# 简要

以下所有代码都来自于JDK1.8.

# JDK序列化

JDK序列化是java自带的一种序列化方式，涉及的主要class包括Serializable，Externalizable，ObjectInputStream，ObjectOutputStream。

通过JDK序列化必须实现Serializable或者Externalizable接口，其中Externalizable接口实现了Serializable接口。

Demo参考Serializable\_demo项目

## 序列化过程

通过Demo项目可以看出，一个类的对象通过JDK序列化的主要步骤如下：

1. 装配该类的实例对象；
2. 创建ObjectOutputStream或者ObjectInputStream
3. 调用writeObject或者readObject方法。

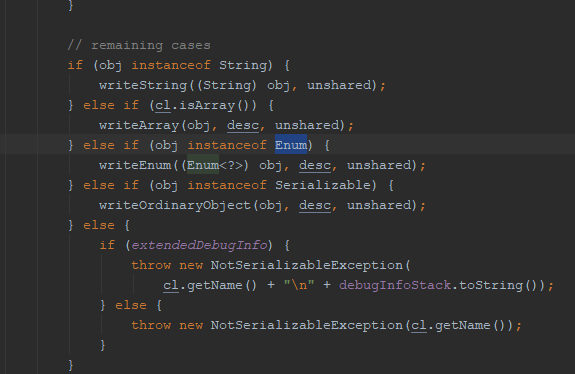
### 原理

JDK序列化是通过调用ObjectOutputStream的writeObject方法来实现的，通过阅读jdk源码可以知道其核心思想是java的反射原理。调用writeObject方法会进入ObjectOutputStream的writeObject0方法，在这个方法中会判断这个对象所属类型：

* String
* 数组
* Enum
* Serializable子类

除了这些类都将抛出异常。代码如下

位置：ObjectOutputStream#writeObject0



代码调用层级：

defaultWriteFields:1548, ObjectOutputStream (java.io)

writeSerialData:1509, ObjectOutputStream (java.io)

writeOrdinaryObject:1432, ObjectOutputStream (java.io)

writeObject0:1178, ObjectOutputStream (java.io)

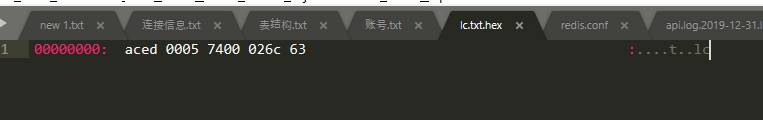
writeObject:348, ObjectOutputStream (java.io)

serialize:21, SerializationUtil (com.lc.app)

main:13, Main (com.lc.app)

#### String对象

进入writeString方法可以发现，序列化String对象需要写三个部分，一个是类型标识符（ObjectStreamConstants类的TC\_STRING或者TC\_LONGSTRING），然后是String对象的长度，最后是String内容。比如写入“lc”之后的二进制如下：



#### 数组对象

如果需要序列化的是数组对象，需要调用writeArray方法，该方法的实现逻辑是，参考ObjectOutputStream#writeArray方法：

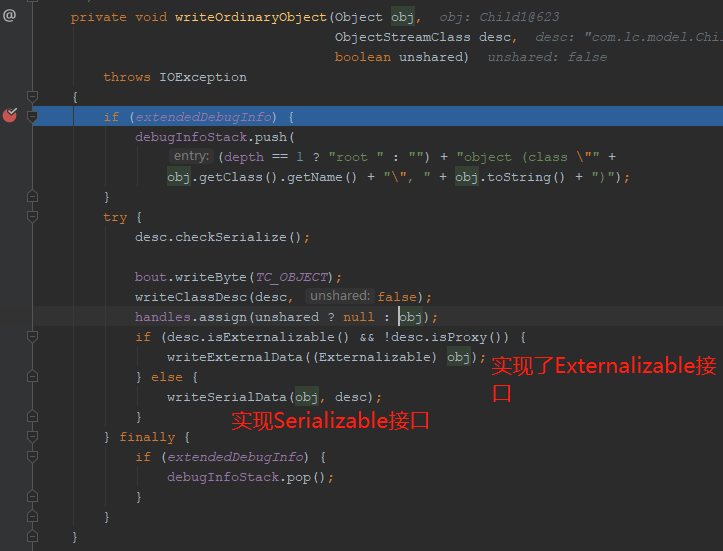
1. 获取数组中元素类型；
2. 如果是java built-in类型，通过批量分配OutputStream的内存，将数组中的元素copy到OutputStream中；
3. 如果是非built-in类型，遍历每一个元素，调用writeObject0方法（相当于递归重新调用序列方法）

#### Enum对象

Enum对象的序列化和String对象的序列化过程很类似，写入的内容也包括三部分，第一部分是类型标识符，第二部分是类型信息，第三部分是枚举对象的name；

#### Serializable子类对象（重点）

Serializable子类，包括java内部实现了Serializable接口的类、用户自定义的实现了Serializable接口的类。这些类序列化时调用writeOrdinaryObject（ObjectOutputStream类中）。代码逻辑如下，一般都是调用writeSerialData方法。



接下来将以writeSerialData为例讲解。

1. 获取所有父类信息，从上往下依次存放；
2. 遍历每个父类，如果遍历的类中自定义了writeObject方法则通过反射调用该类中自定义的writeObject方法，否则调用defaultWriteFields方法
3. defaultWriteFields方法首先将该类中java内置类型的field写入OutputStream，然后逐个遍历非内置类型的field，并依次调用writeObject0方法（相当于重新递归调用执行序列化）

### 总结

JDK序列化在序列化一个非String，Array，Enum类型的对象时，最终会转换为序列化java内置类型，String，Enum类型的对象数据（通过递归逐个剖解类中field）。

序列化的思路如下：

1. 调用writeObject0方法，首先判断序列化对象类型
   1. String类型调用writeString写入OutputStream对象；
   2. Enum类型调用writeEnum写入OutputStream对象；
   3. Array类型调用writeArray，如果数组元素是内置类型，通过内存copy存入OutputStream对象，如果是非内置类型，遍历每个元素并执行第一步；
   4. Serializable子类，调用writeOrdinaryObject，进入第二步；
2. 普通类型对象
   1. 如果实现了Externalizable接口，调用writeExternalData方法，这种方式需要用户自己实现序列化协议（也就是自定义二进制存放格式），不常用。
   2. 否则调用writeSerialData方法，进入第三步
3. 获取该类所有父类（从上往下存放），并逐个遍历每个父类
   1. 如果当前遍历类实现了writeObject方法，则调用自定义writeObject方法（通过反射实现）；
   2. 否则调用defaultWriteField方法，进入第4步；
4. defaultWriteField方法首先将定义在当前类中的并且是内置类型的属性写入OutputStream，然后获取其他类型的属性并逐个遍历，重新调用writeObject0方法，进入第1步。

## 反序列化过程

反序列化是序列化的逆过程，在理解序列化原理之后，反序列化过程就一目了然。当调用ObjectInputStream的readObject时，会调用到readObject0方法，该方法会将二进制流文件读入，根据类型标识符调用不同的方法。

### 原理

#### String对象

当类型标识符是TC\_STRING或者TC\_LONGSTRING时，调用readString方法，从InputStream对象中直接读取二进制数据并转换为String对象。

#### 数组对象

#### Enum对象

#### Serializable子类对象

# Kryo序列化

# Dubbo内嵌Hessian2序列化

# 参考