#### 1、Java序列化与反序列化是什么？

Java序列化是指把Java对象转换为字节序列的过程，而Java反序列化是指把字节序列恢复为Java对象的过程：

序列化：对象序列化最主要的用处就是在传递和保存对象的时候，保证对象的完整性和可传递性。序列化是把对象转换成有序字节流，以便在网络上传输或者保存在本地文件中。核心作用是对象状态的保存与重建。

反序列化：客户端从文件中或网络上获得序列化后的对象字节流，根据字节流中所保存的对象状态及描述信息，通过反序列化重建对象。

#### 2、为什么需要序列化与反序列化

为什么要序列化，序列化有什么什么优点。

**1）对象序列化可以实现分布式对象。**

主要应用例如：RMI（Remote Method Invocation远程调用）要利用对象序列化运行远程主机上的服务，就像在本地机上运行对象时一样。

**2）java对象序列化不仅保留一个对象的数据，而且递归保存对象引用的每个对象的数据。**

可以将整个对象层次写入字节流中，可以保存在文件中或在网络连接上传递。利用对象序列化可以进行对象的“深复制”，即复制对象本身及引用的对象本身。序列化一个对象可能得到整个对象序列。

**3）序列化可以将内存中的类写入文件或数据库中。**

比如：将某个类序列化后存为文件，下次读取时只需将文件中的数据反序列化就可以将原先的类还原到内存中。也可以将类序列化为流数据进行传输。

总的来说就是将一个已经实例化的类转成文件存储，下次需要实例化的时候只要反序列化即可将类实例化到内存中并保留序列化时类中的所有变量和状态。

**4）对象、文件、数据，有许多不同的格式，很难统一传输和保存。**

序列化以后就都是字节流了，无论原来是什么东西，都能变成一样的东西，就可以进行通用的格式传输或保存，传输结束以后，要再次使用，就进行反序列化还原，这样对象还是对象，文件还是文件。

#### 3、如何实现Java序列化与反序列化

首先我们要把准备要序列化类，实现Serializable接口。

例如：我们要Person类里的name和age都序列化。

public class Person implements Serializable {

private String name;  
 private int age;  
  
 public Person(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 @Override  
 public String toString() {  
 return "姓名：" + this.name + ", age=" + this.age;  
 }  
}

然后：我们将name和age序列化（也就是把这2个对象转为二进制，理解为“打碎”）

public class ObjectOutputStreamDemo {

public static void main(String[] args) throws Exception {  
 //序列化后生成指定文件路径  
 File file = new File("/Users/chance/Desktop" + File.*separator* + "person.ser");  
 ObjectOutputStream oos = null;  
 //装饰流（流）  
 oos = new ObjectOutputStream(new FileOutputStream(file));  
  
 //实例化类  
 Person per = new Person("张三", 30);  
 //把类对象序列化  
 oos.writeObject(per);  
 oos.close();  
 }  
}

《Java对象的序列化（Serialization）和反序列化详解》

<https://blog.csdn.net/yaomingyang/article/details/79321939>

《Java 序列化的高级认识》

<https://www.ibm.com/developerworks/cn/java/j-lo-serial/>