1. 序列化的方式

把 Java 对象转换为字节序列,并存储至一个储存媒介的过程。

反序列化：把字节序列恢复为 Java 对象的过程。

简单说法是：序列化把当前对象信息保存下来。反序列化刚好相反的操作，即读取信息设置到当前对象上。

序列化作用：

1) 永久性保存对象，保存对象的字节序列到本地文件中；

2) 通过序列化对象在网络中传递对象；

3) 通过序列化在进程间传递对象。

2. Serializable 和 Parcelable 的区别

1) 编码上：

Serializable 代码量少，写起来方便

Parcelable 代码多一些

2) 效率上：

Parcelable 的速度比 Serializable 高十倍以上

serializable 的迷人之处在于你只需要对某个类以及它的属性

实现 Serializable 接口即可。Serializable 接口是一种标识接口（marker interface），这意味着无需实现方法，Java 便会对这个对象进行高效的序列化操作。这种方法的缺点是使用了反射，序列化的过程较慢。这种机制会在序列化的时候创建许多的临时对象，容易触发垃圾回收。

Parcelable 方式的实现原理是将一个完整的对象进行分解，而分解后的每一部分都是 Intent 所支持的数据类型，这样也就实现传递对象的功能了

3. 如何将一个 Java 对象序列化到文件里？（源码）

将对象序列化到文件

1) 对象需要实现 Seralizable 接口

```
public class StudentBean implements Serializable {
......
}
```

2) 通过 ObjectOutputStream 的 writeObject()方法写入

和 ObjectInputStream 的 readObject()方法来进行读取

//存进去

```
try {
		ObjectOutputStream os = new ObjectOutputStream(
		new FileOutputStream("D:/student.txt"));
		os.writeObject(studentList);
		os.close();
} catch (FileNotFoundException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
}
```

//读出来

```
try {
		ObjectInputStream is = new ObjectInputStream(
		new FileInputStream("D:/student.txt"));
		ArrayList<StudentBean> list = new ArrayList<StudentBean>();
		list = (ArrayList<StudentBean>) is.readObject();
		for (int i = 0; i < list.size(); i++) {
			System.out.println(list.get(i).toString());
		}
} catch (FileNotFoundException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
} catch (ClassNotFoundException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
}
```