# Class 类

## Class类

(1)Class是一个类，一个类在 JVM 中只会有一个Class实例

(2)对象照镜子后可以得到的信息：

某个类的数据成员名、方法和构造器、某个类到底实现了哪些接口。

(3)对于每个类而言，JRE 都为其保留一个不变的 Class 类型的对象。

一个 Class 对象包含了特定某个类的有关信息。

(4)Class 对象只能由系统建立对象

(5)每个类的实例都会记得自己是由哪个 Class 实例所生成

## 获取 Class 对象的4种方式

(1)类名.class

clazz = Person.**class**;

(2)通过对象调用getClass()方法来获取

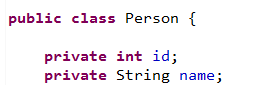
Object obj = **new** Person();

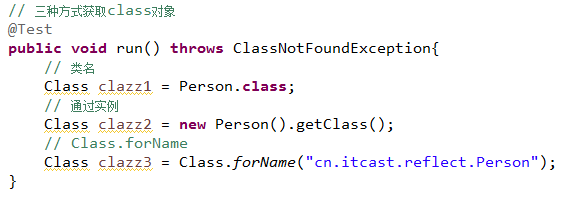
clazz = obj.getClass();

\*\*\*(3)常用--通过全类名的方式获取,要抛出throws ClassNotFoundException

String className = "qwe.Person";

clazz = Class.*forName*(className);





（4）类加载器

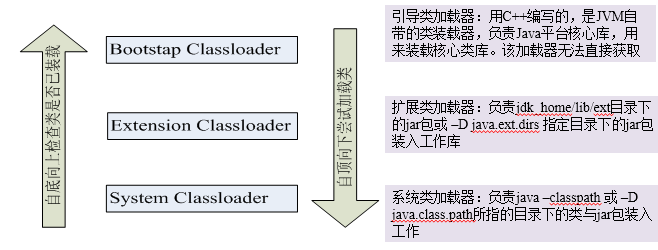
## Class类的常用方法



# ClassLoader类加载器

## 概念

启动类装载器(bootstrap) 用户自定义装载器(user-defined class loader)。



## 演示类加载机制的层次关系

.getSystemClassLoader() .getParent()

(1)获取一个系统缺省的ClassLoader类加载器

public class ClassLoaderDemo {

public static void main(String[] args) {

ClassLoader classloader;

**(1)获取一个系统缺省类加载器**

classloader = ClassLoader.getSystemClassLoader();

System.out.println(classloader);

while (classloader != null) {

**(2) )获取一个系统类加载器的父类加载器**

classloader = classloader.getParent();

System.out.println(classloader);

}

**//表示系统类装载器parent的parent为bootstrap，无法直接获取**

try {

(**3)Object类由bootstrap装载的，无法直接获取**

classloader = Class.forName("java.lang.Object").getClassLoader();

System.out.println("java.lang.Object's loader is " + classloader);

(4)**表示用户类是由系统类装载器装载的**

cl = Class.forName("com.atguigu.javase.reflection.ClassLoaderDemo");

classloader = cl.getClassLoader();

System.out.println("ClassLoaderDemo's loader is " + classloader);

} catch (Exception e) {

}

}

}

**执行结果如下：**

//表示系统类装载器实例化自类sun.misc.Launcher$AppClassLoader

sun.misc.Launcher$AppClassLoader@19821f

//表示系统类装载器的parent实例化自类sun.misc.Launcher$ExtClassLoader

sun.misc.Launcher$ExtClassLoader@addbf1

//表示系统类装载器parent的parent为bootstrap，无法直接获取

null

//表示类Object是由bootstrap装载的

java.lang.Object's loader is null

//表示用户类是由系统类装载器装载的

ClassLoaderDemo's loader is sun.misc.Launcher$AppClassLoader@19821f

## 类加载器的主要方法

调用路径下的文件：比如C3-p0..

//获取类路径下对应的文件输入流

InputStream in = **null**;

in = **this**.getClass().getClassLoader().getResourceAsStream("1.properties ");

