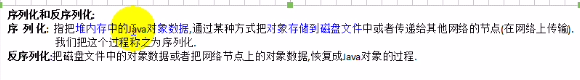
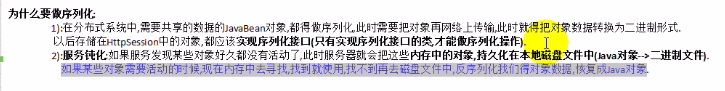
Day21

01序列化和反序列化概述





需要做序列化的对象要实现序列化接口



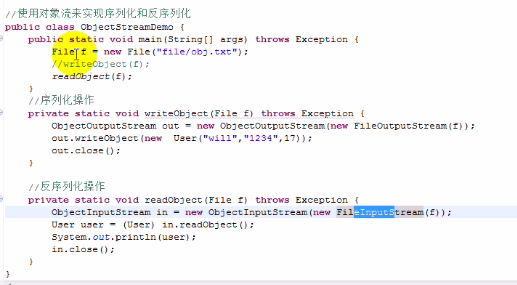
不包含抽象方法的接口是标志接口，用来限制类型（底层在使用的时候回判断instanceof）

02io对象流

**使用对象流来实现序列化和反序列化操作**

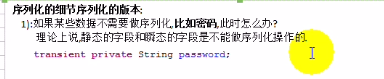
ObjectInputstream：反序列化

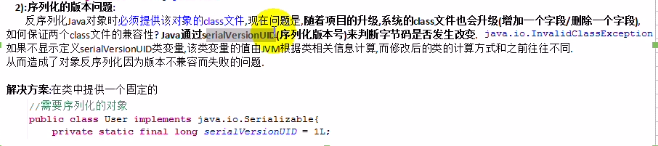
ObjectOutstream：序列化



反序列化一定要存在字节码对象

03对象操作流西河和serialversionuid

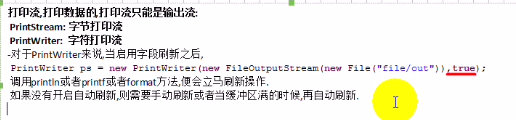


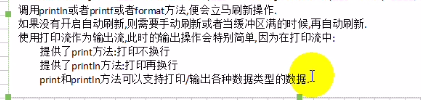


04打印流-printwrite

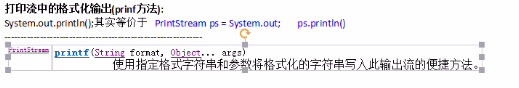
打印字节

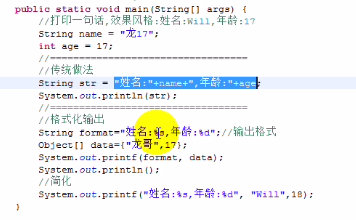
打印字符





05打印流的格式化输出-printf





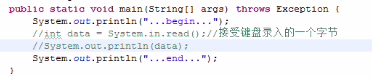
06标准io概述和操作

标准输入：通过键盘输入数据给程序

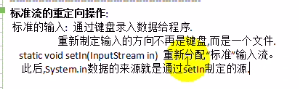
标准输出：通过屏幕显示程序数据

Inputstream =system.in

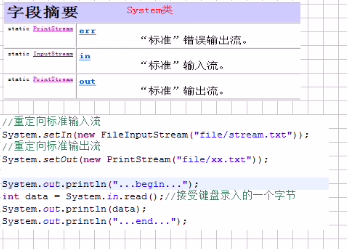
Outputstream=system.out



标准流的重定向：







07扫描器类

Scanner类：扫描器类，表示输入操作

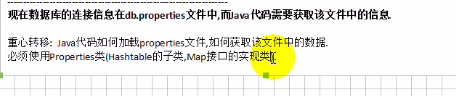




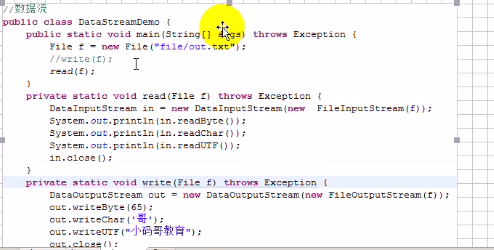
08properties类加载资源文件





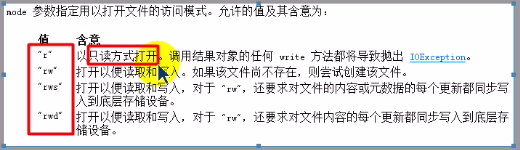


09数据流



10随机访问文件

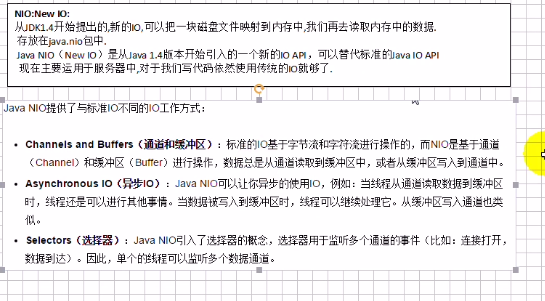
RandomAccessFile：指的是在文件的任何位置读取和写入数据



12nio概述

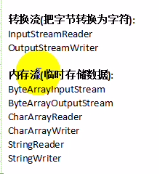
Nio:new io

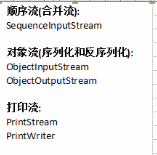
从jdk1.4开始提出的，他可以将磁盘的文件映射到内存中，我们在去读取内存中的数据

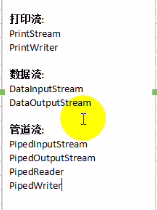


13io流的回顾





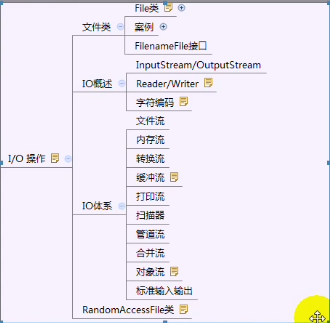




基本流：

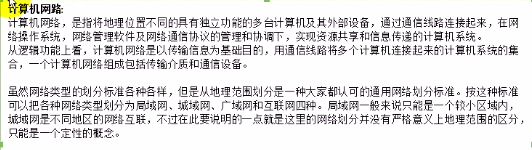


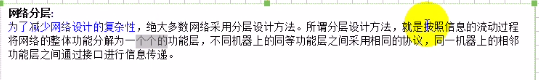
14今日小结

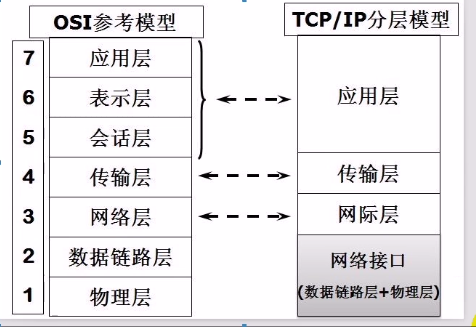


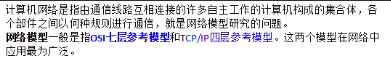
Day22

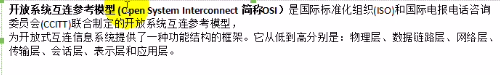
01网络概述和分层模型

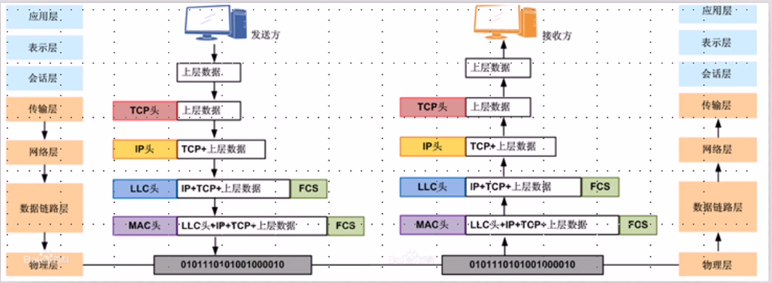




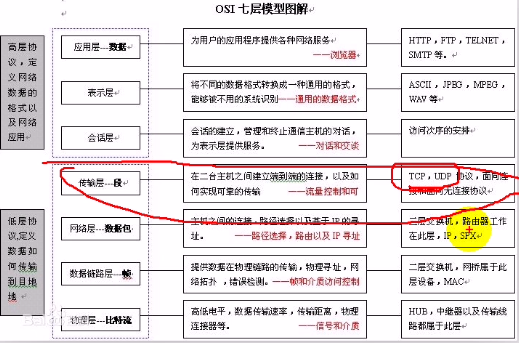




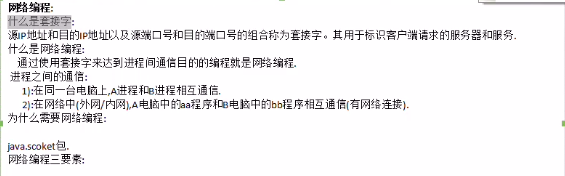




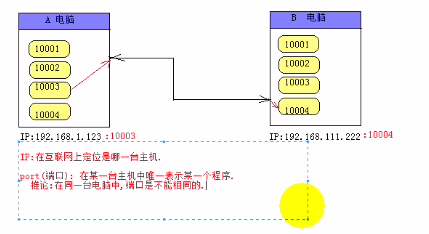


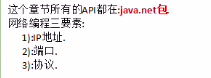


02网络编程概述



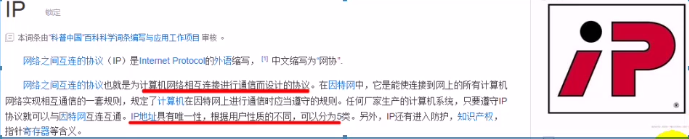
每一个进程都有端口

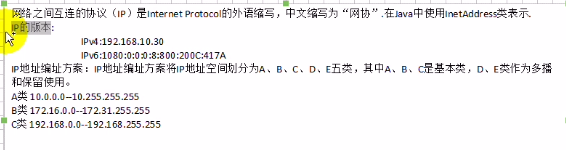


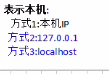


套接字：ip+端口

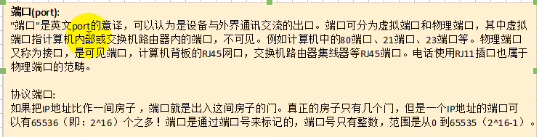
03网络三要素ip地址





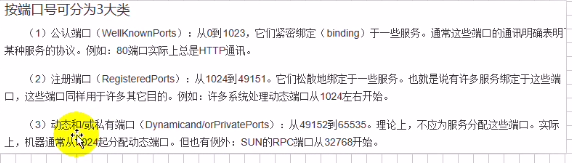


04网络三要素-端口



虚拟端口和物理端口，主要讨论虚拟端口

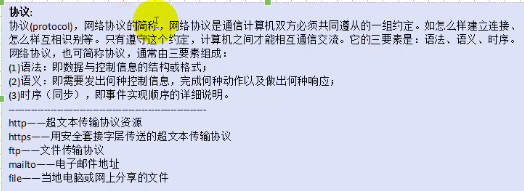
端口范围为0-65535

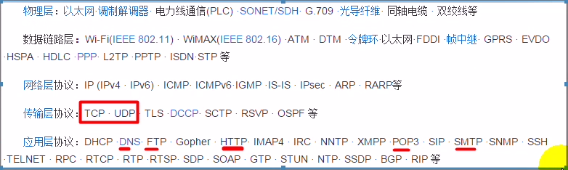




不同的端口代表不同的服务

05网络三要素-协议





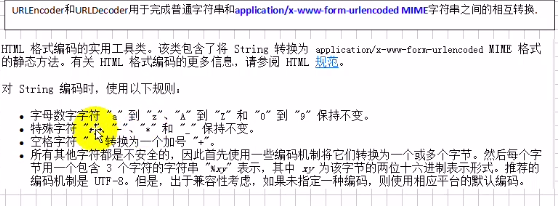
06url和uri概述

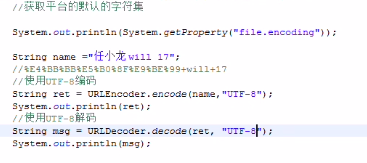


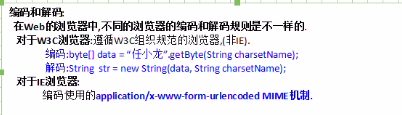


07urlconnection概述

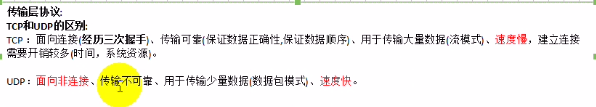
08编码和解码

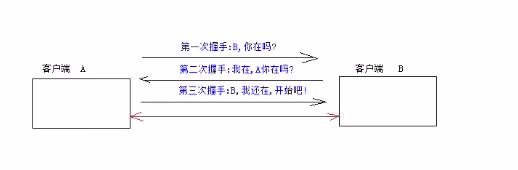






09tcp和udp协议概述

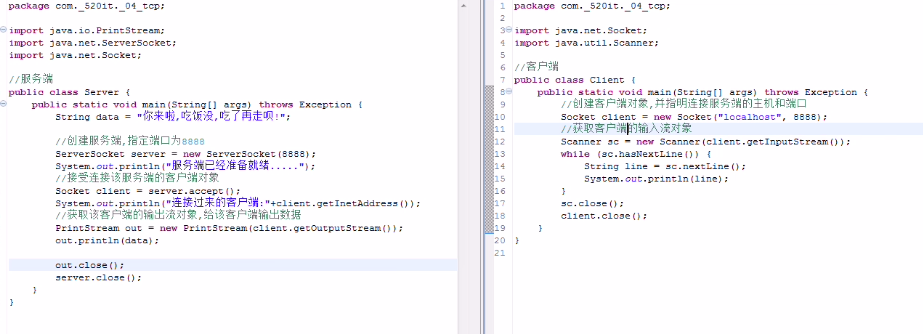




10tcp简单程序

套接字指的就是ip和端口（包括源和目标）

网络编程就是两个进程之间的通信

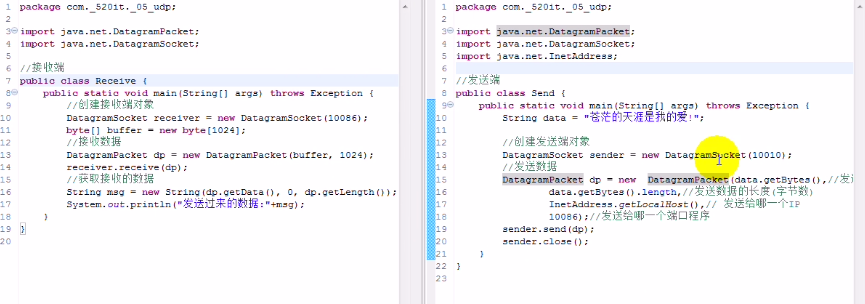


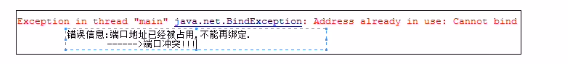
11tcp简单模拟tomcat服务器

12udp简单程序

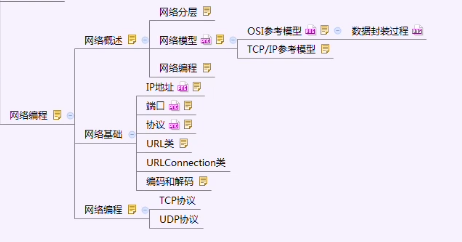
Tcp—流模式

Udp-数据报包，底层还是流





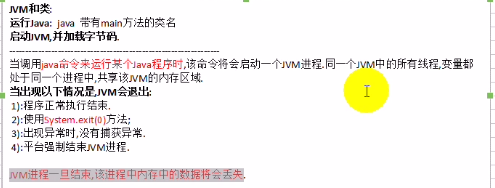
13今日小结

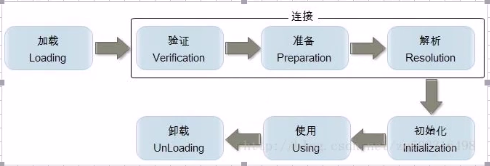


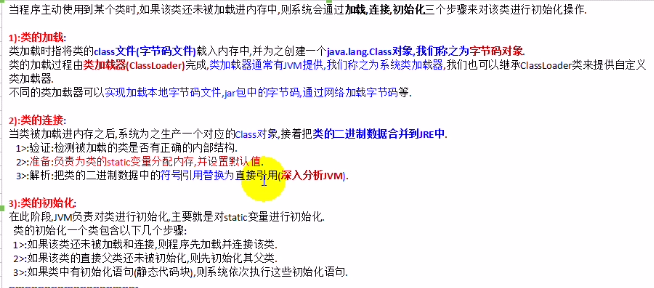
Day23 反射机制

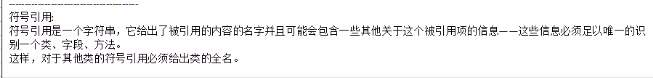


01类加载机制讲解









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

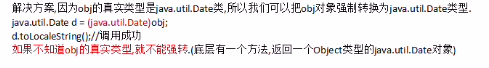
运行类型：java.util.Date

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

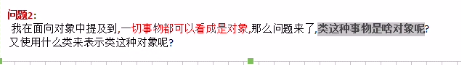
此时编译报错，编译死灰检查编译类型中是否次存在toLocaleString方法

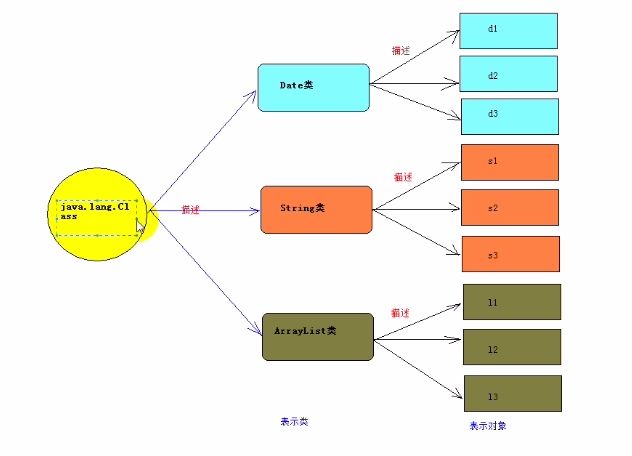
如果存在编译称成功，否则编译失败

解决方案：强制转换



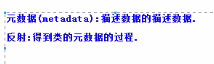
OO的世界：一切皆对象



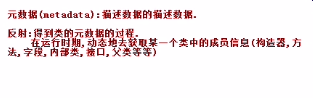


Java.lang.Class描述为字节码对象

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



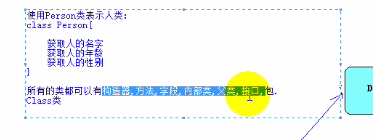
并且把类中的每一种成员都描述为一种新的类。

Class:表示所有的类

Constructor：表示所有的构造器

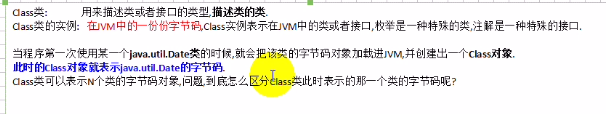
Method:表示所有的方法

Field:表示所有的字段或成员变量

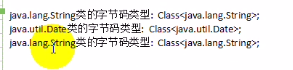


反射功能非常损耗性能

03class类和class实例

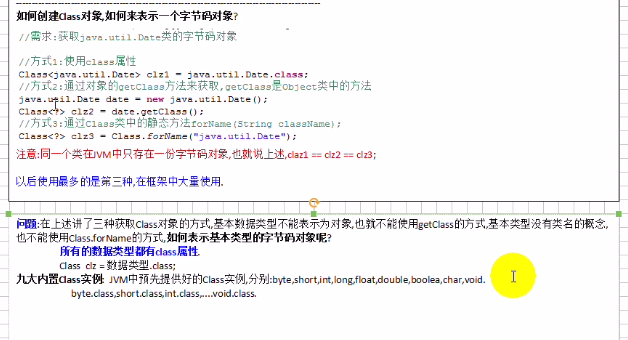


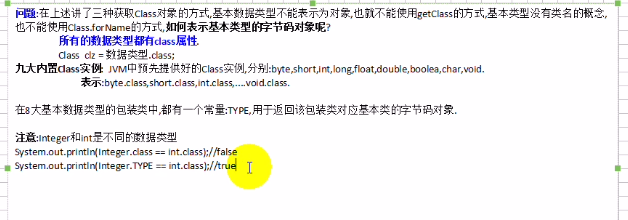
为了解决该问题class类的设计者提供了泛型->Class<T>

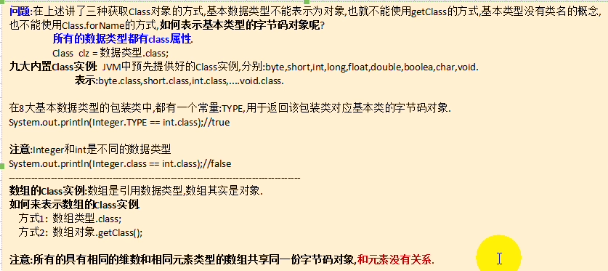




04九大内置class实例和数组的class实例

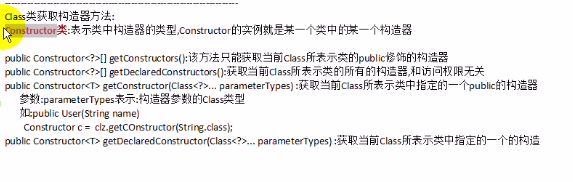






05获取类中的构造器



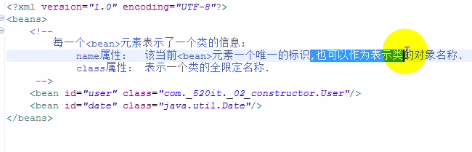


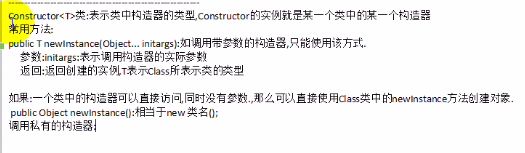
06使用反射调用构造器

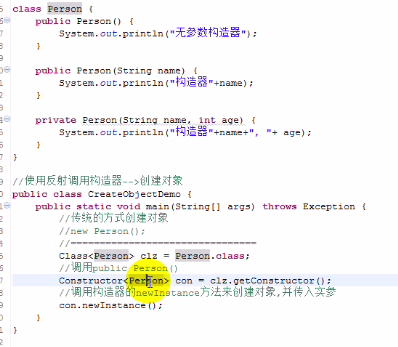
构造器最大的作用就是创建对象

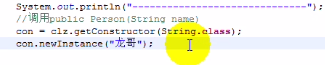
**为什么使用反射创建对象，为什么不直接new呢？**

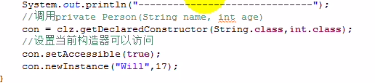
因为框架中提供的都是字符串

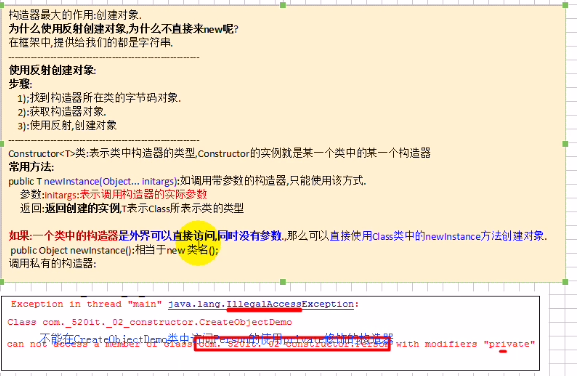






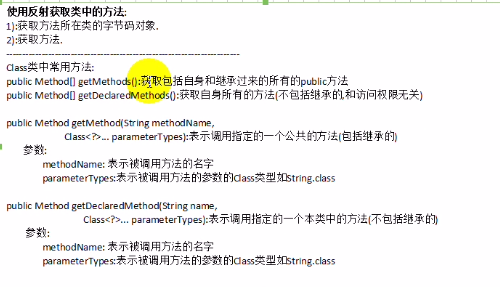






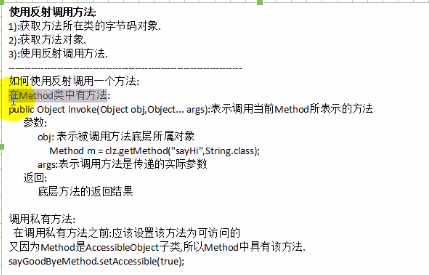


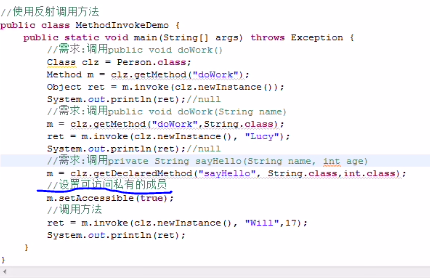
07获取类中的方法





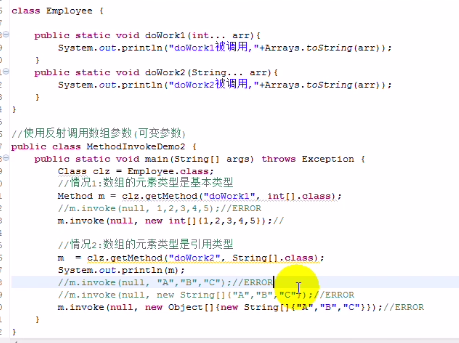
08使用反射调用方法

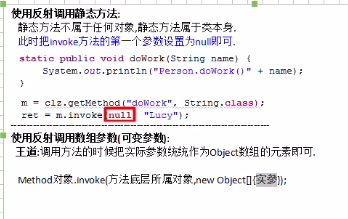


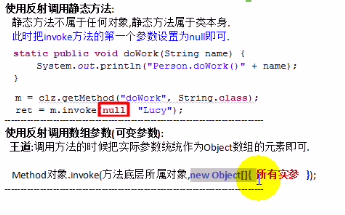


09使用反射调用静态方法和数组参数方法

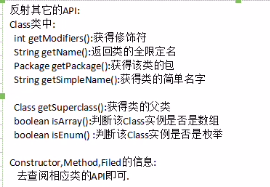


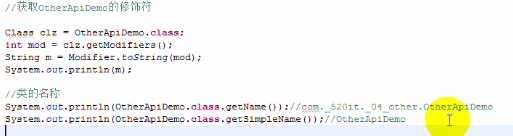




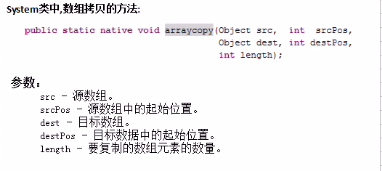


10反射其他相关

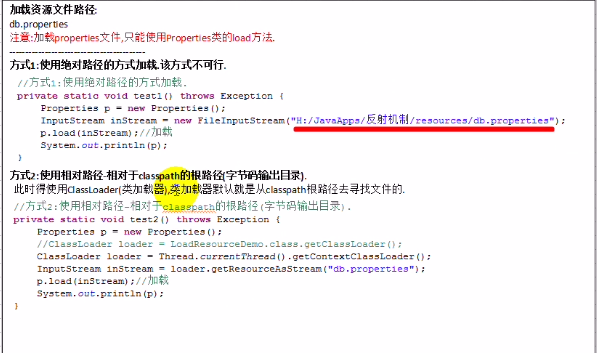


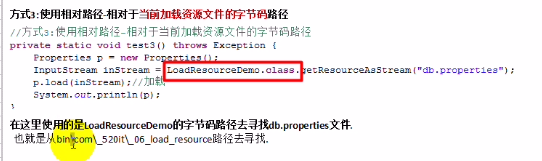


11实现任意类型数组的拷贝操作



12加载资源文件路径





13今日小结

