1. 什么是Java虚拟机？为什么Java被称作是“平台无关的编程语言”？

Java虚拟机是一个可以执行Java字节码的虚拟机进程。Java源文件被编译成能被Java虚拟机执行的字节码文件。

Java被设计成允许应用程序可以运行在任意的平台，而不需要程序员为每一个平台单独重写或者是重新编译。Java虚拟机让这个变为可能，因为它知道底层硬件平台的指令长度和其他特性。

2. JDK和JRE的区别是什么?

JRE： Java Runtime Environment

JDK：Java Development Kit

JRE顾名思义是java运行时环境，包含了java虚拟机，java基础类库。是使用java语言编写的程序运行所需要的软件环境，是提供给想运行java程序的用户使用的。

JDK顾名思义是java开发工具包，是程序员使用java语言编写java程序所需的开发工具包，是提供给程序员使用的。JDK包含了JRE，同时还包含了编译java源码的编译器javac，还包含了很多java程序调试和分析的工具：jconsole，jvisualvm等工具软件，还包含了java程序编写所需的文档和demo例子程序。

如果你需要运行java程序，只需安装JRE就可以了。如果你需要编写java程序，需要安装JDK。

JRE根据不同操作系统（如：windows，linux等）和不同JRE提供商（IBM,ORACLE等）有很多版本

再说说java的跨平台吧:

java源程序先经过javac编译器编译成二进制的.class字节码文件（java的跨平台指的就是.class字节码文件的跨平台，.class字节码文件是与平台无关的），.class文件再运行在jvm上，java解释器（jvm的一部分）会将其解释成对应平台的机器码执行，所以java所谓的跨平台就是在不同平台上安装了不同的jvm，而在不同平台上生成的.class文件都是一样的，而.class文件再由对应平台的jvm解释成对应平台的机器码执行

最后解释下机器码和字节码的区别：

一，机器码，完全依附硬件而存在～并且不同硬件由于内嵌指令集不同，即使相同的0 1代码

意思也可能是不同的～换句话说，根本不存在跨平台性～比如～不同型号的CPU,你给他个指令10001101，他们可能会解析为不同的结果～

二，我们知道JAVA是跨平台的，为什么呢？因为他有一个jvm,不论那种硬件，只要你装有jvm,那么他就认识这个JAVA字节码～～～～至于底层的机器码，咱不用管，有jvm搞定，他会把字节码再翻译成所在机器认识的机器码～～～

3. ”static”关键字是什么意思？Java中是否可以覆盖(override)一个private或者是static的方法?

“static”关键字表明一个成员变量或者是成员方法可以在没有所属的类的实例变量的情况下被访问。

Java中static方法不能被覆盖，因为方法覆盖是基于运行时动态绑定的，而static方法是编译时静态绑定的。static方法跟类的任何实例都不相关，所以概念上不适用。

java中也不可以覆盖private的方法，因为private修饰的变量和方法只能在当前类中使用，如果是其他的类继承当前类是不能访问到private变量或方法的，当然也不能覆盖。

4. 是否可以在static环境中访问非static变量?

static变量在Java中是属于类的，它在所有的实例中的值是一样的。当类被Java虚拟机载入的时候，会对static变量进行初始化。如果你的代码尝试不用实例来访问非static的变量，编译器会报错，因为这些变量还没有被创建出来，还没有跟任何实例关联上。

5. Java支持的数据类型有哪些？什么是自动拆装箱?

Java语言支持的8种基本数据类型是：

byte
short
int
long
float
double
boolean
char

自动装箱是Java编译器在基本数据类型和对应的对象包装类型之间做的一个转化。比如：把int转化成Integer，double转化成Double，等等。反之就是自动拆箱。

Java支持的数据类型包括两种：一种是基本数据类型，包含byte，char,short, boolean ,int , long, float,double;另一种是引用类型：如String等，其实是对象的引用，JVM中虚拟栈中存的是对象的地址，创建的对象实质在堆中，通过地址来找到堆中的对象的过程，即为引用类型。自动装箱就是Java编译器在基本数据类型和对应的对象包装类型间的转化，即int转化为Integer,自动拆箱是Integer调用其方法将其转化为int的过程。

6. Java中的方法覆盖(Overriding)和方法重载(Overload)是什么意思?

Java中的方法重载发生在同一个类里面两个或者是多个方法的方法名相同但是参数不同的情况。与此相对，方法覆盖是说子类重新定义了父类的方法。方法覆盖必须有相同的方法名，参数列表和返回类型。覆盖者可能不会限制它所覆盖的方法的访问。

7. Java中，什么是构造方法？什么是构造方法重载？什么是复制构造方法?

当新对象被创建的时候，构造方法会被调用。每一个类都有构造方法。在程序员没有给类提供构造方法的情况下，Java编译器会为这个类创建一个默认的构造方法。

Java中构造方法重载和方法重载很相似。可以为一个类创建多个构造方法。每一个构造方法必须有它自己唯一的参数列表。

Java不支持像C++中那样的复制构造方法，这个不同点是因为如果你不自己写构造方法的情况下，Java不会创建默认的复制构造方法。

8. Java支持多继承么?

Java中类不支持多继承，只支持单继承（即一个类只有一个父类）。 但是java中的接口支持多继承，，即一个子接口可以有多个父接口。（接口的作用是用来扩展对象的功能，一个子接口继承多个父接口，说明子接口扩展了多个功能，当类实现接口时，类就扩展了相应的功能）。

9. 接口和抽象类的区别是什么?

Java提供和支持创建抽象类和接口。它们的实现有共同点，不同点在于：

接口中所有的方法隐含的都是抽象的。而抽象类则可以同时包含抽象和非抽象的方法。
类可以实现很多个接口，但是只能继承一个抽象类
类可以不实现抽象类和接口声明的所有方法，当然，在这种情况下，类也必须得声明成是抽象的。
抽象类可以在不提供接口方法实现的情况下实现接口。
Java接口中声明的变量默认都是final的。抽象类可以包含非final的变量。
Java接口中的成员函数默认是public的。抽象类的成员函数可以是private，protected或者是public。
接口是绝对抽象的，不可以被实例化，抽象类也不可以被实例化。

也可以参考JDK8中抽象类和接口的区别

10. 进程和线程的区别是什么?

进程是执行着的应用程序，而线程是进程内部的一个执行序列。一个进程可以有多个线程。线程又叫做轻量级进程。

线程与进程的区别归纳：
a.地址空间和其它资源：进程间相互独立，同一进程的各线程间共享。某进程内的线程在其它进程不可见。
b.通信：进程间通信IPC，线程间可以直接读写进程数据段（如全局变量）来进行通信——需要进程同步和互斥手段的辅助，以保证数据的一致性。
c.调度和切换：线程上下文切换比进程上下文切换要快得多。
d.在多线程OS中，进程不是一个可执行的实体。

11. 创建线程有几种不同的方式？你喜欢哪一种？为什么?

有4种方式可以用来创建线程：

继承Thread类

实现Runnable接口

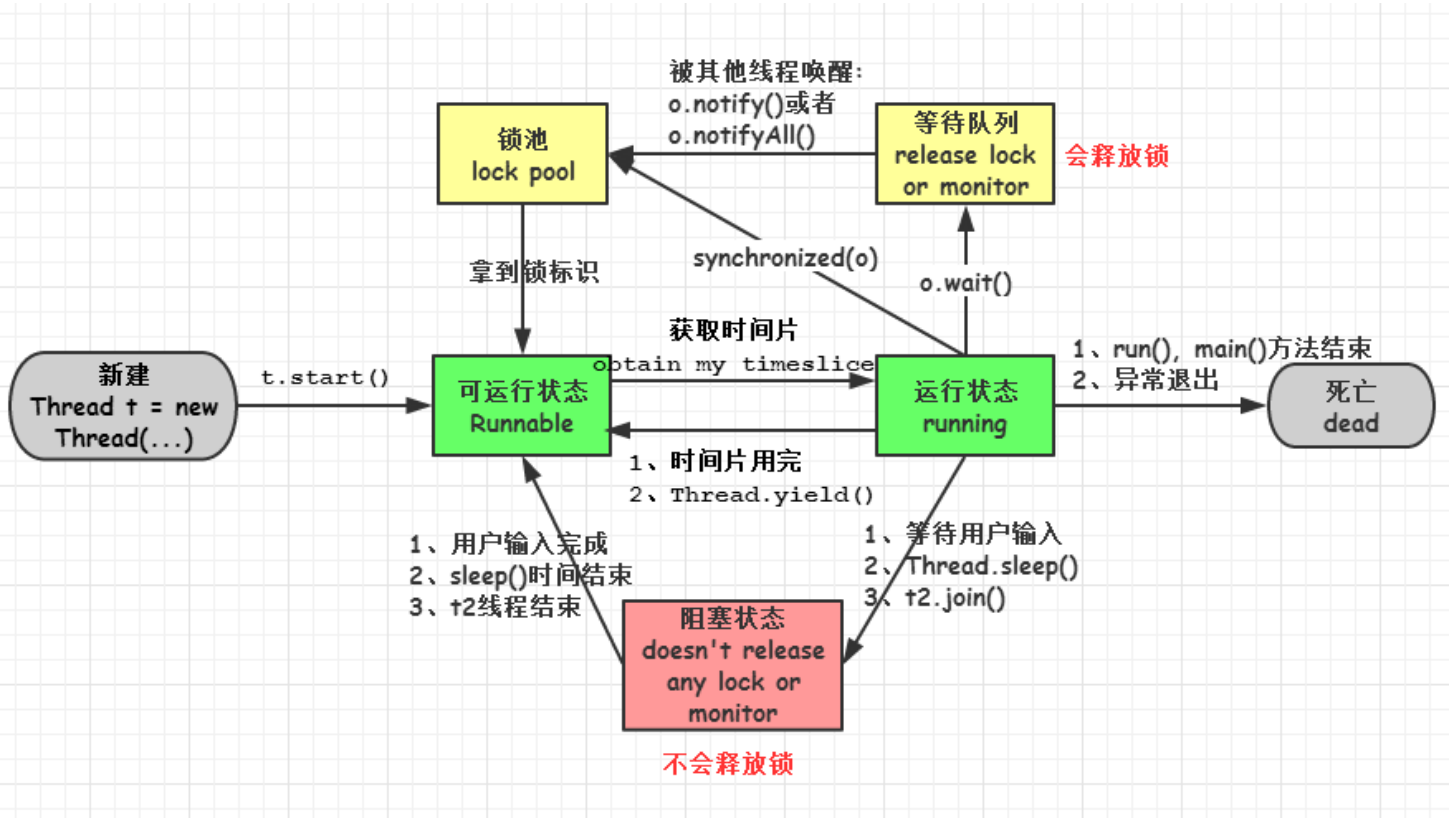
应用程序可以使用Executor框架来创建线程池

实现Runnable接口这种方式更受欢迎，因为这不需要继承Thread类。在应用设计中已经继承了别的对象的情况下，这需要多继承（而Java不支持多继承），只能实现接口。同时，线程池也是非常高效的，很容易实现和使用。

还有一种方式是实现Callable接口

12. 概括的解释下线程的几种可用状态。

1. 新建(new)：新创建了一个线程对象。
2. 可运行(runnable)：线程对象创建后，其他线程(比如 main 线程）调用了该对象 的 start ()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获 取 cpu 的使用权 。
3. 运行(running)：可运行状态(runnable)的线程获得了 cpu 时间片 （ timeslice ） ，执行程序代码。
4. 阻塞(block)：阻塞状态是指线程因为某种原因放弃了 cpu 使用权，也即让出了 cpu timeslice ， 暂时停止运行。直到线程进入可运行(runnable)状态，才有 机会再次获得 cpu timeslice 转到运行(running)状态。阻塞的情况分三种：
(一). 等待阻塞：运行(running)的线程执行 o . wait ()方法， JVM 会把该线程放 入等待队列(waitting queue)中。
(二). 同步阻塞：运行(running)的线程在获取对象的同步锁时，若该同步锁 被别的线程占用，则 JVM 会把该线程放入锁池(lock pool)中。
(三). 其他阻塞: 运行(running)的线程执行 Thread . sleep (long ms)或 t . join ()方法，或者发出了 I / O 请求时， JVM 会把该线程置为阻塞状态。 当 sleep ()状态超时、 join ()等待线程终止或者超时、或者 I / O 处理完毕时，线程重新转入可运行(runnable)状态。
5. 死亡(dead)：线程 run ()、 main () 方法执行结束，或者因异常退出了 run ()方法，则该线程结束生命周期。死亡的线程不可再次复生。



13. 同步方法和同步代码块的区别是什么?

区别:

同步方法默认用this或者当前类class对象作为锁;
同步代码块可以选择以什么来加锁，比同步方法要更细颗粒度，我们可以选择只同步会发生同步问题的部分代

码而不是整个方法；
同步方法使用关键字 synchronized修饰方法，而同步代码块主要是修饰需要进行同步的代码，用synchronized（object）{代码内容}进行修饰；

14. 在监视器(Monitor)内部，是如何做线程同步的？程序应该做哪种级别的同步？

监视器和锁在Java虚拟机中是一块使用的。监视器监视一块同步代码块，确保一次只有一个线程执行同步代码块。每一个监视器都和一个对象引用相关联。线程在获取锁之前不允许执行同步代码。

15. 什么是死锁(deadlock)？

所谓死锁是指多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程都将无法向前推进。死锁产生的4个必要条件：

互斥条件：进程要求对所分配的资源（如打印机）进行排他性控制，即在一段时间内某 资源仅为一个进程所占有。此时若有其他进程请求该资源，则请求进程只能等待。
不剥夺条件：进程所获得的资源在未使用完毕之前，不能被其他进程强行夺走，即只能 由获得该资源的进程自己来释放（只能是主动释放)。
请求和保持条件：进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源 已被其他进程占有，此时请求进程被阻塞，但对自己已获得的资源保持不放。
循环等待条件：存在一种进程资源的循环等待链，链中每一个进程已获得的资源同时被 链中下一个进程所请求。

16. 如何确保N个线程可以访问N个资源同时又不导致死锁？

使用多线程的时候，一种非常简单的避免死锁的方式就是：指定获取锁的顺序，并强制线程按照指定的顺序获取锁。因此，如果所有的线程都是以同样的顺序加锁和释放锁，就不会出现死锁了。

17. Java集合类框架的基本接口有哪些？

集合类接口指定了一组叫做元素的对象。集合类接口的每一种具体的实现类都可以选择以它自己的方式对元素进行保存和排序。有的集合类允许重复的键，有些不允许。

Java集合类提供了一套设计良好的支持对一组对象进行操作的接口和类。Java集合类里面最基本的接口有：

Collection：代表一组对象，每一个对象都是它的子元素。
Set：不包含重复元素的Collection。
List：有顺序的collection，并且可以包含重复元素。
Map：可以把键(key)映射到值(value)的对象，键不能重复。

18. 为什么集合类没有实现Cloneable和Serializable接口？

克隆(cloning)或者是序列化(serialization)的语义和含义是跟具体的实现相关的。因此，应该由集合类的具体实现来决定如何被克隆或者是序列化。

19. 什么是迭代器(Iterator)？

Iterator接口提供了很多对集合元素进行迭代的方法。每一个集合类都包含了可以返回迭代器实例的

迭代方法。迭代器可以在迭代的过程中删除底层集合的元素,但是不可以直接调用集合的

remove(Object Obj)删除，可以通过迭代器的remove()方法删除。

20. Iterator和ListIterator的区别是什么?

下面列出了他们的区别：

Iterator可用来遍历Set和List集合，但是ListIterator只能用来遍历List。

Iterator对集合只能是前向遍历，ListIterator既可以前向也可以后向。

ListIterator实现了Iterator接口，并包含其他的功能，比如：增加元素，替换元素，获取前一个和后一个元素的索引，等等。

21. 快速失败(fail-fast)和安全失败(fail-safe)的区别是什么?

一：快速失败（fail—fast）

在用迭代器遍历一个集合对象时，如果遍历过程中对集合对象的结构进行了修改（增加、删除），则会抛出Concurrent Modification Exception。

原理：迭代器在遍历时直接访问集合中的内容，并且在遍历过程中使用一个 modCount 变量。集合在被遍历期间如果结构发生变化，就会改变modCount的值。每当迭代器使用hashNext()/next()遍历下一个元素之前，都会检测modCount变量是否为expectedmodCount值，是的话就返回遍历；否则抛出异常，终止遍历。

注意：这里异常的抛出条件是检测到 modCount！=expectedmodCount 这个条件。如果集合发生变化时修改modCount值刚好又设置为了expectedmodCount值，则异常不会抛出。因此，不能依赖于这个异常是否抛出而进行并发操作的编程，这个异常只建议用于检测并发修改的bug。

场景：java.util包下的集合类都是快速失败的，不能在多线程下发生并发修改（迭代过程中被修改）。

二：安全失败（fail—safe）

采用安全失败机制的集合容器，在遍历时不是直接在集合内容上访问的，而是先复制原有集合内容，在拷贝的集合上进行遍历。

原理：由于迭代时是对原集合的拷贝进行遍历，所以在遍历过程中对原集合所作的修改并不能被迭代器检测到，所以不会触发Concurrent Modification Exception。

缺点：基于拷贝内容的优点是避免了Concurrent Modification Exception，但同样地，迭代器并不能访问到修改后的内容，即：迭代器遍历的是开始遍历那一刻拿到的集合拷贝，在遍历期间原集合发生的修改迭代器是不知道的。

场景：java.util.concurrent包下的容器都是安全失败，可以在多线程下并发使用，并发修改。

22. Java中的HashMap的工作原理是什么?

Java中的HashMap是以键值对(key-value)的形式存储元素的。HashMap需要一个hash函数，它使用hashCode()和equals()方法来向集合/从集合添加和检索元素。当调用put()方法的时候，HashMap会计算key的hash值，然后把键值对存储在集合中合适的索引上。如果key已经存在了，value会被更新成新值。HashMap的一些重要的特性是它的容量(capacity)，负载因子(load factor)和扩容极限(threshold resizing)。

23. hashCode()和equals()方法的重要性体现在什么地方?

Java中的HashMap使用hashCode()和equals()方法来确定键值对的索引，当根据键获取值的时候也会用到这两个方法。如果没有正确的实现这两个方法，两个不同的键可能会有相同的hash值，因此，可能会被集合认为是相等的。而且，这两个方法也用来发现重复元素。所以这两个方法的实现对HashMap的精确性和正确性是至关重要的。

24. HashMap和Hashtable有什么区别?

HashMap和Hashtable都实现了Map接口，因此很多特性非常相似。但是，他们有以下不同点：

HashMap允许键和值是null，而Hashtable不允许键或者值是null。

Hashtable是同步的，而HashMap不是。因此，HashMap更适合于单线程环境，而Hashtable适合于多线程环境。

HashMap提供了可供应用迭代的键的集合，因此，HashMap是快速失败的。另一方面，Hashtable提供了对键的列举(Enumeration)。

一般认为Hashtable是一个遗留的类。

25. 数组(Array)和列表(ArrayList)有什么区别？什么时候应该使用Array而不是ArrayList?

下面列出了Array和ArrayList的不同点：

Array可以包含基本类型和对象类型，ArrayList只能包含对象类型。

Array大小是固定的，ArrayList的大小是动态变化的。

ArrayList提供了更多的方法和特性，比如：addAll()，removeAll()，iterator()等等。

对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。

26. ArrayList和LinkedList有什么区别?

ArrayList和LinkedList都实现了List接口，他们有以下的不同点：

ArrayList是基于索引的数据接口，它的底层是数组。它可以以O(1)时间复杂度对元素进行随机访问。与此对应，LinkedList是以元素列表的形式存储它的数据，每一个元素都和它的前一个和后一个元素链接在一起，在这种情况下，查找某个元素的时间复杂度是O(n)。

相对于ArrayList，LinkedList的插入，添加，删除操作速度更快，因为当元素被添加到集合任意位置的时候，不需要像数组那样重新计算大小或者是更新索引。

LinkedList比ArrayList更占内存，因为LinkedList为每一个节点存储了两个引用，一个指向前一个元素，一个指向下一个元素。

也可以参考ArrayList vs. LinkedList。

27. Comparable和Comparator接口是干什么的？列出它们的区别。

Java提供了只包含一个compareTo()方法的Comparable接口。这个方法可以个给两个对象排序。具体来说，它返回负数，0，正数来表明已经存在的对象小于，等于，大于输入对象。

Java提供了包含compare()和equals()两个方法的Comparator接口。compare()方法用来给两个输入参数排序，返回负数，0，正数表明第一个参数是小于，等于，大于第二个参数。equals()方法需要一个对象作为参数，它用来决定输入参数是否和comparator相等。只有当输入参数也是一个comparator并且输入参数和当前comparator的排序结果是相同的时候，这个方法才返回true。

28. 什么是Java优先级队列(Priority Queue)?

PriorityQueue是一个基于优先级堆的无界队列，它的元素是按照自然顺序(natural order)排序的。在创建的时候，我们可以给它提供一个负责给元素排序的比较器。PriorityQueue不允许null值，因为他们没有自然顺序，或者说他们没有任何的相关联的比较器。最后，PriorityQueue不是线程安全的，入队和出队的时间复杂度是O(log(n))。

29. 你了解大O符号(big-O notation)么？你能给出不同数据结构的例子么?

大O符号描述了当数据结构里面的元素增加的时候，算法的规模或者是一个渐进上界。

大O符号也可用来描述其他的行为，比如：内存消耗。因为集合类实际上是数据结构，我们一般使用大O符号基于时间，内存和性能来选择最好的实现。大O符号可以对大量数据的性能给出一个很好的说明。

30. 如何权衡是使用无序的数组还是有序的数组?

有序数组最大的好处在于查找的时间复杂度是O(log n)，而无序数组是O(n)。有序数组的缺点是插入操作的时间复杂度是O(n)，因为值大的元素需要往后移动来给新元素腾位置。相反，无序数组的插入时间复杂度是常量O(1)。

31. Java集合类框架的最佳实践有哪些?

根据应用的需要正确选择要使用的集合的类型对性能非常重要，比如：假如元素的数量是固定的，而且能事先知道，我们就应该用Array而不是ArrayList。

有些集合类允许指定初始容量。因此，如果我们能估计出存储的元素的数目，我们可以设置初始容量来避免重新计算hash值或者是扩容。

为了类型安全，可读性和健壮性的原因总是要使用泛型。同时，使用泛型还可以避免运行时的ClassCastException。

使用JDK提供的不变类(immutable class)作为Map的键可以避免为我们自己的类实现hashCode()和equals()方法。

编程的时候接口优于实现。

底层的集合实际上是空的情况下，返回长度是0的集合或者是数组，不要返回null。

32. Enumeration接口和Iterator接口的区别有哪些?

Enumeration速度是Iterator的2倍，同时占用更少的内存。但是，Iterator远远比Enumeration安全，因为其他线程不能够修改正在被iterator遍历的集合里面的对象。同时，Iterator允许调用者删除底层集合里面的元素，这对Enumeration来说是不可能的。

33. HashSet和TreeSet有什么区别?

HashSet是由一个hash表来实现的，因此，它的元素是无序的。add()，remove()，contains()方法的时间复杂度是O(1)。

另一方面，TreeSet是由一个树形的结构来实现的，它里面的元素是有序的。因此，add()，remove()，contains()方法的时间复杂度是O(logn)。

34. Java中垃圾回收有什么目的？什么时候进行垃圾回收?

垃圾回收是在内存中存在没有引用的对象或超过作用域的对象时进行。

垃圾回收的目的是识别并且丢弃应用不再使用的对象来释放和重用资源。

35. System.gc()和Runtime.gc()会做什么事情?

这两个方法用来提示JVM要进行垃圾回收。但是，立即开始还是延迟进行垃圾回收是取决于JVM的。

36. finalize()方法什么时候被调用？析构函数(finalization)的目的是什么?

垃圾回收器(garbage collector)决定回收某对象时，就会运行该对象的finalize()方法 但是在Java中很不幸，如果内存总是充足的，那么垃圾回收可能永远不会进行，也就是说filalize()可能永远不被执行，显然指望它做收尾工作是靠不住的。 那么finalize()究竟是做什么的呢？它最主要的用途是回收特殊渠道申请的内存。Java程序有垃圾回收器，所以一般情况下内存问题不用程序员操心。但有一种JNI(Java Native Interface)调用non-Java程序（C或C++），finalize()的工作就是回收这部分的内存。

37. 如果对象的引用被置为null，垃圾收集器是否会立即释放对象占用的内存?

不会，在下一个垃圾回收周期中，这个对象将是可被回收的。

38. Java堆的结构是什么样子的？什么是堆中的永久代(Perm Gen space)?

JVM的堆是运行时数据区，所有类的实例和数组都是在堆上分配内存。它在JVM启动的时候被创建。对象所占的堆内存是由自动内存管理系统也就是垃圾收集器回收。

堆内存是由存活和死亡的对象组成的。存活的对象是应用可以访问的，不会被垃圾回收。死亡的对象是应用不可访问尚且还没有被垃圾收集器回收掉的对象。一直到垃圾收集器把这些对象回收掉之前，他们会一直占据堆内存空间。

永久代是用于存放静态文件，如Java类、方法等。持久代对垃圾回收没有显著影响，但是有些应用可能动态生成或者调用一些class，例如Hibernate 等，在这种时候需要设置一个比较大的持久代空间来存放这些运行过程中新增的类，永久代中一般包含：

类的方法(字节码...)
类名(Sring对象)
.class文件读到的常量信息
class对象相关的对象列表和类型列表 (e.g., 方法对象的array).
JVM创建的内部对象
JIT编译器优化用的信息

39. 串行(serial)收集器和吞吐量(throughput)收集器的区别是什么?

吞吐量收集器使用并行版本的新生代垃圾收集器，它用于中等规模和大规模数据的应用程序。而串行收集器对大多数的小应用(在现代处理器上需要大概100M左右的内存)就足够了。

40. 在Java中，对象什么时候可以被垃圾回收?

当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

41. JVM的永久代中会发生垃圾回收么?

会，如果永久代满了或者是超过了临界值，会触发完全垃圾回收(Full GC)。如果你仔细查看垃圾收集器的输出信息，就会发现永久代也是被回收的。这就是为什么正确的永久代大小对避免Full GC是非常重要的原因。请参考下Java8：从永久代到元数据区

(注：Java8中已经移除了永久代，新加了一个叫做元数据区的native内存区)

42. Java中的两种异常类型是什么？他们有什么区别?

Java中有两种异常：受检查的(checked)异常和不受检查的(unchecked)异常。不受检查的异常不需要在方法或者是构造函数上声明，就算方法或者是构造函数的执行可能会抛出这样的异常，并且不受检查的异常可以传播到方法或者是构造函数的外面。相反，受检查的异常必须要用throws语句在方法或者是构造函数上声明。这里有Java异常处理的一些小建议。

43. Java中Exception和Error有什么区别?

Exception和Error都是Throwable的子类。Exception用于用户程序可以捕获的异常情况。Error定义了不期望被用户程序捕获的异常。

44. throw和throws有什么区别?

1、Throw用于方法内部，Throws用于方法声明上

2、Throw后跟异常对象，Throws后跟异常类型

3、Throw后只能跟一个异常对象，Throws后可以一次声明多种异常类型

45. 异常处理完成以后，Exception对象会发生什么变化?

Exception对象会在下一个垃圾回收过程中被回收掉。

46. finally代码块和finalize()方法有什么区别?

无论是否抛出异常，finally代码块都会执行，它主要是用来释放应用占用的资源。finalize()方法是Object类的一个protected方法，它是在对象被垃圾回收之前由Java虚拟机来调用的。

47. 什么是Applet?

java applet是能够被包含在HTML页面中并且能被启用了java的客户端浏览器执行的程序。Applet主要用来创建动态交互的web应用程序。

48. 解释一下Applet的生命周期

applet可以经历下面的状态：

Init：每次被载入的时候都会被初始化。

Start：开始执行applet。

Stop：结束执行applet。

Destroy：卸载applet之前，做最后的清理工作。

49. 当applet被载入的时候会发生什么?

首先，创建applet控制类的实例，然后初始化applet，最后开始运行。

50. Applet和普通的Java应用程序有什么区别?

applet是运行在启用了java的浏览器中，Java应用程序是可以在浏览器之外运行的独立的Java程序。但是，它们都需要有Java虚拟机。

进一步来说，Java应用程序需要一个有特定方法签名的main函数来开始执行。Java applet不需要这样的函数来开始执行。

最后，Java applet一般会使用很严格的安全策略，Java应用一般使用比较宽松的安全策略。

51. Java applet有哪些限制条件?

主要是由于安全的原因，给applet施加了以下的限制：

applet不能够载入类库或者定义本地方法。

applet不能在宿主机上读写文件。

applet不能读取特定的系统属性。

applet不能发起网络连接，除非是跟宿主机。

applet不能够开启宿主机上其他任何的程序。

52. 什么是不受信任的applet?

不受信任的applet是不能访问或是执行本地系统文件的Java applet，默认情况下，所有下载的applet都是不受信任的。

53. 从网络上加载的applet和从本地文件系统加载的applet有什么区别?

当applet是从网络上加载的时候，applet是由applet类加载器载入的，它受applet安全管理器的限制。

当applet是从客户端的本地磁盘载入的时候，applet是由文件系统加载器载入的。

从文件系统载入的applet允许在客户端读文件，写文件，加载类库，并且也允许执行其他程序，但是，却通不过字节码校验。

54. applet类加载器是什么？它会做哪些工作?

当applet是从网络上加载的时候，它是由applet类加载器载入的。类加载器有自己的java名称空间等级结构。类加载器会保证来自文件系统的类有唯一的名称空间，来自网络资源的类有唯一的名称空间。

当浏览器通过网络载入applet的时候，applet的类被放置于和applet的源相关联的私有的名称空间中。然后，那些被类加载器载入进来的类都是通过了验证器验证的。验证器会检查类文件格式是否遵守Java语言规范，确保不会出现堆栈溢出(stack overflow)或者下溢(underflow)，传递给字节码指令的参数是正确的。

55. applet安全管理器是什么？它会做哪些工作?

applet安全管理器是给applet施加限制条件的一种机制。浏览器可以只有一个安全管理器。安全管理器在启动的时候被创建，之后不能被替换覆盖或者是扩展。

56. 弹出式选择菜单(Choice)和列表(List)有什么区别

Choice是以一种紧凑的形式展示的，需要下拉才能看到所有的选项。Choice中一次只能选中一个选项。List同时可以有多个元素可见，支持选中一个或者多个元素。

57. 什么是布局管理器?

布局管理器用来在容器中组织组件。

58. 滚动条(Scrollbar)和滚动面板(JScrollPane)有什么区别?

Scrollbar是一个组件，不是容器。而ScrollPane是容器。ScrollPane自己处理滚动事件。

59. 哪些Swing的方法是线程安全的?

只有3个线程安全的方法： repaint(), revalidate(), and invalidate()。

60. 说出三种支持重绘(painting)的组件。

Canvas, Frame, Panel,和Applet支持重绘。

61. 什么是裁剪(clipping)?

限制在一个给定的区域或者形状的绘图操作叫做裁剪。

62. MenuItem和CheckboxMenuItem的区别是什么?

CheckboxMenuItem类继承自MenuItem类，支持菜单选项可以选中或者不选中。

63. 边缘布局(BorderLayout)里面的元素是如何布局的?

BorderLayout里面的元素是按照容器的东西南北中进行布局的。

64. 网格包布局(GridBagLayout)里面的元素是如何布局的?

GridBagLayout里面的元素是按照网格进行布局的。不同大小的元素可能会占据网格的多于1行或一列。因此，行数和列数可以有不同的大小。

65. Window和Frame有什么区别?

Frame类继承了Window类，它定义了一个可以有菜单栏的主应用窗口。

66. 裁剪(clipping)和重绘(repainting)有什么联系?

当窗口被AWT重绘线程进行重绘的时候，它会把裁剪区域设置成需要重绘的窗口的区域。

67. 事件监听器接口(event-listener interface)和事件适配器(event-adapter)有什么关系?

事件监听器接口定义了对特定的事件，事件处理器必须要实现的方法。事件适配器给事件监听器接口提供了默认的实现。

68. GUI组件如何来处理它自己的事件?

GUI组件可以处理它自己的事件，只要它实现相对应的事件监听器接口，并且把自己作为事件监听器。

69. Java的布局管理器比传统的窗口系统有哪些优势?

Java使用布局管理器以一种一致的方式在所有的窗口平台上摆放组件。因为布局管理器不会和组件的绝对大小和位置相绑定，所以他们能够适应跨窗口系统的特定平台的不同。

70. Java的Swing组件使用了哪种设计模式?

Java中的Swing组件使用了MVC（模型-视图-控制器)设计模式