# 类加载

将类的class文件读入内存，并为之创建一个java.lang.Class对象，也就是说当程序中使用任何类时，系统会为之建立一个java.lang.Class对象。类也是一种对象。

类的加载由类加载器完成，类加载器通常由JVM提供，这些类加载器是所有程序运行的基础，JVM提供的这些类加载器通常称为系统类加载器。除此之外，开发者可以通过继承ClassLoader基类来创建自己的类加载器。通过使用不同的类加载器，可以从不同来源加载类的二进制数据。来源的方式：

（1）从本地文件系统加载class文件。

（2）从jar包中加载class文件。

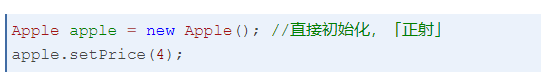
（3）通过网络加载class文件。

（4）把一个java源文件动态编译，并执行加载。

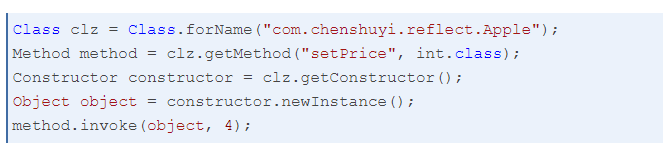
当类被加载之后，系统为之生成一个对应的Class对象，接着进入连接阶段，连接阶段负责把类的二进制数据合并到JRE中。

# 基本概念

一般情况下，我们使用某个类时必定知道它是什么类，是用来做什么的。于是我们直接对这个类进行实例化，之后使用这个类对象进行操作。



而反射则是一开始并不知道我要初始化的类对象是什么，自然也无法使用 new 关键字来创建对象了。这时候，我们使用 JDK 提供的反射 API 进行反射调用：



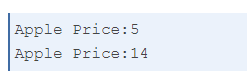
上面两段代码的执行结果，其实是完全一样的。但是其思路完全不一样，第一段代码在未运行时就已经确定了要运行的类（Apple），而第二段代码则是在运行时通过字符串值才得知要运行的类（com.chenshuyi.reflect.Apple）

**反射就是在运行时才知道要操作的类是什么，并且可以在运行时获取类的完整构造，并调用对应的方法。**

# ****简单示例****



从代码中可以看到我们使用反射调用了 setPrice 方法，并传递了 14 的值。之后使用反射调用了 getPrice 方法，输出其价格。上面的代码整个的输出结果是：



从这个简单的例子可以看出，一般情况下我们使用反射获取一个对象的步骤：

（1）获取类的class对象实例

Class clz = Class.forName("com.zhenai.api.Apple");

（2）根据 Class 对象实例获取 Constructor 对象

Constructor appleConstructor = clz.getConstructor();

（3）使用 Constructor 对象的 newInstance 方法获取反射类对象

Object appleObj = appleConstructor.newInstance();

而如果要调用某一个方法，则需要经过下面的步骤：

（4）获取方法的 Method 对象

Method setPriceMethod = clz.getMethod("setPrice", int.class);

（5）利用 invoke 方法调用方法

setPriceMethod.invoke(appleObj, 14);

在 JDK 中，反射相关的 API 可以分为下面几个方面：获取反射的 Class 对象、通过反射创建类对象、通过反射获取类属性方法及构造器。

# 反射常用的API

## 1、获取反射中的Class对象

在反射中，要获取一个类或调用一个类的方法，我们首先需要获取到该类的 Class 对象。

在Java API中，获取 Class 类对象有三种方法：

**第一种，使用 Class.forName 静态方法。**当你知道该类的全路径名时，你可以使用该方法获取 Class 类对象。

Class clz = Class.forName("java.lang.String");

**第二种，使用 .class 方法。**

这种方法只适合在编译前就知道操作的 Class。

Class clz = String.class;

**第三种，使用类对象的 getClass() 方法。**

String str = new String("Hello");

Class clz = str.getClass();

## 2、通过反射创建类对象

通过反射创建类对象主要有两种方式：通过 Class 对象的 newInstance() 方法、通过 Constructor 对象的 newInstance() 方法。

第一种：通过 Class 对象的 newInstance() 方法。

Class clz = Apple.class;

Apple apple = (Apple)clz.newInstance();

第二种：通过 Constructor 对象的 newInstance() 方法

Class clz = Apple.class;

Constructor constructor = clz.getConstructor();

Apple apple = (Apple)constructor.newInstance();

通过 Constructor 对象创建类对象可以选择特定构造方法，而通过 Class 对象则只能使用默认的无参数构造方法。下面的代码就调用了一个有参数的构造方法进行了类对象的初始化。

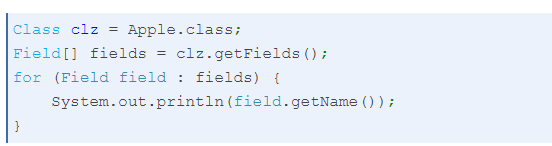
Class clz = Apple.class;

Constructor constructor = clz.getConstructor(String.class, int.class);

Apple apple = (Apple)constructor.newInstance("红富士", 15);

## 3、通过反射获取类属性、方法、构造器

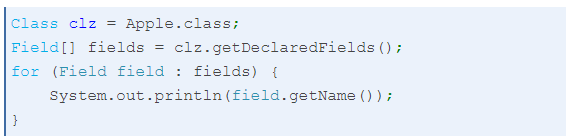
我们通过 Class 对象的 getFields() 方法可以获取 Class 类的属性，但无法获取私有属性。



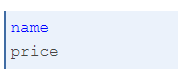
输出结果：



而如果使用 Class 对象的 getDeclaredFields() 方法则可以获取包括私有属性在内的所有属性：



输出结果：



与获取类属性一样，当我们去获取类方法、类构造器时，如果要获取私有方法或私有构造器，则必须使用有 declared 关键字的方法。

# 反射源码解析

https://www.cnblogs.com/chanshuyi/p/head\_first\_of\_reflection.html#top