**1.复制对象和复制引用？**

（1）复制引用：

通俗的说就是new的对象再引用一次：

person p=new person（）；

person p1=p；

这两个对象打印出来的地址是一样的，就是将堆内的对象地址拷贝一份给了引用对象。

（2）复制对象：

通俗的说就是通过clone的方式创建一个新的对象：

person p=new person();

person p1=(person)p.clone();

这两个对象打印的地址是不一样的，clone的方式是在堆区创建一个新的对象，然后将这个对象的地址复制给p1的，所以这是建立的新的对象。

**2.为什么要用到clone？**

解释：在Java语言的编写过程中，我们可能会遇到需要用某个对象A，A中已经有一些写好的属性，通过复制得到了B，但是要改变B而不改变A这个一般的赋值是做不到的，当然有很多其他方法，但是clone是最简单有效的方法，他可以做到在改变B的同时，不改变A对象。

**3. 两个参数相同（x.equals(y)==true），但是会有不同的hshcode，对不对？（推荐书籍：《EffectiveJava》、《Java 编程思想》）？**

不对。Java对equals和hashcode是这样规定的：x.equals(y)，那么x和y的hashcode就是相同的，但是x和y的hashcode的相同，他两个不一定相等。

equals 方法必须满足自反性（x.equals(x)必须返回 true）、对称性（x.equals(y)返回

true 时，y.equals(x)也必须返回 true）、传递性（x.equals(y)和 y.equals(z)都返回 true x.equals(z)

也必须返回 true）和一致性（当x 和 y 引用的对象信息没有被修改时，多次调用 x.equals(y)应该得到同样的返回值），而且对于任何非 null 值的引用 x，x.equals(null)必须返回 false。

1. 使用==操作符检查参数是否为这个对象的引用；

2.使用 instanceof 操作符检查"参数是否为正确的类型"；

3.对于类中的关键属性，检查参数传入对象的属性是否与之相匹配；

4.编写完 equals 方法后，问自己它是否满足对称性、传递性、一致性；

5.重写 equals 时总是要重写 hashCode；

6.不要将 equals 方法参数中的 Object 对象替换为其他的类型，在重写时不要忘@Overrid解。

**4.为什么函数不能根据返回值来判断重载？**

因为调用时不指定类型信息，编译器不知道你要调用那个函数。例如：当调用max(1,2)时，无法知道要调用哪一个，从这一点上来看，返回值是不足以作为判断重载的条件。

函数的返回值只是作为函数运行之后的一个“状态”，他是保持方法的调用者与被调用者进行通信的关键。并不能作为某个方法的“标识”。

**5.int 和Integer的一些关系？**

public static void test() {

Integer i=new Integer(2);

int i1=2;

Integer i3=2;

System.out.println(i==i1);//true

System.out.println(i1==i3);//true

System.out.println(i==i3);//false

}

Integer是int的包装类，当Integer和Integer进行比较时，会先进行地址的比较，当Integer和int进行比较时，Integer会先自动拆箱变成int类型再比较。

public static void test1() {

Integer a=127;

Integer b=127;

Integer c=128;

Integer d=128;

System.out.println(a==b);//true

System.out.println(c==d);//false

}

简单的说，如果整型字面量的值在-128 到 127 之间，那么不会 new 新的 Integer 对象，

而是直接引用常量池中的 Integer 对象，所以上面的a==b 的结果是 true，而 c==d的结果是 false。

**6.list三个子类的特点？**

ArrayList 底层结构是数组,底层查询快,增删慢。

LinkedList 底层结构是链表型的,增删快,查询慢。

voctor 底层结构是数组线程安全的,增删慢,查询慢。

**7. java中反射的理解？**

Java 中的反射首先是能够获取到 Java 中要反射类的字节码，获取字节码有三种方法，

1.Class.forName(className)

2. 类名.class3.this.getClass()。然后将字节码中的方法，变量，构造函数等映射成相应的

Method、Filed、Constructor 等类，这些类提供了丰富的方法可以被我们所使用。

**8.什么是线程池，如何使用？**

线程池就是事先将多个线程对象放到一个容器中，当使用的时候就不用 new 线程而是直接去池中拿线程即可，节省了开辟子线程的时间，提高的代码执行效率。

在 JDK 的 java.util.concurrent.Executors 中提供了生成多种线程池的静态方法。

1.ExecutorServicenewCachedThreadPool=Executors.newCachedThreadPool();

2.ExecutorServicenewFixedThreadPool=Executors.newFixedThreadPool(4);

3.ScheduledExecutorServicenewScheduledThreadPool=Executors.newScheduledThreadPool(4);

4.ExecutorServicenewSingleThreadExecutor=Executors.newSingleThreadExecutor();

然后调用他们的 execute 方法即可。

**9. 常用的线程池有哪些？**

newSingleThreadExecutor：创建一个单线程的线程池，此线程池保证所有任务的执行顺序按照任务的提交顺序执行。

newFixedThreadPool：创建固定大小的线程池，每次提交一个任务就创建一个线程，直到线程达到线程池的最大大小。

newCachedThreadPool：创建一个可缓存的线程池，此线程池不会对线程池大小做限制，线程池大小完全依赖于操作系统（或者说 JVM）能够创建的最大线程大小。

newScheduledThreadPool：创建一个大小无限的线程池，此线程池支持定时以及周期性执行任务的需求。

**10.使用线程池的好处？**

（如果问到了这样的问题，可以展开的说下线程池如何用、线程池的好处、线程池的启动

策略）合理利用线程池能够带来三个好处。

第一：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。第二：

提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。

第三：提高线程的可管理性。线程是稀缺资源，如果无限制的创建，不仅会消耗系统资源，

还会降低系统的稳定性，使用线程池可以进行统一的分配，调优和监控。