1. 单例模式的对象，在spring中直接实例化，prototype模式（[Spring中原型prototype的准确使用](https://blog.csdn.net/unifirst/article/details/50482031)）的会检查引用，如果有引用对象没被实例化，暂停当前对象的实例化，去实例化引用对象，目的是避免循环引用，单例模式不做这种检查？为什么
2. Spring Bean的作用域 （scope）：

Singleton：Bean在spring ioc容器仅存在一个实例，bean以单例方式存在，bean作用域访问的默认值；

Prototype：每次从容器种调用bean时，都返回一个新的实例，即每次调用getBean()时，相当于执行newXxxBean();

Request: 每次HTTP请求都会创建一个新的bean，该作用域仅适用于web的Spring WebApplictionContext环境；

Session：同一个HTTP session共享一个bean，不同session使用不同的bean，该作用域仅适用于web的Spring WebApplictionContext环境。

1. 单例对象的引用也是单例
2. 三次握手四次挥手，有借有还，不然资源会耗尽；
3. 三次握手>数据传输>四次挥手，称为一个最小粒度，不可被分割
4. 应用层：http/ssh，包装数据 -> 传输控制层：tcp/udp，连接网络 -> 网络层:ip，根据路由表找到下一跳的ip，最终找到目标ip -> 链路层：arp协议，列表映射ip地址和mac地址，通过mac地址找到下一跳的设备 -> 物理层：连接设备
5. Ip是端点间的，mac是节点间的
6. 负载均衡服务器特别的快，四层负载（没有网络层），数据包转发级别，不会和client握手，后端服务器是镜像的
7. DR模式要对arp协议修改,不广播自己的ip地址、mac地址？那负载均衡服务器不也拿不到了
8. 代码执行时间短，竞争线程少，用自旋，否则用系统锁（synchronized）
9. ReentrantLock内部通过cas加锁，state前加了volatile
10. 源码阅读原则：跑不起来不读、解决问题就好（要有目的性）、一条线索到底、无关细节略过
11. Spring的声明式事务使用ThreadLocal既可以使用全局变量来保证事务都在同一连接里，也可以保证每个线程都有自己的连接
12. ThreadLocal使用完要回收掉（remove（）），不然key（弱引用）变成null，value还保持强引用，永远无法回收，会引发内存泄漏
13. ThreadLocal，线程自己的map（ThreadLocal.ThreadLocalMap threadLocals）里存放线程独立的数据，ThreadLocal实例是作为map的key来使用的（<https://www.iteye.com/topic/103804>）
14. Volatile两个特性：禁止内存重排序，保证内存可见性
15. Cpu缓存数据一致性，以前通过总线锁实现（bus）,现在通过缓存锁（MESI-缓存一致性协议）+总线锁（一个缓存行可能放不下一次需要的数据-64字节）
16. 对象的创建过程：1.class loading 2.class linking(verification,preparation,resolution) 3.class initializing 4.申请对象内存 5.成员变量赋默认值 6.调用构造方法<init> （1.成员变量顺序赋初始值 （2.执行构造方法语句
17. Hotspot 通过直接指针进行对象定位？