北京圣思园科技有限公司
http://www.shengsiyuan.com

主讲人：张龙

深拷贝（deep clone）与浅拷贝(shallow clone)

- 浅复制与深复制概念
  - 浅复制（浅克隆）：被复制对象的所有变量都含有与原来的对象相同的值，而所有的对其他对象的引用仍然指向原来的对象。换言之，浅复制仅仅复制所考虑的对象，而不复制它所引用的对象。

深拷贝（deep clone）与浅拷贝(shallow clone)

- 浅复制与深复制概念
  - 深复制（深克隆）：被复制对象的所有变量都含有与原来的对象相同的值，除去那些引用其他对象的变量。那些引用其他对象的变量将指向被复制过的新对象，而不再是原有的那些被引用的对象。换言之，深复制把要复制的对象所引用的对象都复制了一遍。

深拷贝（deep clone）与浅拷贝(shallow clone)

- Java的clone()方法【定义在Object类中】
  - clone方法将对象复制了一份并返回给调用者。一般而言，clone（）方法满足：
  - ①对任何的对象x，都有x.clone() !=x
    - 克隆对象与原对象不是同一个对象
  - ②对任何的对象x，都有x.clone().getClass()= =x.getClass()
    - 克隆对象与原对象的类型一样
  - ③如果对象x的equals()方法定义恰当，那么x.clone().equals(x)应该成立。

深拷贝（deep clone）与浅拷贝(shallow clone)

- Java中对象的克隆
  - ①为了获取对象的一份拷贝，我们可以利用Object类的clone()方法。
  - ②在派生类中覆盖基类的clone()方法，并声明为public【Object类中的clone()方法为protected的】。
  - ③在派生类的clone()方法中，调用super.clone()。
  - ④在派生类中实现Cloneable接口。

深拷贝（deep clone）与浅拷贝(shallow clone)

- 说明：
  - ①为什么我们在派生类中覆盖Object的clone()方法时，一定要调用super.clone()呢？
    - 在运行时刻，Object中的clone()识别出你要复制的是哪一个对象，然后为此对象分配空间，并进行对象的复制，将原始对象的内容一一复制到新对象的存储空间中。
  - ②继承自java.lang.Object类的clone()方法是浅复制

利用序列化来做深复制

- 把对象写到流里的过程是序列化（Serilization）过程，而把对象从流中读出来的过程则叫做反序列化（Deserialization）过程。应当指出的是，写在流里的是对象的一个拷贝，而原对象仍然存在于JVM里面。

利用序列化来做深复制

- 在Java语言里深复制一个对象，常常可以先使对象实现Serializable接口，然后把对象（实际上只是对象的一个拷贝）写到一个流里，再从流里读出来，便可以重建对象。
- 这样做的前提是对象以及对象内部所有引用到的对象都是可串行化的，否则，就需要仔细考察那些不可串行化的对象可否设成transient，从而将之排除在复制过程之外。

深拷贝（deep clone）与浅拷贝(shallow clone)

- 注意：Cloneable与Serializable都是marker Interface,也就是说他们只是一个标识接口，没有定义任何方法。

关于实现Serializable接口的类中的serialVersionUID问题

- 当一个类实现了Serializable接口时，表明该类可被序列化，这个时候Eclipse会要求你为该类定义一个字段，该字段名字为serialVersionUID，类型为long，提示信息如下

关于实现Serializable接口的类中的serialVersionUID问题

- The serializable class Student4 does not declare a static final serialVersionUID field of type long

关于实现Serializable接口的类中的serialVersionUID问题

- 你可以随便写一个，在Eclipse中它替你生成一个，有两种生成方式：
  一个是默认的1L，比如：private static final long serialVersionUID = 1L;
  一个是根据类名、接口名、成员方法及属性等来生成一个64位的哈希字段，比如：private static final long serialVersionUID = 8940196742313994740L;之类的。

关于实现Serializable接口的类中的serialVersionUID问题

- 当你一个类实现了Serializable接口，如果没有定义serialVersionUID，Eclipse会提供这个提示功能告诉你去定义之。
  在Eclipse中点击类中warning的图标一下【即那个黄色的图标】，Eclipse就会自动给定两种生成的方式，如上面所述。如果不想定义它，在Eclipse的设置中也可以把它关掉的，设置如下

关于实现Serializable接口的类中的serialVersionUID问题

- Window ==> Preferences ==> Java ==> Compiler ==> Error/Warnings ==> Potential programming problems
  将Serializable class without serialVersionUID的warning改成ignore即可。

关于实现Serializable接口的类中的serialVersionUID问题

- 如果你没有考虑到兼容性问题时，就把它关掉，不过有这个功能是好的，只要任何类别实现了Serializable这个接口的话，如果没有加入serialVersionUID，Eclipse都会给你warning提示，这个serialVersionUID为了让该类别Serializable向后兼容。

关于实现Serializable接口的类中的serialVersionUID问题

- 如果你的对象序列化后存到硬盘上面后，可是后来你却更改了类的field(增加或减少或改名)，当你反序列化时，就会出现Exception的，这样就会造成不兼容性的问题。
- 但当serialVersionUID相同时，它就会将不一样的field以type的缺省值Deserialize，这个可以避开不兼容性的问题。