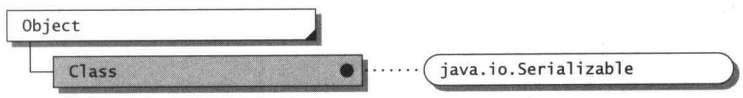
java.lang
Class



语法

public final class Class implements Serializable

描述

Class类提供了检索关于某个类的信息的方法，以及一个创建类实例的方法。在Java中，每一个对象都是某个类的一个实例。Java为每个类都保存了一个固定的Class对象，其中包含了该对象的相关信息。我们说这个Class对象代表或反映这个类。这个Class对象包含的信息有：这个类的名字（用字符串表示），该类的超类，它所实现的接口，以及用于加载该类的加载器。

Class对象也提供关于接口的信息，尽管一个接口与一个类有所不同。当谈论一个Class对象时，我们在明白Class对象实际上可能反映的不是一个类而是一个接口的情况下，不严格地把它叫做某某类。

有关这个类的的详细信息见《The Java Class Libraries，Seconds Edition，Volume 1》。

资源名称

资源是指一些被程序所使用的数据，例如，图像、音频文件或一个HTML文件。在典型的情况下，资源被包装成一个JAR文件，与使用它们的类文件存储在一起。Class包含了getResource()和getResourceAsStream()两个方法，调用它们可以通过传递一个资源名称获取一个资源。如果所传递的资源名称不包含反斜杠（“/”），那么，用于定位资源的资源名称是以一个类所在包的包名作为前缀，所有的点字符（“.”）都由反斜杠所代替。例如，调用在类p.C上的getResource(“index.html”)是在请求资源“p/index.html”。如果传递的资源名称包含有反斜杠，那么，实际上所请求的资源是名称中并没有反斜杠的资源。例如，调用类p.C上的getResource(“/index.html”)，实际上是在请求资源“index.html”。

被用于getResource()和getResourceAsStream()方法的把一个资源名映射到一个资源的算法在《The Java Class Libraries，Seconds Edition，Volume 1》中有所描述。注意：该算法没有被用于ClassLoader中与资源有关的方法中。

版本1.2中所作的修改

一个新的forName()方法的重载形式允许你说明在获取一个类的Class对象时所使用的加载器。这对于希望通过另一个实体的行为对类进行加载的库代码来说是有用的。

有一个新类，Package，代表一个包。现在可以使用getPackage()方法获取一个类所在的包对象。

在版本1.2的安全模式中，每一个类都与一个保护域相联系，这个保护域说明了这个类的

代码源——这个类从何处被加载，谁进行的授权——以及对这个代码源的许可。可以通过使用getProtectionDomain()方法询问类的保护域。如果想了解新的安全模式请参见SecurityManager。

在版本1.1中，方法getClasses()、getDeclaredClasses()和getDeclaringClass()没有被实现。在版本1.2中它们得到了实现。关于它们的描述请见附录。

	成员概述	
	类检索方法	
Δ	forName()	检索由类的类型描述器所说明的Class对象。
Δ	getClasses()	检索所有公共的类与接口成员的Class对象数组。
Δ	getDeclaredClasses()	检索所有类与接口成员的Class对象数组。
	getSuperclass()	检索该Class对象扩展的超类。
	创建实例的方法	
	newInstance()	创建一个由该Class对象表示的类的实例。
	资源获取方法	
	getResource()	获取一个与由该Class对象所表示的类相关的资源。
	getResourceAsStream()	获取并创建一个该Class对象所表示的类的一个资源的输入流。
	对象类型查询方法	
	isArray()	判断这个Class对象是否表示一个数组类型。
	isAsignableFrom()	判断由这个Class对象所代表的类是否是一个类的超类。
	isInstance()	判断一个对象是否是由这个Class对象所表示的类的一个实例。
	isInterface()	判断这个Class对象是否代表一个接口。
	isPrimitive()	判断这个Class对象是否代表一个Java基本类型。
	获取名字与包的方法	
	getName()	检索这个Class对象的类型描述器。
1.2	getPackage()	检索这个Class对象的Package对象。
	映像方法	
	getComponentType()	检索一个数组的成员类型。
	getConstructor()	检索代表一个公共构造函数的Constructor对象。
	getConstructors()	检索代表所有公共构造函数的Constructor对象。
	getDeclaredConstructor()	检索代表一个构造函数的Constructor对象。
	getDeclaredConstructors()	检索代表所有构造函数的Constructor对象。
	getDeclaredField()	检索代表一个已声明的域的Fileld对象。
	getDeclaredFields()	检索代表所有已声明的域的Fileld对象。
	getDeclaredMethod()	检索代表一个已声明的方法的Method对象。
	getDeclaredMethods()	检索代表所有已声明的方法的Method对象。
Δ	getDeclaringClass()	检索该Class对象所代表的正在声明的类。
	getField()	检索代表一个可访问的公共域的Field对象。
	getFields()	检索代表所有可访问的公共域的Field对象。

(续)

成员概述	
getInterfaces()	检索由这个Class对象实现的接口。
getMethod()	检索代表一个可访问的公共方法的Method对象。
getMethods()	检索代表所有可访问的公共方法的Method对象。
getModifiers()	检索代表该Class对象所代表的类的Java语言修改器。
安全方法	
1.2 getProtectionDomain()	检索这个Class对象的ProtectionDomain。
getSigners()	检索这个Class的签名人。
类加载器方法	
getClassLoader()	检索这个Class所的类加载器。
对象方法	
toString()	产生一个代表该Class对象的字符串。

参见

ClassLoader.loadClass()、Package、SecurityManager。
《The Java Class Libraries，Second Edition，Volume 1》中的Class类。

▲ forName()

目的 检索由类的类型描述器指定的Class对象。

语法
public static Class forName(String className) throws ClassNotFoundException

从版本1.2
public static Class forName(String className， boolean init， ClassLoader loader) throws ClassNotFoundException

描述 该方法用于检索由className的类型描述器指定以classname说明的Class对象。className是由getName()返回的字符串。有关Class对象的类型描述器的详细内容见《The Java Class Libraries，Second Edition，Volume 1》中关于Class对象的描述器部分。

该方法不能被用于返回基本类型的Class对象。要想返回基本类型的Class对象请参考《The Java Class Libraries，Second Edition，Volume 1》中关于这个类的描述器部分。

该方法的第一种形式用于获取由className所说明的Class对象。如果这个类还没有被加载，该形式将使用这个正在被调用方法的类加载器在不对这个类进行初始化的情况下对这个类进行加载。

该方法的第二种形式也用于获取由className所说明的Class对象。如果这个类还没有被加载，该形式将使用loader对象对其进行加载。如果被请求的类的一个实例已经被创建或被请求类的方法已被唤醒，init将被设为true。如果那个被说明的类被加载只是为了检查它是否存在或为了得到它的超类，init将被设为false。参数init被Java虚拟机用于完善它的类加载机制。它将不会对将来加载的类产生任何影响。例如，当一个类被加载时init已被设为false，

那么，程序对这个类第一次请求时， Java虚拟机将创建该类的一个实例，并会自动连接并初始化这个类。如果 loader为null，这个正在调用的方法的类加载器将会被使用。

版本1.2中的改动 在版本1.2中，该方法的第二种形式是新增的。在版本 1.1中，如果 className的语法非法， 那么 forName()将抛出一个IllegalArgumentException异常。在版本1.2中，如果className的语法非法，那么forName()将抛出一个ClassNotFoundException异常。

参数

className	Class对象的一个非空类型描述器（例如，“ java.lang.String ”）。
init	如果为true，则对className所描述的类进行链接和初始化，以使其准备创建新的实例；如果为false，则不进行初始化。
loader	一个可能被使用的可能为null的类加载器。如果为null，则使用调用者的类加载器。

返回 由className所表示的一个非空Class对象。

异常

ClassNotFoundException	如果加载器没有找到要加载的类。
ExceptionInInitializerError	如果对要加载的类的静态初始化失败。
LinkageError	如果因为错误类不能被链接上。
SecurityException	如果调用者没有获得RuntimePermission(“ getClassLoader ”)。

参见 ClassLoader.loadClass()，getName()，RuntimePermission(“ getClassLoader ”)，Thread.getContextClassLoader()。

示例 请见《The Java Class Libraries，Second Edition，Volume 1》中的使用该方法的第一种形式的示例。请见 getProtectionDomain()和Thread.getContextClassLoader()中使用该方法第二种形式的示例。

∧ getClasses()

目的 检索包含所有的公共类和接口成员的Class对象数组。

语法 public Class[] getClasses()

描述 此方法检索包含所有公共类和由这个 Class对象表示的类成员接口的Class对象数组。它包括从超类继承得到的公共类和接口成员，及类声明的公共类和接口成员。

版本1.2中的改动 在版本1.1中没有实现这个方法，总是返回空数组。

返回 非空的Class对象数组。

异常

SecurityException	当调用者没有访问成员信息的权限时。

参见 getDeclaredClasses()、SecurityManager.checkMemberAccess()。

示例

A.java

```
public interface A {
  public class B {
  }

  interface C {
  }
}
```

D.java

```
public class D implements A {
  public class E {
  }
}
```

F.java

```
public class F extends D {
  public class G {
  }

  class H {
  }
}
```

Main.java

```
public class Main {
  public static void main(String[] args) {

    printClasses("D.getClasses", D.class.getClasses()); // D$E
    printClasses("D.getDeclaredClasses", D.class.getDeclaredClasses());
                                                        // D$E

    printClasses("A.getClasses", A.class.getClasses()); // A$B A$C
    printClasses("A.getDeclaredClasses", A.class.getDeclaredClasses());
                                                        // A$B A$C

    printClasses("F.getClasses", F.class.getClasses()); // F$G D$E
    printClasses("F.getDeclaredClasses", F.class.getDeclaredClasses());
                                                        // F$G F$H

    System.out.println(A.class.getDeclaringClass());   // null
    System.out.println(A.B.class.getDeclaringClass()); // interface A
    System.out.println(D.E.class.getDeclaringClass()); // class D
    System.out.println(F.E.class.getDeclaringClass()); // class D
  }

  public static void printClasses(String msg, Class[] cls) {
    System.out.print(msg + ": ");
    for (int i = 0; cls != null && i < cls.length; i++) {
      System.out.print(cls[i].getName() + " ");
    }
    System.out.println();
  }
}
```

输出

```
D.getClasses: D$E
D.getDeclaredClasses: D$E
A.getClasses: A$B A$C
A.getDeclaredClasses: A$B A$C
F.getClasses: F$G D$E
F.getDeclaredClasses: F$G F$H
```

```
null
interface A
class D
class D
```

getDeclaredClasses()

目的	检索包含所有类和接口成员的Class对象数组。
语法	public Class[] getDeclaredClasses() throws SecurityException
描述	此方法检索包含所有类和接口的Class对象数组。在这里接口被声明为由这个Class对象表示的成员。
版本1.2中的改动	在版本1.1中没有实现这个方法，总是返回空数组。
返回	Class对象的非空数组。
参见	getClasses()、getDeclaringClass、SecurityManager. checkMemberAccess()。
异常	
SecurityException	
	当访问信息被拒绝时。
示例	参见getClasses()。

getDeclaringClass()

目的	检索此Class对象的声明类。
语法	public Class getDeclaringClass()
描述	如果此Class对象表示的接口是另一个类的成员，此方法检索代表声明类(类或方法是它的成员)的Class对象。当此类或接口不是其他任何类的成员时，返回null。
版本1.2中的改动	在版本1.1中没有实现这个方法,总是返回null。
返回	可能为null的Class对象。
参见	getClass()。
示例	参见getClasses()。

1.2 getPackage()

目的	获取与该Class对象的Package对象。
语法	public Package getPackage()
描述	每一个类加载器都保留了一个有关它所加载的类所在包的列表。该方法使用这个Class的加载器（getClassLoader()）来检索它所代表的类的Package对象。
返回	这个Class对象所对应的Package对象，可能为null。如果为null，那么就表示在加载这个类的加载器中没有关于这个类所在包的信息。
参见	ClassLoader.getPackage()，ClassLoader.getPackages()，getClassLoader()，Package。
示例	见ClassLoader类中Main.java的示例。

1.2 getProtectionDomain()

目的	获取该类的ProtectionDomain对象。

语法 public ProtectionDomain getProtectionDomain()

描述 该方法用于获取这个类的保护域。当一个类被加载，它的类加载器就登记下了这个类的保护域。保护域说明了这个类的代码源（这个类从哪里被加载的，以及签署代码的证明）还有Java运行安全策略对这个代码源的权限。该方法返回与这个类相对应的保护域。如果对这个类的保护域没有记载，那么，该方法返回的保护域对象将包含一个null代码源（代码下载的地点以及签署代码的证明）以及所有的授权。

调用者在使用该方法时必须有RuntimePermission（“getProtectionDomain”）；否则将抛出一个SecurityException异常。

返回 这个类的一个非空ProtectionDomain。

异常

SecurityException

如果有一个安全管理器，并且它不允许调用者得到 Permission Domain时，该异常将被抛出。

参见 java.security.ProtectionDomain、RuntimePermission(“ get ProtectionDomain ”)。

示例

```
try {
    Class c = Class.forName("ShowFiles", true,
        ClassLoader.getSystemClassLoader());

    ProtectionDomain pd = c.getProtectionDomain();
    System.out.println("code source: " + pd.getCodeSource());
    System.out.println("permissions: " + pd.getPermissions());
} catch (ClassNotFoundException e) {
    System.out.println(e);
}
```

输出

```
# java -Djava.security.policy=.policy -cp showfiles.jar\:. Main > o.txt
code source: (file:/tmp/lang/Class/getProtectionDomain/showfiles.jar [
[
  Version: V1
  Subject: CN=Duke, OU=JavaSoft, O=Sun, L=Cupertino, ST=CA, C=US
  Signature Algorithm: SHA1withDSA, OID = 1.2.840.10040.4.3

  Key:  Sun DSA Public Key
    Parameters:DSA
        p:     fd7f5381 1d751229 52df4a9c 2eece4e7 f611b752 3cef4400 c31e3
b6512669
    455d4022 51fb593d 8d58fabf c5f5ba30 f6cb9b55 6cd7813b 801d346f f26660b
    6b9950a5 a49f9fe8 047b1022 c24fbba9 d7feb7c6 1bf83b57 e7c6a8a6 150f04f
    83f6d3c5 1ec30235 54135a16 9132f675 f3ae2b61 d72aeff2 2203199d d14801c
        q:     9760508f 15230bcc b292b982 a2eb840b f0581cf5
        g:     f7e1a085 d69b3dde cbbcab5c 36b857b9 7994afbb fa3aea82 f9574
3d078267
    5159578e bad4594f e6710710 8180b449 167123e8 4c281613 b7cf0932 8cc8a6e
    3c167a8b 547c8d28 e0a3ae1e 2bb3a675 916ea37f 0bfa2135 62f1fb62 7a01243
    cca4f1be a8519089 a883dfe1 5ae59f06 928b665e 807b5525 64014c3b fecf492

  y:
```

```
    e5ca7e17 db82e902 c62235fe 8c053101 6e59a7d8 997b06db 7feafed2 b39c96b9
    1e26019c 8c67597a a101e063 bd3a5435 3bdec61a eab09bef a2d6d46d ace66710
    967357a9 6fbfc158 5a214078 cb69cb39 83bb69c2 189651bd 48b343d3 e6d3cb22
    7497bc31 4f2cb08d 840ac659 01c05794 64947c5b 1df8a931 f28283e6 00850e6e

  Validity: [From: Fri Sep 04 12:10:13 GMT-07:00 1998,
              To: Tue Jan 20 11:10:13 GMT-08:00 2026]
  Issuer: CN=Duke, OU=JavaSoft, O=Sun, L=Cupertino, ST=CA, C=US
  SerialNumber: [    35f03b15 ]

]
  Algorithm: [SHA1withDSA]
  Signature:
0000: 30 2C 02 14 6F D1 AD 78   B3 80 C6 7E 65 BB 94 52  0,..o..x....e..R
0010: 9F F7 CC 37 B2 49 E3 89   02 14 35 53 73 FF 75 41  ...7.I....5Ss.uA
0020: D5 65 DA AF 63 74 41 A5   FF CB 73 4B ED 41        .e..ctA...sK.A

])
permissions: java.security.Permissions@60a7101c (
 (java.net.SocketPermission localhost:1024- listen,resolve)
 (java.util.PropertyPermission java.specification.name read)
 (java.util.PropertyPermission java.version read)
 (java.util.PropertyPermission java.specification.version read)
 (java.util.PropertyPermission java.vm.vendor read)
 (java.util.PropertyPermission java.vm.specification.version read)
 (java.util.PropertyPermission os.arch read)
 (java.util.PropertyPermission java.vendor.url read)
 (java.util.PropertyPermission line.separator read)
 (java.util.PropertyPermission os.name read)
 (java.util.PropertyPermission java.vendor read)
 (java.util.PropertyPermission java.vm.specification.vendor read)
 (java.util.PropertyPermission java.specification.vendor read)
 (java.util.PropertyPermission java.vm.name read)
 (java.util.PropertyPermission java.vm.specification.name read)
 (java.util.PropertyPermission java.class.version read)
 (java.util.PropertyPermission os.version read)
 (java.util.PropertyPermission java.vm.version read)
 (java.util.PropertyPermission path.separator read)
 (java.util.PropertyPermission file.separator read)
 (java.lang.RuntimePermission stopThread )
 (java.lang.RuntimePermission exitVM )
 (java.io.FilePermission /tmp/lang/Class/getProtectionDomain/showfiles.jar
read)
 (java.io.FilePermission /tmp/lang/Class/getProtectionDomain/- read)
)
```