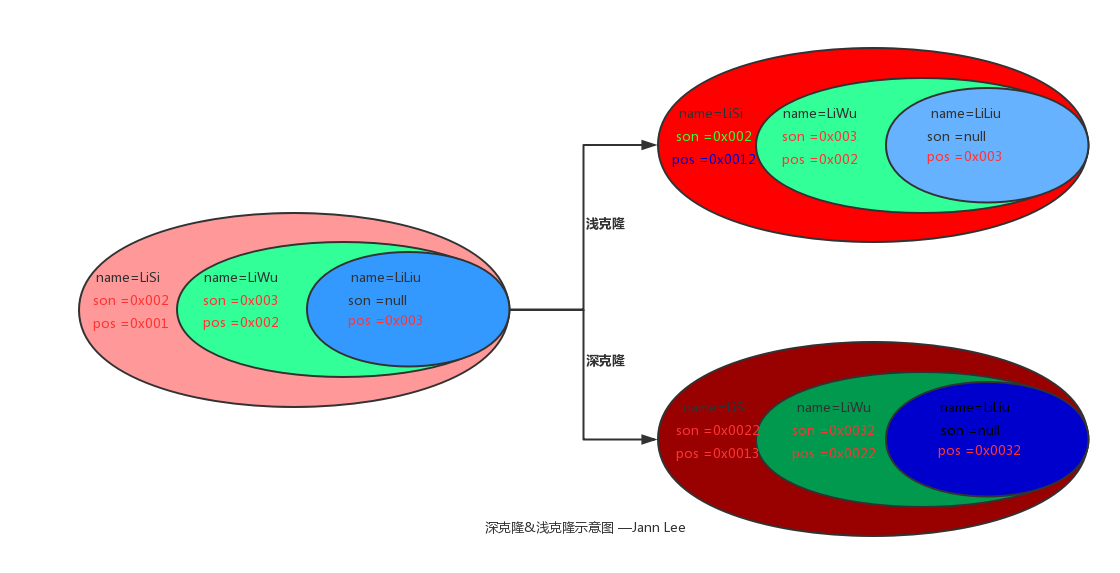
1. **八种基本数据类型和其包装类的使用场景和区别**
2. 基本数据类型对应包装类：byte – 0 – Byte、short – 0 – Short、int – 0 – Interger、long – 0– Long、float – 0.0 – Float、double – 0.0 – Double、char – 空字符（\u0000）– Character、boolean – false – Boolean
3. 使用场景：
   1. 泛型种使用包装类。
   2. 集合类只能使用包装类。
   3. 反射调用函数时使用包装类。
   4. 希望存在null就用包装类，反之用基本数据类型。
   5. 包装类有一些方法，有复杂逻辑处理时，建议使用包装类。
4. 注意事项：
   1. 使用>或<比较大小时会自动拆箱，使用==比较时不会。
   2. Java常量池种数值范围为【-128，127】，使用字面值为一个包装类初始化，如果字面值在【-128，127】之间，jvm会在常量池中寻找该常量，那么该包装类的引用指向常量池中的一片区域，如果字面值超过这个范围，会创建包装类对象。
   3. 基本数据类型存放在java栈中，运算效率较高，包装类存在在堆中，有一些常用方法的实现。
5. **深克隆和浅克隆**



1. 克隆的实现：java类实现Cloneable接口并重写clone方法。
2. 浅克隆：创建一个对象，新对象的属性和原来对象完全相同，对于非基本类型的属性，仍指向原来对象属性的内存地址。
3. 深克隆：创建一个对象，属性中引用的其他对象也会被克隆，不在指向原有对象地址。
4. 如果只是简单的super.clone()重写，只能实现浅克隆；要实现深克隆，就要把对象的引用类型属性和属性的引用类型属性都是实现Cloneale，并重写clone方法，然后在clone方法中调用。
5. 实现深克隆的另一种方法：使用对象流将对象写入流中，然后再读出来。
6. 在选择深克隆方法时，应该根据对象的复杂程度判断，如果对象只有一层或者两层引用类型的属性，选择在clone方法里面实现较好，反之使用对象流较好。
7. **Java浮点为什么会丢失**
8. 并不是java的浮点数精度会丢失，而是所有用二进制存储的浮点数都有可能会精度丢失，所以计算机存储的浮点数都存在精度丢失的风险，不过一般这个丢失的精度对我们正常使用不会构成影响。
9. 小数在转化为二进制时并不一定能用一个精确的二进制表示，大多数时候都是取一个近似值，这就造成了精度丢失。如果再用这个二进制进行计算，明显计算的结果会进一步丢失。
10. 小数如何用二进制表示：首先给出任意实数，整数部分用普通二进制表示；小数部分，将该数字乘以2，取出整数部分作为二进制表示的第一位，然后在将小数部分乘以2，将得到的整数部分作为二进制表示的第二位，以此类推直到小数部分为0。

例如：0.6

0.6 \* 2 = 1.2 ——————- 1   
0.2 \* 2 = 0.4 ——————- 0   
0.4 \* 2 = 0.8 ——————- 0   
0.8 \* 2 = 1.6 ——————- 1   
0.6 \* 2 = 1.2 ——————- 1   
…………

我们可以发现计算机中出现了循环，0.6的二进制表示为100110011001……

如果是10.6，那完整的二进制表示为1010.10011001……

1. 二进制表示的小数如何转化为十进制：从左到右，v[i] \* 2^( - i ), i 为从左到右的index，v[i]为该位的值。

例如：0.6=1\*2^-1 + 0\*2^-2 + 0\*2^-3 + 1\*2^-4 + ……

1. **针对小数精度不够的问题（例如0.1），软件可以人为的在数据最后一位补5，也就是0.15，这样牺牲一位，但可以保证数据精度，还原再把尾巴那个5去掉？？？**
2. BigDecimal就是将小数扩大n倍，转成整数后再进行计算，同时结合指数，得出没有精度损失的结果。
3. **Ddd**