基础加强

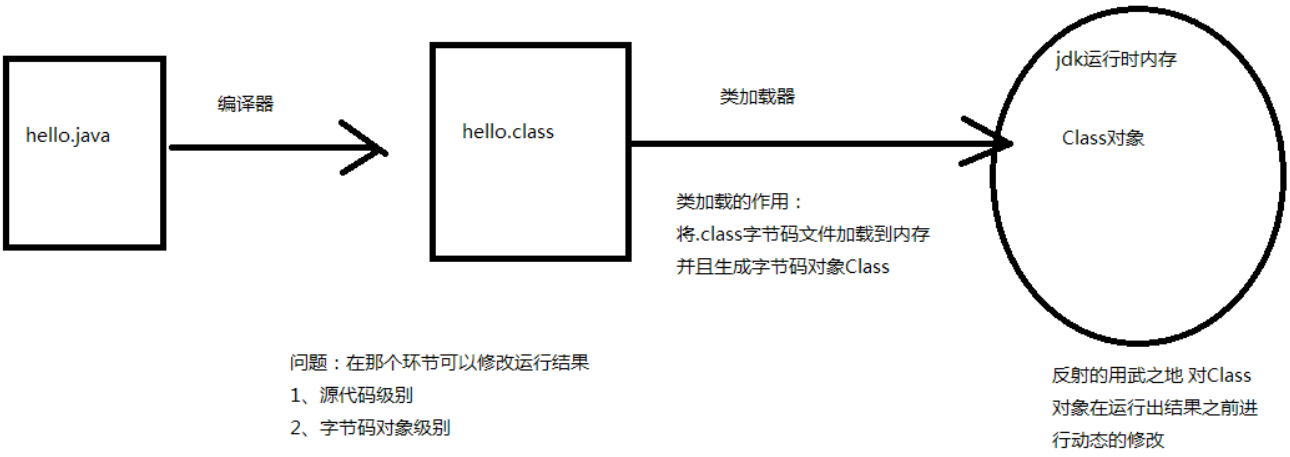
教学导航

教学目标	案例-自定义单元测试@MyTest 案例-全局的编码的解决
教学方法	

一、类加载器

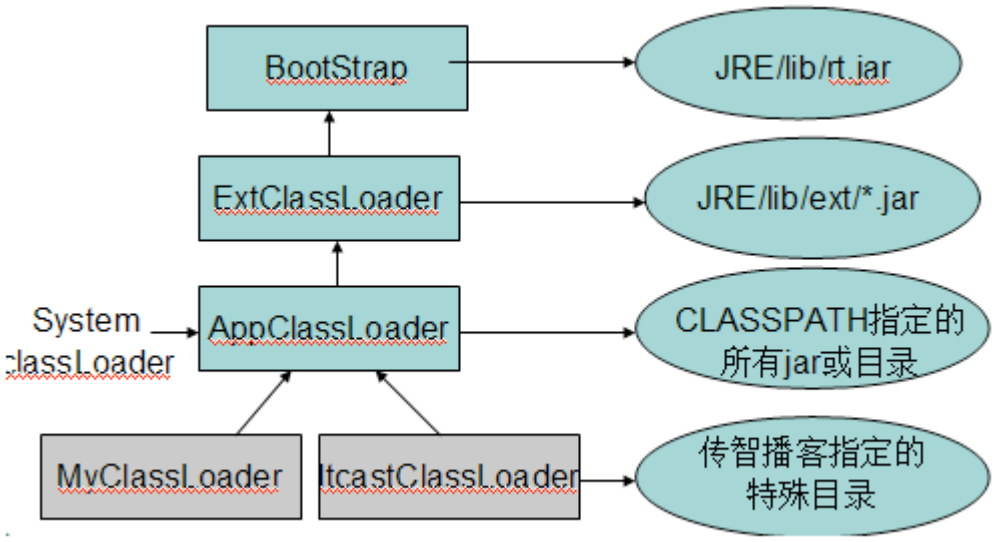
1．什么是类加载器，作用是什么？

类加载器就加载字节码文件(.class)



2．类加载器的种类

类加载器有三种，不同类加载器加载不同的



1）BootStrap：引导类加载器：加载都是最基础的文件
2）ExtClassLoader：扩展类加载器：加载都是基础的文件
3）AppClassLoader：应用类加载器：三方 jar 包和自己编写 java 文件

怎么获得类加载器？(重点)
ClassLoader 字节码对象.getClassLoader();

二、注解 @xxx

1．什么是注解，注解作用

注解就是符合一定格式的语法 @xxxx
注解作用：
注释：在阅读程序时清楚----给程序员看的
注解：给 jvm 看的，给机器看的

注解在目前而言最主流的应用：代替配置文件
关于配置文件与注解开发的优缺点：

注解优点：开发效率高 成本低

注解缺点：耦合性大 并且不利于后期维护

2．jdk5 提供的注解

@Override：告知编译器此方法是覆盖父类的

@Deprecated：标注过时

@SuppressWarnings：压制警告

发现的问题：

不同的注解只能在不同的位置使用(方法上、字段上、类上)

3．自定义注解（了解）

1）怎样去编写一个自定义的注解

2）怎样去使用注解

3）怎样去解析注解-----使用反射知识

(1)编写一个注解

关键字：@interface

注解的属性：

语法：返回值 名称();

注意：如果属性的名字是 value，并且注解的属性值有一个 那么在使用注解时可以省略 value

```
public @interface MyAnno {

    String value();

    //String addr();

    int age() default 28;
}
```

注解属性类型只能是以下几种

1.基本类型

2.String
3.枚举类型
4.注解类型
5.Class 类型
6.以上类型的一维数组类型

(2)使用注解

在类/方法/字段 上面是@XXX

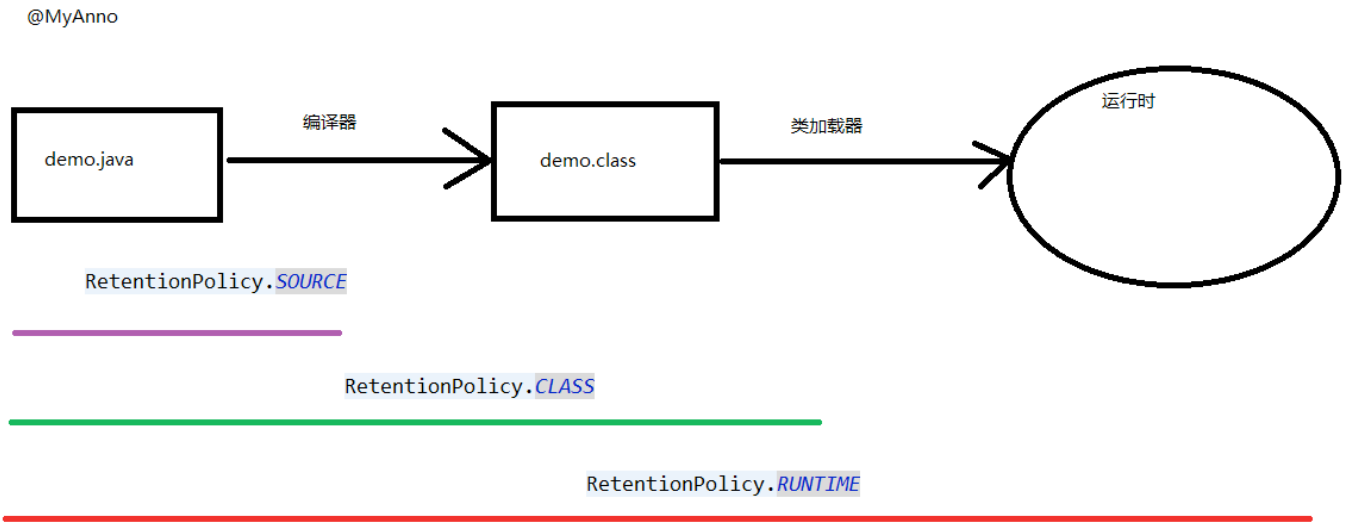
```
@MyAnno("zhangsan")
public class TestAnno {

}
```

(3)解析使用了注解的类

介入一个概念：元注解：代表修饰注解的注解，作用：限制定义的注解的特性

@Retention
SOURCE: 注解在源码级别可见
CLASS：注解在字节码文件级别可见
RUNTIME：注解在整个运行阶段都可见

@Target
代表注解修饰的范围：类上使用，方法上使用，字段上使用
FIELD:字段上可用此注解
METHOD:方法上可以用此注解
TYPE:类/接口上可以使用此注解



注意：要想解析使用了注解的类 ， 那么该注解的 Retention 必须设置成 Runtime

关于注解解析的实质：从注解中解析出属性值

字节码对象存在于获得注解相关的方法

isAnnotationPresent(Class<? extends Annotation> annotationClass) ： 判断该字节码对象身上是否使用该注解了

getAnnotation(Class<A> annotationClass) ： 获得该字节码对象身上的注解对象

三、动态代理

1. 什么是代理(中介)

目标对象/被代理对象 ------ 房主：真正的租房的方法
代理对象 ------- 黑中介：有租房子的方法（调用房主的租房的方法）
执行代理对象方法的对象 ---- 租房的人

流程：我们要租房----->中介（租房的方法）------>房主（租房的方法）
抽象：调用对象----->代理对象------>目标对象

2．动态代理

动态代理：不用手动编写一个代理对象，不需要一一编写与目标对象相同的方法，这个过程，在运行时 的内存中动态生成代理对象。------字节码对象级别的代理对象

动态代理的 API：

在 jdk 的 API 中存在一个 Proxy 中存在一个生成动态代理的的方法 newProxyInstance

static Object	**newProxyInstance**(ClassLoader loader, Class<?>[] interfaces, InvocationHandler h)

返回值：Object 就是代理对象

参数：loader：代表与目标对象相同的类加载器-------目标对象.getClass().getClassLoader()

interfaces：代表与目标对象实现的所有的接口字节码对象数组

h：具体的代理的操作，InvocationHandler 接口

注意 :JDK 的 Proxy 方式实现的动态代理 目标对象必须有接口 没有接口不能实现 jdk 版动态代理