Class java.lang.ClassLoader

```
java.lang.Object
   |
   +----java.lang.ClassLoader
```

public class **ClassLoader**
extends Object

ClassLoader is an abstract Class that can be used to define a policy for loading Java classes into the runtime environment. By default, the runtime system loads classes that originate as files by reading them from the directory defined by the `CLASSPATH` environment variable (this is platform dependent). The default mechanism does not involve a Class loader.

However, some classes may not originate from a file; they could be loaded from some other source, e.g., the network. Classes loaded from the network are an array of bytes. A ClassLoader can be used to tell the runtime system to convert an array of bytes into an instance of class Class. This conversion information is passed to the runtime using the defineClass() method.

Classes that are created through the defineClass() mechanism can reference other classes by name. To resolve those names, the runtime system calls the ClassLoader that originally created the Class. The runtime system calls the abstract method loadClass() to load the referenced classes.

```
ClassLoader loader = new NetworkClassLoader(host, port);
Object main = loader.loadClass("Main").newInstance();
....
```

The NetworkClassLoader subclass must define the method loadClass() to load a Class from the network. Once it has downloaded the bytes that make up the Class it should use the method defineClass() to create a Class instance. A sample implementation could be:

```
class NetworkClassLoader {
    String host;
    int port;
    Hashtable cache = new Hashtable();
    private byte loadClassData(String name)[] {
        // load the class data from the connection
        ...
    }
    public synchronized Class loadClass(String name) {
        Class c = cache.get(name);
        if (c == null) {
            byte data[] = loadClassData(name);
```

```
                cache.put(name, defineClass(data, 0, data.length));
            }
            return c;
        }
    }
```

See Also:
Class
Version:
1.27, 08/21/95
Author:
Arthur van Hoff

Constructor Index

o **ClassLoader**()
Constructs a new Class loader and initializes it.

Method Index

o **defineClass**(byte[], int, int)
Converts an array of bytes to an instance of class Class.
o **findSystemClass**(String)
Loads a system Class.
o **loadClass**(String, boolean)
Resolves the specified name to a Class.
o **resolveClass**(Class)
Resolves classes referenced by this Class.

Constructors

o **ClassLoader**

```
protected ClassLoader()
```

Constructs a new Class loader and initializes it.

Methods

o **loadClass**

```
protected abstract Class loadClass(String name,
                                   boolean resolve) throws ClassNotFoundException
```

Resolves the specified name to a Class. The method loadClass() is called by the virtual

machine. As an abstract method, loadClass() must be defined in a subclass of ClassLoader. By using a Hashtable, you can avoid loading the same Class more than once.

Parameters:

name – the name of the desired Class

resolve – true if the Class needs to be resolved

Returns:

the resulting Class, or null if it was not found.

See Also:

Hashtable

o **defineClass**

```
protected final Class defineClass(byte data[],
                                  int offset,
                                  int length)
```

Converts an array of bytes to an instance of class Class. Before the Class can be used it must be resolved.

Parameters:

data – the bytes that make up the Class

offset – the start offset of the Class data

length – the length of the Class data

Returns:

the Class object which was created from the data.

Throws: ClassFormatError

If the data does not contain a valid Class.

See Also:

loadClass, resolveClass

o **resolveClass**

```
protected final void resolveClass(Class c)
```

Resolves classes referenced by this Class. This must be done before the Class can be used. Class names referenced by the resulting Class are resolved by calling loadClass().

Parameters:

c – the Class to be resolved

See Also:

defineClass

o **findSystemClass**

```
protected final Class findSystemClass(String name) throws ClassNotFoundException
```

Loads a system Class. A system Class is a class with the primordial Class loader (which is null).

Parameters:

name – the name of the system Class

Throws: NoClassDefFoundError

If the Class is not found.