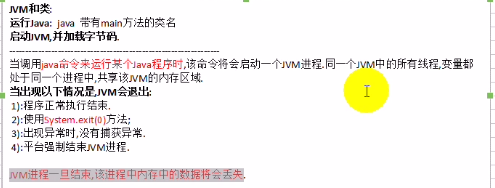
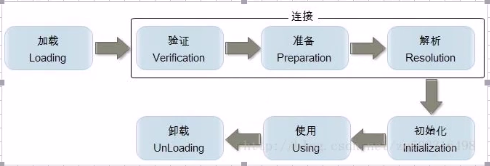
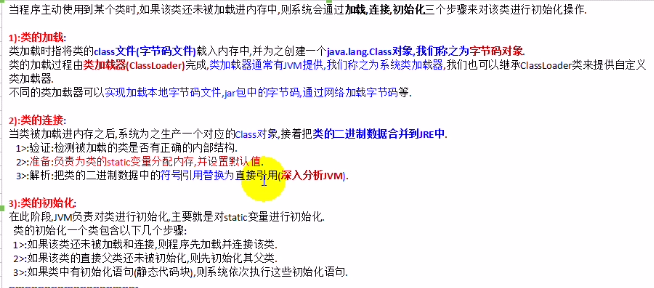
Day23 反射机制

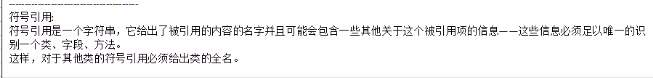


**01类加载机制讲解**









02引出反射和什么是反射



对象都有编译类型和运行类型

编译类相关：Object

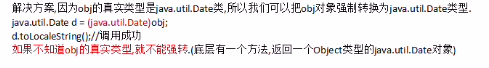
运行类型：java.util.Date

需求：通过obj对象调用java.util>toLocaleString方法

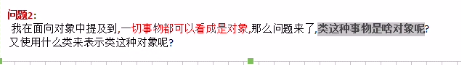
此时编译报错，编译死灰检查编译类型中是否次存在toLocaleString方法

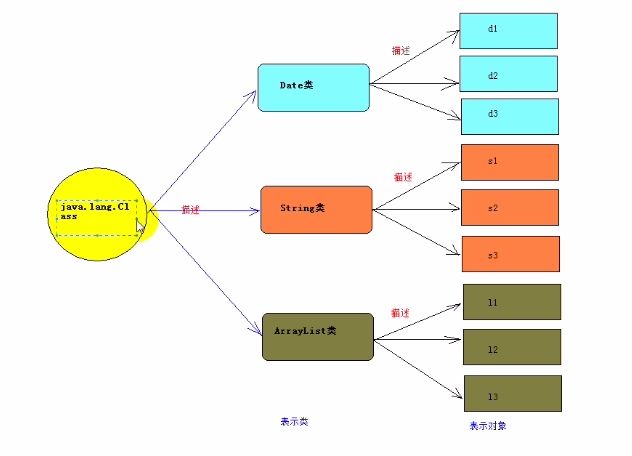
如果存在编译称成功，否则编译失败

解决方案：强制转换



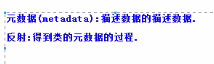
OO的世界：一切皆对象



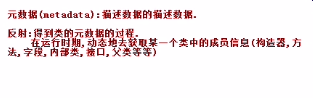


Java.lang.Class描述为字节码对象

类的对象加载进内存就是字节码



反射：在运行时期动态的获取类中的成员信息



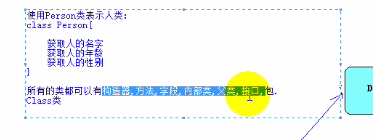
并且把类中的每一种成员都描述为一种新的类。

Class:表示所有的类

Constructor：表示所有的构造器

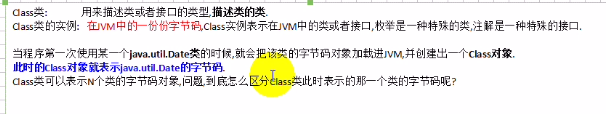
Method:表示所有的方法

Field:表示所有的字段或成员变量

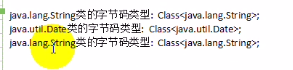


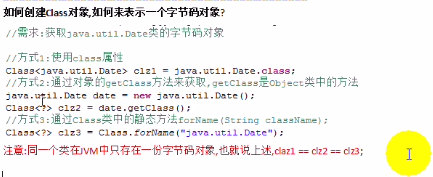
反射功能非常损耗性能

03class类和class实例

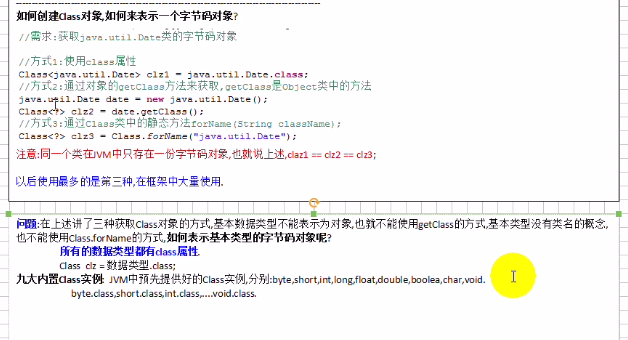


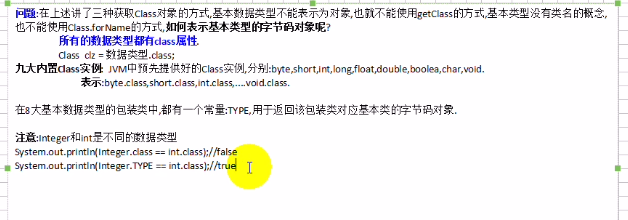
为了解决该问题class类的设计者提供了泛型->Class<T>

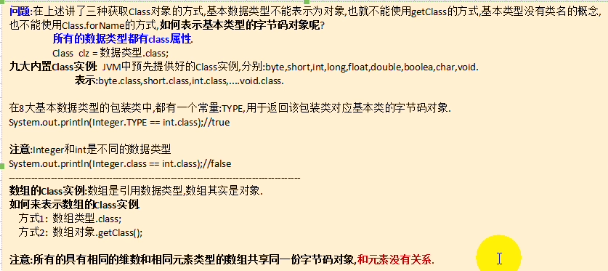




04九大内置class实例和数组的class实例

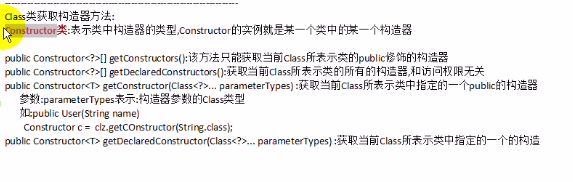






05获取类中的构造器



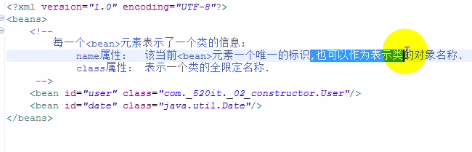


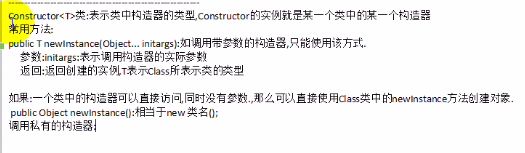
06使用反射调用构造器

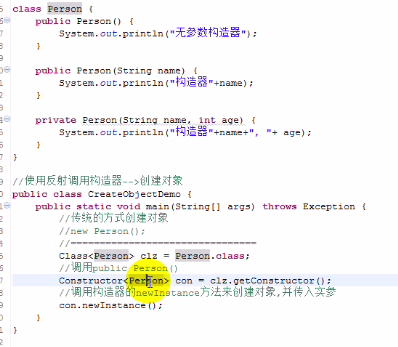
构造器最大的作用就是创建对象

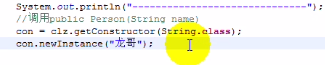
**为什么使用反射创建对象，为什么不直接new呢？**

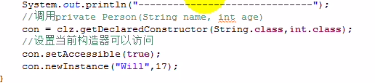
因为框架中提供的都是字符串

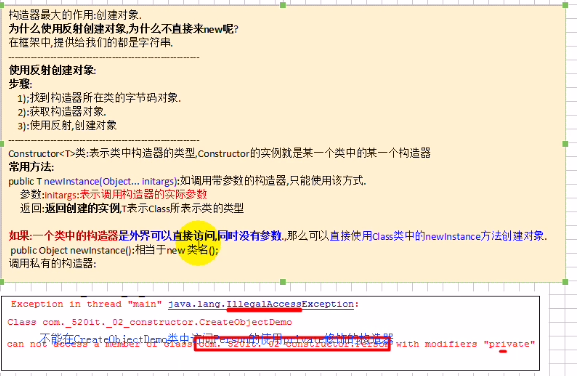






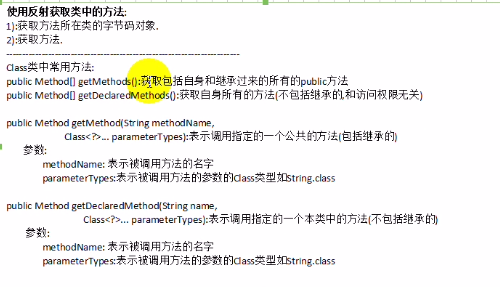


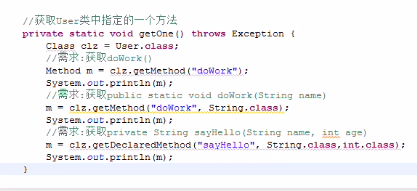




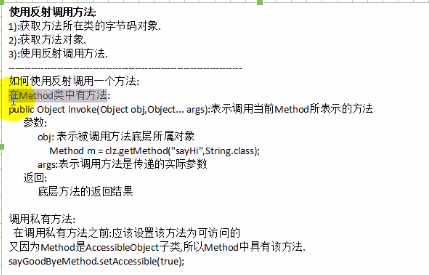


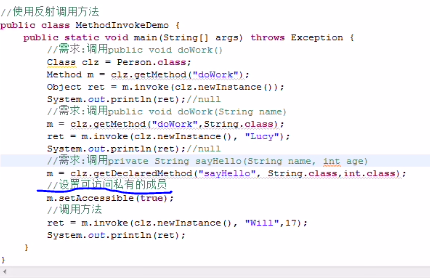
07获取类中的方法





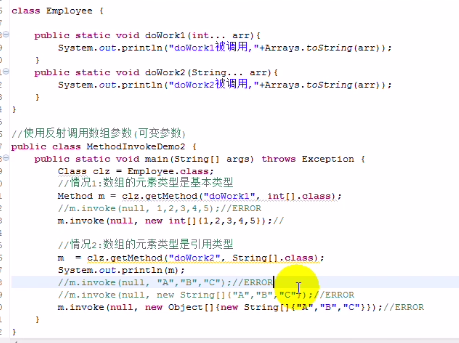
08使用反射调用方法

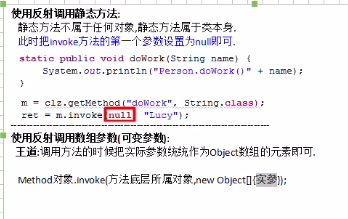


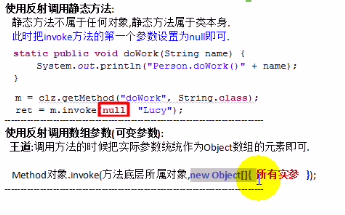


09使用反射调用静态方法和数组参数方法

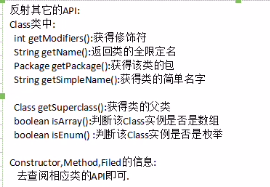


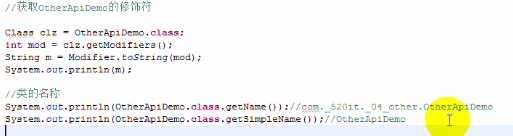




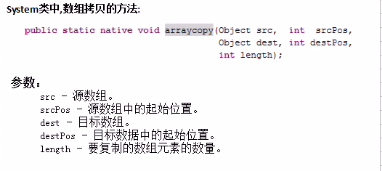


10反射其他相关

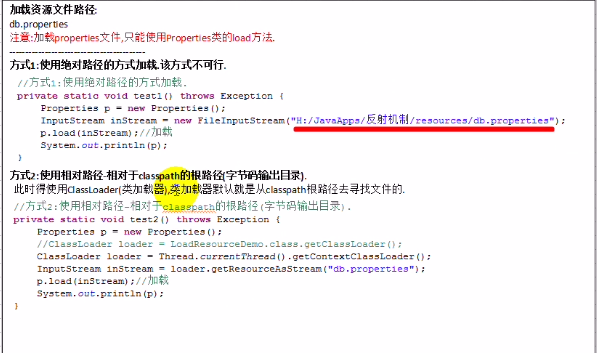


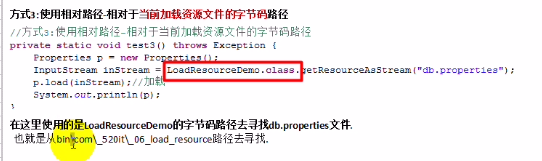


11实现任意类型数组的拷贝操作



12加载资源文件路径





13今日小结

