16·面向对象的特征？（概率60%）

**封装**

封装最好理解了。封装是面向对象的特征之一，是对象和类概念的主要特性。

封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

**继承**

面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

**多态**

多态性（polymorphisn）是允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

实现多态，有二种方式，覆盖，重载。

**覆盖**，是指子类重新定义父类的虚函数的做法。

1:方法名相同,参数类型相同,子类的返回值类型大于等于父类,修饰符子类的访问权限要大于等于父类,子类抛出的异常不能大于父类.

**重载**，是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。

1:同一方法中方法名相同,参数列表不同,参数的个数,类型,顺序(具体使用自定义adapter)

49.静态变量合实例变量的区别（概率30%）

语法：静态变量前要加static来修饰，而实例变量前则不加；

程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量；静态变量不属于某个实例对象，而是属于类，所以也称为类变量，静态变量就会被分配空间，静态变量就可以被使用了。

51.什么是匿名内部类，在什么时候调用（概率30%）

内部类：内部类可以是static的或者非static的，static内部类只能包含静态方法和静态类变量，只能访问外部类的静态元素，内部类可以实例化，多次使用。

匿名内部类：它只能使用一次，不区分static和非static。如果用到外部类的变量的话，必须是类变量或者实例变量，就是必须是类定义的变量，或者final的局部变量。匿名内部类如果是继承某个类的话是可以重写那个类的方法的，这个和普通内部类一样。

实现事件监听器的时候

用匿名内部类编码非常简洁，也容易读懂，不重复利用时使用。

31·&和&&的区别（概率10%）

一个位运算符，一个是逻辑运算符

具体代码中&&具有短路的作用(前面如果条件不成立,后面就不走了), &没有,

19·Final，finally，finalized，区别；（概率30%）

final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。(geitview内部类访问外部类的时候需要加final)

finally是异常处理语句结构的一部分，表示总是执行。(捕获异常的时候用过)

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此

方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

8.ArrayList和LinkedList区别？（概率30%）

存数据，ArrayList**数组存储数据**，索引值以下标来搜索，**查询比较方便，删除增加比较麻烦**，

但是linkedList以**链表式存储数据**，对于增删比较方便。

可以这样说：当操作是在一列数据的后面添加数据而不是在前面或中间,并且需要随机地访问其中的元素时,使用ArrayList会提供比较好的性能；

当你的操作是在一列数据的前面或中间添加或删除数据,并且按照顺序访问其中的元素时,就应该使用LinkedList了。

ArrayList和LinkedList在性能上各 有优缺点，都有各自所适用的地方，总的说来可以描述如下：

对ArrayList和LinkedList而言，在列表末尾增加一个元素所花的开销都是固定的。对 ArrayList而言，主要是在内部数组中增加一项，指向所添加的元素，偶尔可能会导致对数组重新进行分配；而对LinkedList而言，这个开销是 统一的，分配一个内部Entry对象。

在ArrayList的 中间插入或删除一个元素意味着这个列表中剩余的元素都会被移动；而在LinkedList的中间插入或删除一个元素的开销是固定的。

LinkedList不支持高效的随机元素访问。

ArrayList的空间浪费主要体现在在list列表的结尾预留一定的容量空间，而LinkedList的空间花费则体现在它的每一个元素都需要消耗相当的空间

47.Sleep和wait的区别（概率30%）

1.这两个方法来自不同的类分别是，sleep来自Thread类，和wait来自Object

类。

2.最主要是sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可

以使用同步控制块或者方法。sleep不出让系统资源；wait是进入线程等待池

等待，出让系统资源，其他线程可以占用CPU。一般wait不会加时间限制，

因为如果wait线程的运行资源不够，再出来也没用，要等待其他线程调用

notify/notifyAll唤醒等待池中的所有线程，才会进入就绪队列等待OS分配系

统资源。sleep(milliseconds)可以用时间指定使它自动唤醒过来，如果时间不到

只能调用interrupt()强行打断。

3.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而

sleep可以在任何地方使用

4. Sleep需要捕获异常,而wait不需要