by this specification.

The modifier encodings are defined in *The Java Virtual Machine Specification*, table 4.1.

**Returns:**the int representing the modifiers for this class**Since:** JDK1.1 **See Also:**[Modifier](http://docs.google.com/java/lang/reflect/Modifier.html)

### getSigners

public [Object](http://docs.google.com/java/lang/Object.html)[] **getSigners**()

Gets the signers of this class.

**Returns:**the signers of this class, or null if there are no signers. In particular, this method returns null if this object represents a primitive type or void.**Since:** JDK1.1

### getEnclosingMethod

public [Method](http://docs.google.com/java/lang/reflect/Method.html) **getEnclosingMethod**()

If this Class object represents a local or anonymous class within a method, returns a [Method](http://docs.google.com/java/lang/reflect/Method.html) object representing the immediately enclosing method of the underlying class. Returns null otherwise. In particular, this method returns null if the underlying class is a local or anonymous class immediately enclosed by a type declaration, instance initializer or static initializer.

**Returns:**the immediately enclosing method of the underlying class, if that class is a local or anonymous class; otherwise null.**Since:** 1.5

### getEnclosingConstructor

public [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<?> **getEnclosingConstructor**()

If this Class object represents a local or anonymous class within a constructor, returns a [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html) object representing the immediately enclosing constructor of the underlying class. Returns null otherwise. In particular, this method returns null if the underlying class is a local or anonymous class immediately enclosed by a type declaration, instance initializer or static initializer.

**Returns:**the immediately enclosing constructor of the underlying class, if that class is a local or anonymous class; otherwise null.**Since:** 1.5

### getDeclaringClass

public [Class](http://docs.google.com/java/lang/Class.html)<?> **getDeclaringClass**()

If the class or interface represented by this Class object is a member of another class, returns the Class object representing the class in which it was declared. This method returns null if this class or interface is not a member of any other class. If this Class object represents an array class, a primitive type, or void,then this method returns null.

**Returns:**the declaring class for this class**Since:** JDK1.1

### getEnclosingClass

public [Class](http://docs.google.com/java/lang/Class.html)<?> **getEnclosingClass**()

Returns the immediately enclosing class of the underlying class. If the underlying class is a top level class this method returns null.

**Returns:**the immediately enclosing class of the underlying class**Since:** 1.5

### getSimpleName

public [String](http://docs.google.com/java/lang/String.html) **getSimpleName**()

Returns the simple name of the underlying class as given in the source code. Returns an empty string if the underlying class is anonymous.

The simple name of an array is the simple name of the component type with "[]" appended. In particular the simple name of an array whose component type is anonymous is "[]".

**Returns:**the simple name of the underlying class**Since:** 1.5

### getCanonicalName

public [String](http://docs.google.com/java/lang/String.html) **getCanonicalName**()

Returns the canonical name of the underlying class as defined by the Java Language Specification. Returns null if the underlying class does not have a canonical name (i.e., if it is a local or anonymous class or an array whose component type does not have a canonical name).

**Returns:**the canonical name of the underlying class if it exists, and null otherwise.**Since:** 1.5

### isAnonymousClass

public boolean **isAnonymousClass**()

Returns true if and only if the underlying class is an anonymous class.

**Returns:**true if and only if this class is an anonymous class.**Since:** 1.5

### isLocalClass

public boolean **isLocalClass**()

Returns true if and only if the underlying class is a local class.

**Returns:**true if and only if this class is a local class.**Since:** 1.5

### isMemberClass

public boolean **isMemberClass**()

Returns true if and only if the underlying class is a member class.

**Returns:**true if and only if this class is a member class.**Since:** 1.5

### getClasses

public [Class](http://docs.google.com/java/lang/Class.html)<?>[] **getClasses**()

Returns an array containing Class objects representing all the public classes and interfaces that are members of the class represented by this Class object. This includes public class and interface members inherited from superclasses and public class and interface members declared by the class. This method returns an array of length 0 if this Class object has no public member classes or interfaces. This method also returns an array of length 0 if this Class object represents a primitive type, an array class, or void.

**Returns:**the array of Class objects representing the public members of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) method denies access to the classes within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getFields

public [Field](http://docs.google.com/java/lang/reflect/Field.html)[] **getFields**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array containing Field objects reflecting all the accessible public fields of the class or interface represented by this Class object. The elements in the array returned are not sorted and are not in any particular order. This method returns an array of length 0 if the class or interface has no accessible public fields, or if it represents an array class, a primitive type, or void.

Specifically, if this Class object represents a class, this method returns the public fields of this class and of all its superclasses. If this Class object represents an interface, this method returns the fields of this interface and of all its superinterfaces.

The implicit length field for array class is not reflected by this method. User code should use the methods of class Array to manipulate arrays.

See *The Java Language Specification*, sections 8.2 and 8.3.

**Returns:**the array of Field objects representing the public fields **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the fields within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getMethods

public [Method](http://docs.google.com/java/lang/reflect/Method.html)[] **getMethods**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array containing Method objects reflecting all the public *member* methods of the class or interface represented by this Class object, including those declared by the class or interface and those inherited from superclasses and superinterfaces. Array classes return all the (public) member methods inherited from the Object class. The elements in the array returned are not sorted and are not in any particular order. This method returns an array of length 0 if this Class object represents a class or interface that has no public member methods, or if this Class object represents a primitive type or void.

The class initialization method <clinit> is not included in the returned array. If the class declares multiple public member methods with the same parameter types, they are all included in the returned array.

See *The Java Language Specification*, sections 8.2 and 8.4.

**Returns:**the array of Method objects representing the public methods of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the methods within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getConstructors

public [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<?>[] **getConstructors**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array containing Constructor objects reflecting all the public constructors of the class represented by this Class object. An array of length 0 is returned if the class has no public constructors, or if the class is an array class, or if the class reflects a primitive type or void. Note that while this method returns an array of Constructor<T> objects (that is an array of constructors from this class), the return type of this method is Constructor<?>[] and *not* Constructor<T>[] as might be expected. This less informative return type is necessary since after being returned from this method, the array could be modified to hold Constructor objects for different classes, which would violate the type guarantees of Constructor<T>[].

**Returns:**the array of Constructor objects representing the public constructors of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the constructors within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getField

public [Field](http://docs.google.com/java/lang/reflect/Field.html) **getField**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NoSuchFieldException](http://docs.google.com/java/lang/NoSuchFieldException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns a Field object that reflects the specified public member field of the class or interface represented by this Class object. The name parameter is a String specifying the simple name of the desired field.

The field to be reflected is determined by the algorithm that follows. Let C be the class represented by this object:

1. If C declares a public field with the name specified, that is the field to be reflected.
2. If no field was found in step 1 above, this algorithm is applied recursively to each direct superinterface of C. The direct superinterfaces are searched in the order they were declared.
3. If no field was found in steps 1 and 2 above, and C has a superclass S, then this algorithm is invoked recursively upon S. If C has no superclass, then a NoSuchFieldException is thrown.

See *The Java Language Specification*, sections 8.2 and 8.3.

**Parameters:**name - the field name **Returns:**the Field object of this class specified by name **Throws:** [NoSuchFieldException](http://docs.google.com/java/lang/NoSuchFieldException.html) - if a field with the specified name is not found. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name is null [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the field
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getMethod

public [Method](http://docs.google.com/java/lang/reflect/Method.html) **getMethod**([String](http://docs.google.com/java/lang/String.html) name,  
 [Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)  
 throws [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns a Method object that reflects the specified public member method of the class or interface represented by this Class object. The name parameter is a String specifying the simple name of the desired method. The parameterTypes parameter is an array of Class objects that identify the method's formal parameter types, in declared order. If parameterTypes is null, it is treated as if it were an empty array.

If the name is "<init>;"or "<clinit>" a NoSuchMethodException is raised. Otherwise, the method to be reflected is determined by the algorithm that follows. Let C be the class represented by this object:

1. C is searched for any *matching methods*. If no matching method is found, the algorithm of step 1 is invoked recursively on the superclass of C.
2. If no method was found in step 1 above, the superinterfaces of C are searched for a matching method. If any such method is found, it is reflected.

To find a matching method in a class C:  If C declares exactly one public method with the specified name and exactly the same formal parameter types, that is the method reflected. If more than one such method is found in C, and one of these methods has a return type that is more specific than any of the others, that method is reflected; otherwise one of the methods is chosen arbitrarily.

Note that there may be more than one matching method in a class because while the Java language forbids a class to declare multiple methods with the same signature but different return types, the Java virtual machine does not. This increased flexibility in the virtual machine can be used to implement various language features. For example, covariant returns can be implemented with [bridge methods](http://docs.google.com/java/lang/reflect/Method.html#isBridge()); the bridge method and the method being overridden would have the same signature but different return types.

See *The Java Language Specification*, sections 8.2 and 8.4.

**Parameters:**name - the name of the methodparameterTypes - the list of parameters **Returns:**the Method object that matches the specified name and parameterTypes **Throws:** [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html) - if a matching method is not found or if the name is "<init>"or "<clinit>". [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name is null [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the method
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getConstructor

public [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<[T](http://docs.google.com/java/lang/Class.html)> **getConstructor**([Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)  
 throws [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns a Constructor object that reflects the specified public constructor of the class represented by this Class object. The parameterTypes parameter is an array of Class objects that identify the constructor's formal parameter types, in declared order. If this Class object represents an inner class declared in a non-static context, the formal parameter types include the explicit enclosing instance as the first parameter.

The constructor to reflect is the public constructor of the class represented by this Class object whose formal parameter types match those specified by parameterTypes.

**Parameters:**parameterTypes - the parameter array **Returns:**the Constructor object of the public constructor that matches the specified parameterTypes **Throws:** [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html) - if a matching method is not found. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.PUBLIC)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the constructor
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredClasses

public [Class](http://docs.google.com/java/lang/Class.html)<?>[] **getDeclaredClasses**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array of Class objects reflecting all the classes and interfaces declared as members of the class represented by this Class object. This includes public, protected, default (package) access, and private classes and interfaces declared by the class, but excludes inherited classes and interfaces. This method returns an array of length 0 if the class declares no classes or interfaces as members, or if this Class object represents a primitive type, an array class, or void.

**Returns:**the array of Class objects representing all the declared members of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared classes within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredFields

public [Field](http://docs.google.com/java/lang/reflect/Field.html)[] **getDeclaredFields**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array of Field objects reflecting all the fields declared by the class or interface represented by this Class object. This includes public, protected, default (package) access, and private fields, but excludes inherited fields. The elements in the array returned are not sorted and are not in any particular order. This method returns an array of length 0 if the class or interface declares no fields, or if this Class object represents a primitive type, an array class, or void.

See *The Java Language Specification*, sections 8.2 and 8.3.

**Returns:**the array of Field objects representing all the declared fields of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared fields within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredMethods

public [Method](http://docs.google.com/java/lang/reflect/Method.html)[] **getDeclaredMethods**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array of Method objects reflecting all the methods declared by the class or interface represented by this Class object. This includes public, protected, default (package) access, and private methods, but excludes inherited methods. The elements in the array returned are not sorted and are not in any particular order. This method returns an array of length 0 if the class or interface declares no methods, or if this Class object represents a primitive type, an array class, or void. The class initialization method <clinit> is not included in the returned array. If the class declares multiple public member methods with the same parameter types, they are all included in the returned array.

See *The Java Language Specification*, section 8.2.

**Returns:**the array of Method objects representing all the declared methods of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared methods within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredConstructors

public [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<?>[] **getDeclaredConstructors**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns an array of Constructor objects reflecting all the constructors declared by the class represented by this Class object. These are public, protected, default (package) access, and private constructors. The elements in the array returned are not sorted and are not in any particular order. If the class has a default constructor, it is included in the returned array. This method returns an array of length 0 if this Class object represents an interface, a primitive type, an array class, or void.

See *The Java Language Specification*, section 8.2.

**Returns:**the array of Constructor objects representing all the declared constructors of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared constructors within this class
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredField

public [Field](http://docs.google.com/java/lang/reflect/Field.html) **getDeclaredField**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NoSuchFieldException](http://docs.google.com/java/lang/NoSuchFieldException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns a Field object that reflects the specified declared field of the class or interface represented by this Class object. The name parameter is a String that specifies the simple name of the desired field. Note that this method will not reflect the length field of an array class.

**Parameters:**name - the name of the field **Returns:**the Field object for the specified field in this class **Throws:** [NoSuchFieldException](http://docs.google.com/java/lang/NoSuchFieldException.html) - if a field with the specified name is not found. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name is null [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared field
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredMethod

public [Method](http://docs.google.com/java/lang/reflect/Method.html) **getDeclaredMethod**([String](http://docs.google.com/java/lang/String.html) name,  
 [Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)  
 throws [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns a Method object that reflects the specified declared method of the class or interface represented by this Class object. The name parameter is a String that specifies the simple name of the desired method, and the parameterTypes parameter is an array of Class objects that identify the method's formal parameter types, in declared order. If more than one method with the same parameter types is declared in a class, and one of these methods has a return type that is more specific than any of the others, that method is returned; otherwise one of the methods is chosen arbitrarily. If the name is "<init>"or "<clinit>" a NoSuchMethodException is raised.

**Parameters:**name - the name of the methodparameterTypes - the parameter array **Returns:**the Method object for the method of this class matching the specified name and parameters **Throws:** [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html) - if a matching method is not found. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name is null [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared method
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getDeclaredConstructor

public [Constructor](http://docs.google.com/java/lang/reflect/Constructor.html)<[T](http://docs.google.com/java/lang/Class.html)> **getDeclaredConstructor**([Class](http://docs.google.com/java/lang/Class.html)<?>... parameterTypes)  
 throws [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Returns a Constructor object that reflects the specified constructor of the class or interface represented by this Class object. The parameterTypes parameter is an array of Class objects that identify the constructor's formal parameter types, in declared order. If this Class object represents an inner class declared in a non-static context, the formal parameter types include the explicit enclosing instance as the first parameter.

**Parameters:**parameterTypes - the parameter array **Returns:**The Constructor object for the constructor with the specified parameter list **Throws:** [NoSuchMethodException](http://docs.google.com/java/lang/NoSuchMethodException.html) - if a matching method is not found. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager, *s*, is present and any of the following conditions is met:

* invocation of [s.checkMemberAccess(this, Member.DECLARED)](http://docs.google.com/java/lang/SecurityManager.html#checkMemberAccess(java.lang.Class,%20int)) denies access to the declared constructor
* the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of [s.checkPackageAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPackageAccess(java.lang.String)) denies access to the package of this class

**Since:** JDK1.1

### getResourceAsStream

public [InputStream](http://docs.google.com/java/io/InputStream.html) **getResourceAsStream**([String](http://docs.google.com/java/lang/String.html) name)

Finds a resource with a given name. The rules for searching resources associated with a given class are implemented by the defining [class loader](http://docs.google.com/java/lang/ClassLoader.html) of the class. This method delegates to this object's class loader. If this object was loaded by the bootstrap class loader, the method delegates to [ClassLoader.getSystemResourceAsStream(java.lang.String)](http://docs.google.com/java/lang/ClassLoader.html#getSystemResourceAsStream(java.lang.String)).

Before delegation, an absolute resource name is constructed from the given resource name using this algorithm:

* If the name begins with a '/' ('\u002f'), then the absolute name of the resource is the portion of the name following the '/'.
* Otherwise, the absolute name is of the following form: modified\_package\_name/name  
     
  Where the modified\_package\_name is the package name of this object with '/' substituted for '.' ('\u002e').

**Parameters:**name - name of the desired resource **Returns:**A [InputStream](http://docs.google.com/java/io/InputStream.html) object or null if no resource with this name is found **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If name is null**Since:** JDK1.1

### getResource

public [URL](http://docs.google.com/java/net/URL.html) **getResource**([String](http://docs.google.com/java/lang/String.html) name)

Finds a resource with a given name. The rules for searching resources associated with a given class are implemented by the defining [class loader](http://docs.google.com/java/lang/ClassLoader.html) of the class. This method delegates to this object's class loader. If this object was loaded by the bootstrap class loader, the method delegates to [ClassLoader.getSystemResource(java.lang.String)](http://docs.google.com/java/lang/ClassLoader.html#getSystemResource(java.lang.String)).

Before delegation, an absolute resource name is constructed from the given resource name using this algorithm:

* If the name begins with a '/' ('\u002f'), then the absolute name of the resource is the portion of the name following the '/'.
* Otherwise, the absolute name is of the following form: modified\_package\_name/name  
     
  Where the modified\_package\_name is the package name of this object with '/' substituted for '.' ('\u002e').

**Parameters:**name - name of the desired resource **Returns:**A [URL](http://docs.google.com/java/net/URL.html) object or null if no resource with this name is found**Since:** JDK1.1

### getProtectionDomain

public [ProtectionDomain](http://docs.google.com/java/security/ProtectionDomain.html) **getProtectionDomain**()

Returns the ProtectionDomain of this class. If there is a security manager installed, this method first calls the security manager's checkPermission method with a RuntimePermission("getProtectionDomain") permission to ensure it's ok to get the ProtectionDomain.

**Returns:**the ProtectionDomain of this class **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkPermission method doesn't allow getting the ProtectionDomain.**Since:** 1.2 **See Also:**[ProtectionDomain](http://docs.google.com/java/security/ProtectionDomain.html), [SecurityManager.checkPermission(java.security.Permission)](http://docs.google.com/java/lang/SecurityManager.html#checkPermission(java.security.Permission)), [RuntimePermission](http://docs.google.com/java/lang/RuntimePermission.html)

### desiredAssertionStatus

public boolean **desiredAssertionStatus**()

Returns the assertion status that would be assigned to this class if it were to be initialized at the time this method is invoked. If this class has had its assertion status set, the most recent setting will be returned; otherwise, if any package default assertion status pertains to this class, the most recent setting for the most specific pertinent package default assertion status is returned; otherwise, if this class is not a system class (i.e., it has a class loader) its class loader's default assertion status is returned; otherwise, the system class default assertion status is returned.

Few programmers will have any need for this method; it is provided for the benefit of the JRE itself. (It allows a class to determine at the time that it is initialized whether assertions should be enabled.) Note that this method is not guaranteed to return the actual assertion status that was (or will be) associated with the specified class when it was (or will be) initialized.

**Returns:**the desired assertion status of the specified class.**Since:** 1.4 **See Also:**[ClassLoader.setClassAssertionStatus(java.lang.String, boolean)](http://docs.google.com/java/lang/ClassLoader.html#setClassAssertionStatus(java.lang.String,%20boolean)), [ClassLoader.setPackageAssertionStatus(java.lang.String, boolean)](http://docs.google.com/java/lang/ClassLoader.html#setPackageAssertionStatus(java.lang.String,%20boolean)), [ClassLoader.setDefaultAssertionStatus(boolean)](http://docs.google.com/java/lang/ClassLoader.html#setDefaultAssertionStatus(boolean))

### isEnum

public boolean **isEnum**()

Returns true if and only if this class was declared as an enum in the source code.

**Returns:**true if and only if this class was declared as an enum in the source code**Since:** 1.5

### getEnumConstants

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

Returns the elements of this enum class or null if this Class object does not represent an enum type.

**Returns:**an array containing the values comprising the enum class represented by this Class object in the order they're declared, or null if this Class object does not represent an enum type**Since:** 1.5

### cast

public [T](http://docs.google.com/java/lang/Class.html) **cast**([Object](http://docs.google.com/java/lang/Object.html) obj)

Casts an object to the class or interface represented by this Class object.

**Parameters:**obj - the object to be cast **Returns:**the object after casting, or null if obj is null **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the object is not null and is not assignable to the type T.**Since:** 1.5

### asSubclass

public <U> [Class](http://docs.google.com/java/lang/Class.html)<? extends U> **asSubclass**([Class](http://docs.google.com/java/lang/Class.html)<U> clazz)

Casts this Class object to represent a subclass of the class represented by the specified class object. Checks that that the cast is valid, and throws a ClassCastException if it is not. If this method succeeds, it always returns a reference to this class object.

This method is useful when a client needs to "narrow" the type of a Class object to pass it to an API that restricts the Class objects that it is willing to accept. A cast would generate a compile-time warning, as the correctness of the cast could not be checked at runtime (because generic types are implemented by erasure).

**Returns:**this Class object, cast to represent a subclass of the specified class object. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if this Class object does not represent a subclass of the specified class (here "subclass" includes the class itself).**Since:** 1.5

### getAnnotation

public <A extends [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)> A **getAnnotation**([Class](http://docs.google.com/java/lang/Class.html)<A> annotationClass)

**Description copied from interface:** [**AnnotatedElement**](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#getAnnotation(java.lang.Class)) Returns this element's annotation for the specified type if such an annotation is present, else null.

**Specified by:**[getAnnotation](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#getAnnotation(java.lang.Class)) in interface [AnnotatedElement](http://docs.google.com/java/lang/reflect/AnnotatedElement.html) **Parameters:**annotationClass - the Class object corresponding to the annotation type **Returns:**this element's annotation for the specified annotation type if present on this element, else null **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the given annotation class is null**Since:** 1.5

### isAnnotationPresent

public boolean **isAnnotationPresent**([Class](http://docs.google.com/java/lang/Class.html)<? extends [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)> annotationClass)

**Description copied from interface:** [**AnnotatedElement**](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#isAnnotationPresent(java.lang.Class)) Returns true if an annotation for the specified type is present on this element, else false. This method is designed primarily for convenient access to marker annotations.

**Specified by:**[isAnnotationPresent](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#isAnnotationPresent(java.lang.Class)) in interface [AnnotatedElement](http://docs.google.com/java/lang/reflect/AnnotatedElement.html) **Parameters:**annotationClass - the Class object corresponding to the annotation type **Returns:**true if an annotation for the specified annotation type is present on this element, else false **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the given annotation class is null**Since:** 1.5

### getAnnotations

public [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)[] **getAnnotations**()

**Description copied from interface:** [**AnnotatedElement**](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#getAnnotations()) Returns all annotations present on this element. (Returns an array of length zero if this element has no annotations.) The caller of this method is free to modify the returned array; it will have no effect on the arrays returned to other callers.

**Specified by:**[getAnnotations](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#getAnnotations()) in interface [AnnotatedElement](http://docs.google.com/java/lang/reflect/AnnotatedElement.html) **Returns:**all annotations present on this element**Since:** 1.5

### getDeclaredAnnotations

public [Annotation](http://docs.google.com/java/lang/annotation/Annotation.html)[] **getDeclaredAnnotations**()

**Description copied from interface:** [**AnnotatedElement**](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#getDeclaredAnnotations()) Returns all annotations that are directly present on this element. Unlike the other methods in this interface, this method ignores inherited annotations. (Returns an array of length zero if no annotations are directly present on this element.) The caller of this method is free to modify the returned array; it will have no effect on the arrays returned to other callers.

**Specified by:**[getDeclaredAnnotations](http://docs.google.com/java/lang/reflect/AnnotatedElement.html#getDeclaredAnnotations()) in interface [AnnotatedElement](http://docs.google.com/java/lang/reflect/AnnotatedElement.html) **Returns:**All annotations directly present on this element**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Class.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/CharSequence.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/ClassCastException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Class.html)    [**NO FRAMES**](http://docs.google.com/Class.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[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).