一、Object对象方法有哪些？（高御科技）

> 非常常规的问题，打开IDEA直接瞜一眼
>
> - getClass() - 获取运行是的Class对象，属于反射的内一套，获取运行是的一些数据。
> - hashCode() - 返回对象的hash值。目的是为了更好的支持哈希表，比如基于Java中的HashMap使用
> - equals() - 比较两个对象是否相等，默认 ==
> - clone() - 创建对象的副本。深拷贝和浅拷贝的内容
>   - 默认是浅拷贝，将当前对象复制一份，其中的基本数据类型直接复用值，引用数据类型是复用地址值。
>   - 深拷贝，将当前对象复制一份，其中的基本数据类型直接复用值，引用数据类型会重新的创建一个，不会复制之前的地址。 深拷贝要自己编写克隆内部的引用类型对象。
> - toString - 返回一个以字符串形式表示当前对象的信息。
> - wait - 当某个线程持有当前对象锁时，可以执行对象锁.wait，将持有对象锁的线程挂起等待。
> - notify - 当某个线程持有当前对象锁时，可以执行对象锁.notify，唤醒之前基于wait挂起的一个线程。
> - notifyAll - 当某个线程持有当前对象锁时，可以执行对象锁.notifyAll方法，唤醒之前所有基于wait挂起的线程。
> - finalize - 当触发垃圾回收时，如果当前对象无法基于可达性分析定位到，就会被垃圾回收器回收掉，在回收之前，如果这个对象重写了finalize，那就会触发finalize方法执行。可以执行一些其他的清理工作。（Finalize在JVM中，他不保证一定执行，他用的守护线程）

二、创建对象的方式？（高御科技）

> 首先，这个问题意义不大，更多的跟前面的Object类似，是一个引导向的问题。
>
> - new 关键字。
> - Construct，利用Construct的newInstance去构建对象。 **（引导Spring~）**
> - 反序列化方式：
>   - 流反序列化。
>   - JSON字符串反序列化等等.....
> - clone，克隆也是ok的。
> - 工厂模式去构建对象。 **（引导Spring~）**

三、为啥要有clone？（高御科技）

> Java中提供的clone方法主要是允许将一个对象进行copy，做到快速的复制一个对象。
>
> 而不需要new一个空对象，然后自己慢慢导入数据，开发效率太低。
>
> clone也可以简化一些代码。
>
> 如果再问clone，基本就是聊聊深拷贝，和浅拷贝。
>
> - 默认是浅拷贝，将当前对象复制一份，其中的基本数据类型直接复用值，引用数据类型是复用地址值。

- 深拷贝，将当前对象复制一份，其中的基本数据类型直接复用值，引用数据类型会重新的创建一个，不会复制之前的地址。 深拷贝要自己编写克隆内部的引用类型对象。

四、构造器是否可以被重写？（弘玑）

这个问题明显就是在问方法重载和方法重写的问题。

构造器是不可以重写的，每个类的构造方法都是当前类名。

但是构造器是可以重载的。你写的内些有参，我无参不都是方法重载么~~

- 重写（Overrides）：是指子类提供父类中声明的方法的具体实现。也可以覆盖父类中提供好的实现内容。 **要求是：方法声明都完全一致，除了访问修饰符不能更低。**
- 重载：是指一个类中，定义多个名字相同的方法，但是方法的参数不一样，与返回结果无关。

五、跳出循环？如何在内层循环跳出外层循环？（上海天正）

很基础的问题：

- continue：结束当次循环，继续下次循环。
- break：结束循环。

在内层跳出外层，专业点就加锚点，加标签，其实就是起个名字。

咱们可以break或者continue的时候，指定要你要跳出的名字。就可以了，如下。

```
public static void main(String[] args) {
 outter: for(;;){
     inner: for(;;){
         break outter;
     }
 }
 xxx: if(true){
     //111
     //222
     break xxx;
 }
 yyy: while(true){
     break yyy;
 }
}
```

六、什么是泛型，怎么使用？（鸿盛天极）

比如List集合啥的，都定义了一个泛型，咱们可以在new的时候，将泛型具体化。

Java中的泛型就是一种规避类型错误的一种安全机制。

```
ClassCastException
```

当你在声明一个集合并且指定泛型的类型后，你存储的数据类型就会有限制了，存储的不允许的类型，在编译时期就会直接报错了。

- 类型的安全，前面可以看出来。

- 减少强制类型转换的操作

后面一半会衍生出来 **泛型擦除** 的问题。

泛型擦出去是编译器在编译是的一种机制，你编写的Java代码变为class文件后，他就没有泛型这个东西了，底层都是最顶级的Object。所以泛型只是在编写时加上，其次底层运行时，压根就没泛型这个东西。

就比如List运行后，他就是List。

泛型擦除有什么影响嘛?

反射的时候，获取一些实例的时候，需要手动强转一下，满不爽的。。。

七、Java类的加载顺序？（老虎证券）

我在听录音听到这个问题的时候我懵了。

这个问题问的很奇怪，具体想聊的估计只有两个

- **类加载过程**
  - 加载： 先找到字节码文件（.class文件），加载到JVM内存中的方法区里。 然后在内存中的体现就是一个Class对象。
  - 验证： 验证加载到内存里的.class文件是否被篡改过，确认没有安全问题，以及符合JVM规范。
  - 准备： 为类中的一些变量分配内存空间，并且设置一下默认值。
  - 解析： 将常量池内的符号引用转为直接引用。
    - 符号引用：符号引用是一种泛指，com/mashibing/A-findAll()void
    - 直接引用：直接指向的内库的具体位置，直接就是内存偏移量。后面调用会更快。
  - 初始化：对所有静态变量复制，执行静态代码块，初始化好父类~~
  - 前面走完，到这，这个.class就可以在Java程序中使用了，new一个对象，类名.静态方法都可以了
- **双亲委派**
  - 他其实就是加载这个过程的细节，需要先掌握一下Java中默认的三种类加载器
    - BootstrapClassLoader：负责加载jdk/jre/lib/rt.jar
    - ExtensionClassLoader：负责加载jdk/jre/lib/ext目录下的jar文件
    - ApplicationClassLoader：负责加载classpath目录下的各种class文件。 所谓的classpath，其实就是编译后的classes目录。
    - 其实还有一个自定义的，你自己去继承ClassLoader，重新他的方法，指定你要加载的位置。
  - 双亲委派的过程。当需要用到某个class文件时，撇掉自定义类加载器，他会按照这个方式去加载
    - 先调度AppClassLoader，先查看AppClassLoader加载过么？没加载过，往上问。
    - 问到ExtClassLoader，先查看ExtClassLoader加载过么？没加载过，网上问。
    - 最终问到BootstrapClassLoader，先查看BootstrapClassLoader加载过么？没加载过，尝试加载！如果rt.jar里没有这个.class文件可以加载，往下分配。
    - 分配到ExtClassLoader，他去尝试在ext目录下去加载，如果也没加载到，往下分配。

        - 最终分配到AppClassLoader，他尝试去classpath目录下找这个.class文件加载。
        - 如果没找到，也没加载到，抛一个异常，ClassNotFoundException。
    - 双亲委派解决了什么问题，搞的这么麻烦？？
        - 防止类的重复加载……
        - 防止你破坏JDK的结构……
            - 比如现在我要加载一个java.lang.String这个类！

八、两个List的交集元素？（闪送）

> 首先，Java中就提供了一个API，让咱们可以直接使用的。
>
> 提供的方式叫retainAll方法。
>
> 可以直接 list1.retainAll(list2)，他会将list1中的数据，只保留list1和list2的交集结果。
>
> 他内部采用的机制是双指针。
>
> - 指针A：初始在0位置，如果有交集元素，赋值，然后向后移动一个位置
> - 指针B：每次都会往后移动一个位置指向一个元素，同时利用contains判断元素是否存在，存在，在指针A的位置赋值，然后指针A移动，指针B也移动。如果不存在交集情况，A不动，B动。
>
> ---
>
> 如果不让用Java中提供的API，那就自己搞个Hash表，比如HashMap似的东西，利用计数器的方式去解决这个问题。
>
> 遍历list1，在哈希表里存储key = 值，value = 1。
>
> 遍历list2，直接哈希表里去get，如果拿到的是1，证明存在，是交集数据，存到一个临时的集合中。
>
> 临时的集合，放着所有交集的数据。
>
> O(m + n)