1、Synchronized在哪里标记锁状态->markword ->除了末两位还存储那些信息？(当时只记得有现成线程信息)->在没有锁的情况下，存储内容

JDK8 markword实现表：

锁状态	25bit		4bit	1bit	2bit
	23bit	2bit		是否偏向锁	锁标志位
无锁态	对象的hashCode		分代年龄	0	01
轻量级锁	指向栈中锁记录的指针				00
重量级锁	指向互斥量(重量级锁)的指针				10
GC标记	空				11
偏向锁	线程ID	Epoch	分代年龄	1	01

2、service hashMap 操作是否有线程安全性问题?

hashMap 中 key是否可以用 object 比如user?

hashMap 大小为啥是2的N次方？？？这个我有点迷？为啥不是2的N次方。。没get到点。

hashMap 如何确定存储的下标　计算hash 然后取模

3、mysql 事务隔离级别，重复读业务场景解释

read uncommited ：读到未提交数据

read committed：脏读，不可重复读

repeatable read：可重读

serializable ：串行事物

阿朱补充->MVCC 内部具体实现机制， redo undo binlog 具体分别是干嘛的，如何保证不可重复读

Repeatable Read（可重读） >> 这是 MySQL 的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会看到同样的数据行。不过理论上，这会导致另一个棘手的问题：幻读（Phantom Read）。简单的说，幻读指当用户读取某一范围的数据行时，另一个事务又在该范围内插入了新行，当用户再读取该范围的数据行时，会发现有新的“幻影” 行。InnoDB 和 Falcon 存储引擎通过多版本并发控制（MVCC，Multiversion Concurrency Control 间隙锁）机制解决了该问题。注：其实多版本只是解决不可重复读问题，而加上间隙锁（也就是它这里所谓的并发控制）才解决了幻读问题

4、sql优化->索引->explain->除了是否用到索引,其他查看项？->mysql索引在什么时候失效？where 后条件- >聚合索引 ->a,b,c 当查询 只有 a,c时 索引是否有效

5、count(*) 、count(1)、count(列)

执行效果上：

count(*)包括了所有的列，相当于行数，在统计结果的时候，不会忽略列值为NULL

count(1)包括了忽略所有列，用1代表代码行，在统计结果的时候，不会忽略列值为NULL

count(列名)只包括列名那一列，在统计结果的时候，会忽略列值为空（这里的空不是只空字符串或者0，而是表示null）的计数，即某个字段值为NULL时，不统计。

执行效率上：

列名为主键，count(列名)会比count(1)快

列名不为主键，count(1)会比count(列名)快

如果表多个列并且没有主键，则 count（1） 的执行效率优于 count（*）

如果有主键，则 select count（主键）的执行效率是最优的

如果表只有一个字段，则 select count（*）最优。

5、对象的生成方式：IOC，反射（Class类的newInstance）、new 其他那？只想到class.newinstance -->动态代理 也是反射。估计想要说序列化创建。

反射：Class类的newInstance、Constructor类的newInstance方法、采用clone、采用序列化机制

1.new

2.clone

3.newInstance

4.反序列化

5.String s = "abc"（这个是比较特殊的）

6、项目问题 数据量-->这个我回答了目前生产的服务器数量

7、个人在团队中定位

1、mysql索引为什选择B+树？

2、技术名称：聚簇索引、回表、索引覆盖、最左匹配、索引下推