115-Java-Interview-Questions-and-Answers

snowdream

Published
with GitBook

目錄

#115个Java面试题及回答

简介

在本教程中,我们将讨论在Java面试中,用人单位用来测试应聘者Java以及面向对象的能力的面试题目.以下章节我们将按照以下结构讨论面试问题，面向对象编程及其特性，Java及其特性的一般问题,集合,垃圾回收，异常处理,Java applets,Swing,JDBC,RMI, Servlet 和 JSP. 来，我们一起出发吧。

面向对象编程(OOP)

Java是支持并发，基于类的以及面向对象的一种计算机编程语言. 以下列举了面向对象编程的优势:

- 模块化编程，使维护和修改更加容易
- 代码重用
- 提高代码的可靠性以及灵活性
- 提高代码可读性

面向对象编程有其非常明显的特性，比如说封装,继承,多态和抽象. 下面我们来分析一下每种特性.
封装**(Encapsulation)**
封装，提供的对象隐藏内部特性和行为的一种能力，每个对象提供了一些方法，其他的对象可以访问并改变其内部数据。在Java中，提供了三种访问修饰符: 公有的，私有的以及保护的。每个修饰符都设定了不同的访问权限，这个权限设置不会因为包的不同而有差异。
下面是使用封装的一些优点:

- 通过隐藏属性来保护对象的内部信息
- 因为可以独立的修改或者扩展对象的行为，从而提高代码的可用性以及维护性
- 隔离，通过阻止对象使用不希望的互动方式来交互，以此来提高模块化程度。 你可以通过链接访问我们的教程查看关于封装更多的细节和实例.

多态**(Polymorphism)**
多态就是针对不同的基础数据类型呈现相同接口的一种能力，多态类型就是其操作可以适用于不同类型值的一种类型。
继承**(Inheritance)**
继承提供了一个对象从基类获取字段和方法的一种能力.继承提供了代码的重用性，并且在不更改现有类的情况下，对现有类增加额外的功能。
抽象**(Abstraction)**
抽象是从具体的实例中分离想法的过程，根据他们各自的功能而非具体的实现来开发类. Java中支持创建和存在暴露接口的抽象类， 而没有包括方法的具体实现。抽象方法的宗旨就是将类的行为和具体实现分离开。
抽象和封装的异同
抽象和封装是个互补的概念。一方面，抽象专注在对象的行为上，而另外一方面，封装专注于对象的行为的具体实现。封装是通过隐藏对象的内部信息来实现的，因此也可以被看做是抽象的一种策略。

有关Java的一般问题

1. 什么是JVM? 为什么称Java为跨平台的编程语言?

Java虚拟机(Java Virtual Machine)是可以执行Java字节码的虚拟机，每个Java源文件将被编译成字节码文件，然后在JVM中执行。Java之所以被设计成可以在任意的平台运行，而不需要重写或者在不同的平台下重新编译，这些都要归功于Java虚拟机(JVM),因为JVM非常了解特定的指令的长度以及底层硬件平台的特殊性。

2. JDK和JRE之间的差异是什么？

Java运行环境(Java Runtime Enviroment) 是运行Java程序的基本的Java虚拟机，包括执行applet的浏览器插件。JDK (Java Development Kit) 是为了开发，编译和执行Java应用程序，针对Java的全功能的软件开发包，包含了JRE，编译器和工具(比如说 JavaDoc 和Java Debugger)。

3. “static” 关键字是什么意思？在Java里可以 override private 或 static 的方法吗？ keyword mean ? Can you override private or static method in Java ?

static 关键字表示，访问这个成员变量或方法时，不必获取它属于的类的实例。 Java 里的 static 方法不能被 override，因为 override 的机制是运行时(runtime)的动态绑定，而 static 方法是在编译时静态绑定的。static 方法并不与任何类的具体实例有关，因此无法应用继承的概念。

4. 在静态方法里可以访问非静态变量吗?

Java 中的 static 变量归相应的类所有，它的值对于类的所有实例都是相同的。static 变量是在 JVM 加载类的时候初始化的。如果代码试图访问非静态的变量，而且不是通过类的实例去访问，编译器会报错，因为这些非静态变量还没有被创建呢，并且它们没有与实例相关联。

5. Java 支持哪些数据类型？什么是 Autoboxing 和 Unboxing？

Java语言支持的8个基本数据类型如下：

- byte
- short
- int
- long
- float
- double
- boolean
- char

Autoboxing 是指在基本数据类型和对应的包装(wrapper)类之间Java 编译器所做的自动转换。例如，编译器将 int 转换为 Integer，将 double 转换为 [Double])(http://docs.oracle.com/javase/7/docs/api/java/lang/Double.html)，等等。逆向的转换称为 unboxing。

6. 在Java中什么是方法的 Override(覆盖) 和 Overload(重载)?

Java中方法的 overload 发生的条件是，同一个类里，有两个或以上的方法名称完全相同，但参数列表不同。另一方面，方法的 override 是指，子类重定义了父类里的同一个方法。Override 的方法必须方法名、参数列表和返回类型都完全相同。Override 的方法不会限制原方法的访问权限。

7.Java中构造函数、构造函数重载的概念和拷贝构造函数

当类的对象被创建的时候，调用它的构造函数。每个类都有一个构造函数。如果程序员没有为类编写构造函数，Java编译器自动为类创建一个缺省的构造函数。 构造函数重载和Java中函数重载类似，可以为同一个类创建不同的构造函数，每个构造函数必须拥有唯一的参数列表。 Java与C++不同，它不支持拷贝构造函数，但是区别仅仅是，如果你没有编写类的拷贝构造函数，Java不会自动创建它。

8.Java支持多继承吗？

Java不支持多继承，每个类只允许继承一个类，但是可以实现多个接口。

9.接口和抽象类有什么不同？

Java同时提供和支持抽象类和接口，它们的实现有一些共同的特点，也有如下不同：

- 接口中所有的方法默认都是抽象的，而抽象类可以同时包含抽象和非抽象的方法。
- 一个类可以实现多个接口，但它只能继承一个抽象类。
- 一个类要实现某个接口，必须实现这个接口声明的所有方法。而一个类不需要实现抽象父类中声明的所有方法，不过，这时候这个类也必须声明为抽象类。
- 抽象类可以实现接口，而且不需要实现接口中的方法。
- 接口中声明的变量默认是final的，而抽象类可以包含非final的变量。
- 接口中的成员默认是public的，而抽象类的成员可以是private，protected，或public的。
- 接口是绝对抽象的，不可实例化，抽象类也不可以实例化，但可以在main方法中触发实例化（注：通过匿名类实现）。

也可以查阅 Abstract class and Interface differences for JDK 8 。

10.传引用和传值

当对象通过传值调用时，传递的是这个对象的一个拷贝。因此，即使函数修改这个对象，也不会影响原对象的值。
当对象通过传引用调用时，对象本身没有被传递，而传递的是对象的一个引用。因此，外部函数对这个对象的修改，也会反映到任何出现这个对象的地方。

Java 线程

11. 进程与线程的区别？

进程是一个程序的执行(即正在运行的程序), 然而线程是在进程中独立的执行序列. 一个进程可以包含很多线程. 线程有时被称为轻量级的进程.

12. 说下创建线程的不同方式. 你倾向于哪种方式并说明原因？

有三种创建线程的方式:

- 继承Thread类.
- 实现Runnable接口.
- 通过Executor框架创建线程池.

首选方式是实现Runnable接口, 因为它不需要继承Thread类. 当你的程序设计需要多继承时, 使用接口会有所帮助. 另外, 线程池效率是很高的, 并且实施起来也很简单.

13. 解释下可用的线程状态.

在执行期间, 线程会处于以下状态中的一种:

- Runnable: 线程已准备就绪, 但没有立即运行.
- Running: 处理器正在执行的线程代码.
- Waiting: 处于阻塞状态的线程, 等待外部某种处理的结束.
- Sleeping: 被强制休眠的线程.
- Blocked on I/O: 等待I/O操作的完成.
- Blocked on Synchronization: 等待取得线程锁.
- Dead: 线程已经执行结束.

14. 同步方法与同步块的区别？

在Java程序中, 每个对象都拥有一个锁. 线程可以通过使用synchronized关键字来获取一个对象上的锁. synchronized关键字可以用于方法级别(粗粒度锁)或代码块级别(细粒度锁).

15. 在监视器中的线程同步是怎样发生的? 你可以使用哪些级别的同步？

JVM使用结合了监视器的锁. 监视器是一个守护者, 它看管一个同步代码的序列, 并且确保在一个时刻只能有一个线程执行同步代码片段. 每个监视器关联着一个对象引用. 只能得到锁的线程才可以执行同步代码.

16. 什么是死锁？

当两个进程相互等待对方执行完毕时, 其结果是它们会永远等待下去.

17. 怎样确保N个线程访问N个资源时不会发生死锁？

使用N个线程时一个非常简单的避免死锁的方法是为所有的锁排序, 并强制每个线程也按那种方式排序. 这样, 如果所有线程以相同的顺序锁定和解锁互斥资源就不会发生死锁了.

Java Collections

18. Java Collections框架的基本接口？

Java Collections 框架 提供了一系列支持对象集合操作的设计良好的接口和类. Java Collections框架中最基础的接口是:

- Collection, 表示一组对象(元素).
- Set, 包含非重复元素的集合.
- List, 包含非重复元素的有序集合.
- Map, 包含非重复键的键值对的对象.

19. 为什么Collection没有继承Cloneable和Serializable接口？

Collection接口描述的是由元素组成的对象组. Collection的每一个具体实现可以选择自己的方式来管理元素. 一些集合允许存在重复键, 另一些则不允许.
当处理实际实现时复制和序列化的语义和效果才会起作用. 因此, 集合类的具体实现应该决定它们将怎样被复制和序列化.

20. 什么是Iterator(叠代器)？

Iterator接口提供了许多能够叠代集合的方法.每个java集合(Collection)都含有一个返回Iterator实例的iterator方法. 叠代器在叠代过程中能够移除底层集合中的元素.

21. Iterator 和ListIterator之间的不同？

它们之间的不同如下:

- Iterator能够遍历Set和List集合, 而ListIterator只能用来遍历List.
- Iterator只能正向遍历集合, 而ListIterator可以双向遍历List.
- ListIterator实现了Iterator接口并提供了额外的功能, 例如添加元素, 替换元素, 取得上一个或上一个元素索引, 等等.

22. fail-fast与fail-safe的区别?

Iterator的fail_safe特性是对底层集合的拷贝进行操作, 因此对集合的任何改变都不会有影响. java.util包下的所有集合类是fail-fast的, 但java.util.concurrent包下的集合类是fail-safe的. fail-fast叠代器会抛出ConcurrentModificationException, 而fail-safe叠代器不会抛出这种异常.

23. Java中的 HashMap 是怎么工作的?

Java 中的 HashMap 是用来存储键值对的。 HashMap 需要一个 hash 函数，它使用 hashCode 和 equals 方法，来进行 collection 中元素的保存和查找。调用 put 方法时， HashMap 会计算键(key)的 hash 值，然后将键值对存到 collection 的适当索引下。如果键已经存在，那么相应的值会更新。HashMap 的重要特征主要有它的容量，装载因子(load factor)和容量扩充(threshold resizing)。

24. hashCode() 和 equals() 方法重要性何在？

Java中的 HashMap 使用 hashCode 和 equals方法来确定键值对的索引。根据键去查询对应的值时，同样会用到这两个方法。如果这两个方法没有正确实现，两个不同的键可能会产生相同的 hash 值，因此会被 collection 认为是相同的。并且，这两个方法在检测重复时也会用到。因此，这两个方法都要正确实现，对 HashMap 的正确性和效率都至关重要。

25.HashMap与HashTable之间有哪些不同？

HashMap、HashTable这两个类都实现了Map接口，因此有些非常相似的特征，但他们在以下特性中又有所不同：

- HashMap的key与value都允许null值的存在，而HashTable则既不允许key为null，也不允许value为null。
- HashTable是线程同步的，而HashMap则不是。因此，在单线程环境下HashMap是首选，而HashTable更适合在多线程环境下使用。
- HashMap提供了它键的set集合，因此Java程序可以通过set进行迭代。因此，HashMap是快速失败的。另一方面，HashTable提供了它键的枚举。
- HashTable类被当做遗留类（译者注：Java遗留类说明）

26.Array与ArrayList间有什么不同？与ArrayList相比你什么时候会用Array？

Array与ArrayList类在以下特性中有所区别：

- Arrays可以包含基础数据类型或者对象，而ArrayList只能包含对象。
- Arrays有固定长度，而ArrayList长度则是动态的。
- ArrayList类提供了更多的方法和特性，比如addAll,removeAll,iterator,等等
- 对于一个基础数据类型的list，集合框架使用了自动装箱去减少编码的工作。但针对固定长度的的基础数据类型，这种方法会使得它们变得更慢。

27.ArrayList与LinkedList间有什么不同？

ArrayList、LinkedList这两个类都实现了List接口，但他们以下特性中又有所不同：

- ArrayList是基于索引的数据结构，底层由Array支持实现。它提供了以时间复杂度为O(1)的性能随机访问它的元素，另一方面，LinkedList以元素列表的方式来存储它的数据，每一个元素与它前一个和后一个元素都是相连的。对元素查询操作的时间复杂度为O(n).
- 对元素的插入、添加、移除操作，与ArrayList相比，LinkedList更快，因为，当一个元素被添加到集合内部的任意位置时，LinkedList不需要重新调整数组大小或者更新索引。
- LinkedList比ArrayList消耗更多的内存，因为LinkedList中每一个节点都存储了两个引用，一个是它前一个元素，一个是它后一个元素 也可以查看我们的文章ArrayList vs. LinkedList

28. Comparable 和 Comparator 接口分别是什么？列出它们的区别。

Java 提供的 Comparable 接口，其中只包含一个方法，就是 compareTo。这个方法会比较两个对象，从而得出它们的顺序关系。具体来说，它会返回一个负整数，零，或一个正整数，分别表示传入的对象小于，等于或大于已有的对象。 Java 提供的 Comparator 接口，包含两个方法，compare 和 equals 。compare 方法比较两个参数，得出它们的顺序关系。它会返回一个负整数，零，或一个正整数，分别表示第一个参数小于，等于或大于已有的对象。equals 方法有1个参数，它用来确定参数对象是否等于这个 camparator。这个方法仅在要比较的对象也是一个 comparator，同时它的序关系与这个 comparator 相同时，才会返回 true。

29.Java Priority Queue是什么?

PriorityQueue是一个基于优先级堆的无界队列，它的元素都以他们的自然顺序有序排列。在它创建的时候，我们可以可以提供一个比较器(Comparator)来负责PriorityQueue中元素的排序。PriorityQueue不允许null元素，不允许不提供自然排序的对象，也不允许没有任何关联Comparator的对象。最后，PriorityQueue不是线程安全的，在执行入队和出队操作它需要O(log(n))的时间复杂度。

30. 关于Big-O符号你了解些什么？你能针对不同数据结构举些例子吗？

Big-O符号简单描述了，在一种数据结构中随着元素的不断增加，在最坏的情况下，一个算法的扩展或者执行能有多好。Big-O符号还可以用来描述其他的行为，比如内存消耗。由于collection集合下的类实际上都是数据结构，我们通常用Big-O符号来选择基于时间、内存、性能前提下的最好实现去使用。Big-O符号能就大量数据的性能给出一个很好的指示。

31. 如何权衡有序数组和无序数组？

有序数组最关键的优势在于搜索的时间复杂度为 O(log n)，而无序数组的时间复杂度是 O (n)。有序数组的劣势就在于插入操

作的时间复杂度为 O(n)，因为较高值的元素需要挪动位置腾出空间给新元素。与之不同的是，无序数组的插入操作的时间复杂度为 O(1)。

32. 有哪些关于 Java 集合框架的最佳实践？

- 基于应用的需求来选择使用正确类型的集合，这对性能来说是非常重要的。例如，如果元素的大小是固定的，并且知道优先级，我们将会使用一个 Array，而不是 ArrayList。
- 一些集合类允许我们指定他们的初始容量。因此，如果我们知道存储数据的大概数值，就可以避免重散列或者大小的调整。
- 总是使用泛型来保证类型安全，可靠性和健壮性。同时，使用泛型还可以避免运行时的 ClassCastException。
- 在 Map 中使用 Java Development Kit (JDK) 提供的不可变类作为一个 key，这样可以避免 hashCode 的实现和我们自定义类的 equals 方法。
- 应该依照接口而不是实现来编程。
- 返回零长度的集合或者数组，而不是返回一个 null，这样可以防止底层集合是空的。

33. Enumeration 和 Iterator 接口有什么不同？

Enumeration 跟 Iterator 相比较快两倍，而且占用更少的内存。但是，Iterator 相对于 Enumeration 更安全，因为其他线程不能修改当前迭代器遍历的集合对象。同时，Iterators 允许调用者从底层集合中移除元素，这些 Enumerations 都没法完成。

34. HashSet 和TreeSet 有什么不同?

HashSet 是用一个 hash 表来实现的，因此，它的元素是无序的。添加，删除和 HashSet 包括的方法的持续时间复杂度是 O(1)。
另一个方面，TreeSet 是用一个树形结构实现的，因此，它是有序的。添加，删除和 TreeSet 包含的方法的持续时间复杂度是 O(logn)。

垃圾回收器

35. Java中垃圾回收的目的是什么, 它什么时候被使用 ?

垃圾回收用于识别和丢弃程序不再需要的对象, 以便回收和复用资源.

36. System.gc() 和Runtime.gc()方法用途?

这些方法用于提醒JVM开始垃圾回收. 然而开始垃圾回收的时机是由JVM决定的.

37. finalize()什么时候被调用 ? 它的目的是什么 ?

finallize方法是在释放该对象内存前由gc(垃圾回收器)调用. 通常建议在这个方法中释放该对象持有的资源.

38.如果一个对象的引用被设置为null, gc会立即释放该对象的内存么?

不会, 这个对象将会在下一次gc循环中被回收.

39. Java堆的结构是什么 ? 堆中的Perm Gen(全称是Permanent Generation)空间是什么 ?

JVM有一个运行时数据区,即堆(heap).所有的类实例和数组的内存都是从堆中分配的. 它在JVM启动时被创建. 对象所占用的堆内存会被一个称为垃圾回收器的自动内存管理系统收回.
堆内存中包含活的和死的对象. 活的对象可以被程序访问并且不会被垃圾回收. 死的对象是那些不会被程序访问的, 但还没有被垃圾回收器收回的对象. 这种对象会占用堆内存空间直到最终被垃圾回收器收回.

40. Serial 垃圾回收器与 Throughput 垃圾回收器区别 ?

Throughput垃圾回收器使用并行版本的新生代回收器, 它用于中到大型数据集的应用. 另一方面, Serial回收器通常足以应对大多数的小应用(在现代处理器上不会超过约100MB的堆内存).

41. 什么时候对象会被回收 ?

当当前程序无法访问到某个对象时, 该对象将被回收.

42. 垃圾回收发生在指定的JVM区域 ?

垃圾回收在 PermGen 里发生。如果 PermGen满了或是到了瓶颈，就会触发完全回收。如果仔细观察垃圾回收过程，会发现PermGen最后也被回收了. 这就是为什么要设置合理的PermGen大小以避免完全垃圾回收. 可以看看这篇文章 Java 8: PermGen to Metaspace.

异常处理

43. Java中的两种异常是什么?它们之间的区别?

Java有两种类型的异常:checked与unchecked(检查与未检查) 异常. 如果unchecked异常可能会在方法或构造函数的执行时被抛出从而蔓延到方法或构造函数的外部, 它们也不需要要在方法或构造函数中声明throws子句. 然而, checked异常必须通过方法或构造函数的throws子句声明. 关于java异常处理的建议请参考这里Java exception handling.

44. Java中异常与错误的区别?

Exception和Error都是Throwable类的子类. Exception用于用户程序需要捕获的异常条件. Error定义了用户程序不可预见的异常

45. throw与throws的区别 ?

关键字throw用于在程序中显式地抛出一个异常. 相反, throws子句用于指出在该方法中没有处理的异常. 每个方法必须显式指明哪些异常没有处理, 以便该方法的调用者可以预防可能发生的异常. 最后, 多个异常用逗号分隔.

45. 异常处理中finally语句块的重要性?

不管程序是否发生了异常, finally语句块都会被执行. 甚至当没有catch声明但抛出了一个异常时, finally语句块也会被执行. 最后要说一点: finally语句块通常用于释放资源, 如I/O缓冲区, 数据库连接等等.

46. 异常被处理后异常对象会发生什么?

异常对象会在下次gc执行时被回收.

47. 怎样区分finally语句块与finalize()方法?

不管是否抛出异常finally语句块都会被执行, 它通常用于释放程序持有的资源. finalize是Object类中的一个protected方法, 当一个对象被gc回收前它会被jvm调用.

Java Applets

48. 什么是 Applet ?

一个 Java Applet 可以包含在 HTML 页面中并且可以在启用 Java 客户端的浏览器中运行。Applets 可以用来创建动态和交互式的 web 应用程序。

49. Applet 生命周期的说明

一个 Applet 可能会经历以下的几个状态：

- Init: 每次加载的时候都进行初始化
- Start: 开始执行一个 applet
- Stop: 停止执行一个 applet
- Destroy: 卸载 applet 之前执行最后的清理

50. 当 applet 加载的时候会发生什么？

首先，会创建一个 applet 的控制类的实例。然后，applet 初始化，最后 applet 开始运行。

51. Applet和Java应用程序有什么不同?

Applets需要一个支持Java的浏览器, 但是Java应用程序可以被单独执行。但是，他们都需要一个Java虚拟机，JVM。 还有, 一个Java应用程序需要一个main方法与特定的签名，来确保启动. Java applets并不需要这样一些东西。 最后, Java Applet通常使用严格的安全策略，而Java应用程序通常使用较宽松的安全策略。

52. 什么是强加给Java Applet限制？

这主要是由于安全原因，以下限制强加在Java小程序：

- Applet无法加载库或定义本地方法。
- Applet 通常无法读取或执行主机的文件。
- Applet无法读取某些系统属性。
- 除非连接来自主机否则不能进行网络连接。
- Applet程序不能启动主机上执行它的任何程序。

53. 什么是不可信的Applet ?

不受信任的Applet是那些Java Applet不能访问或执行本地系统的文件的程序。默认情况下，所有下载的Applet被认为是不可信的。

54. 通过文件系统加载的小程序在加载了互联网和Applet之间的区别是什么 ?

关于其中一个applet被加载在互联网的情况下，applet是由applet类加载器加载并须受该applet安全管理器执行的限制。
关于其中一个applet是从客户端的本地磁盘加载的情况下，applet是由文件系统加载器加载的。
通过文件系统加载的applet允许读取文件，写入文件并加载在客户端上。还有，通过文件系统加载applet允许执行，最后，通过文件系统加载的applet不管是否通过字节码验证都可加载。

55. 什么是applet类加载器，它提供了什么？

当在互联网上加载applet，该applet是由applet的类加载器加载的。类加载器强制执行Java命名空间的层次结构。此外，类加

载器保证这是在来自本地文件系统的类中唯一的命名空间，以及在每个网络源中唯一的命名空间。

当浏览器在网络上加载applet，applet的类被放置在私人的和applet起始地址有关的命名空间中。那么，那些类加载器加载的类就会通过校验。这个校验会检查类文件是否符合Java语言规范。除此之外，校验器会确保没有堆栈溢出或者向下溢出，参数的所有字节码指令也是正确的。

56. 什么是applet安全管理，它提供什么？

applet安全管理是对Java applet做了限制的机制。浏览器只能有一个安全管理器。安全管理器在启动的时候建立，之后它不能被取代，重载，重写或者延长。

Swing

57. Choice和List之间的区别是什么？

Choice是一种紧凑的方式展示，必须要拉下，是为了让用户能够看到所有的可选选项列表。Choice只能选中一个选项。List是以几个List选项是可见的方式展示的。List支持选中一个或多个List选项。

58. 什么是布局管理器？

布局管理器是用来组织容器内的组件。

59. Scrollbar和JScrollPane 的区别是什么？

Scrollbar是一个组件， 但不是一个容器，而ScrollPane是一个容器。 ScrollPane处理它自身的事件并执行它自己的滑动。

60. 哪些Swing方法是线程安全？

只有3个方法是线程安全的：repaint, revalidate, 和invalidate。

61. 说出3个支持绘图的子类。

Canvas,Frame,Panel, 和Applet类都支持绘图。

62. 什么是裁剪？

裁剪是指在有限的区域和图形类进行绘图操作。

63. MenuItem和CheckboxMenuItem的区别是什么？

CheckboxMenuItem类继承了MenuItem类，并支持选中或者取消菜单选项。

64. BorderLayout的元素是怎样组织的？

BorderLayout的元素都是在有序地分布在边缘部分(东，南，西，北)和容器的中心。

65. GridBagLayout的元素是怎样组织的？

GridBagLayout的元素根据网格组织的。元素具有不同的尺寸，并且可以占据一行或列的网格。因此，行和列可以有不同的尺寸。

66. Window和Frame有什么区别？

Frame类是继承Window类，并定义了一些主要的带菜单栏的应用程序窗口。

67. 裁剪和重画之间的关系？

当窗口被AWT绘图线程重画，它设置了裁剪区域到窗口中请求重画的区域。

68. 事件监听器接口和事件适配器类之间是什么关系？

事件监听器接口定义了一个特定事件的事件处理程序所必须实现的一些方法。事件适配器提供了一个事件侦听器接口的默认

实现。

69. 一个GUI组件怎么处理自己的事件？

GUI组件可以通过实现相应的事件监听器接口和添加它自己作为本身的事件侦听器来处理自己的事件。

70. Java布局管理器提供超过传统的窗口系统的什么样的优势？

Java使用布局管理器以一致的方式来布局组件，跨所有窗口平台。由于布局管理器不绑定绝对化的尺寸和位置，所以它们能够容纳不同窗口系统的平台具体差异。

71. Java为所有Swing组件使用的设计模式是什么？

Java为所有Swing组件使用的设计模式是是 模型视图控制器（MVC）模式。

JDBC

72. 什么是 JDBC ？

JDBC是一个抽象层，允许用户在不同数据库间进行选择。JDBC使开发人员能够在Java中编写数据库应用程序，而不必让自己关心一个特定的数据库的底层细节。

73. 解释JDBC中驱动的作用。

JDBC驱动提供了 对 JDBC API 所提供的抽象类 的 数据库供应商的特定实现。每个驱动必须提供java.sql包的以下的类实现：Connection,Statement,PreparedStatement,CallableStatement,ResultSet 和 Driver。

74.Class.forName 方法的目的是什么?

此方法用于加载驱动程序，以建立与数据库的连接。

75.与Statement相比PreparedStatement的优点?

PreparedStatement是预编译的，因此它有更好的性能。另外，PreparedStatement可以被不同输入值的查询重用。

76. CallableStatement的用途？指出用于创建CallableStatement的方法.

CallableStatement用于执行存储过程。存储过程由数据库保存并提供。存储过程可以根据用户的输入返回结果。强烈建议使用存储过程，因为它提供了安全性和模块化。准备CallableStatement的方法如下： CallableStament.prepareCall();

77. 连接池是什么？

打开和关闭数据库连接时与数据库的交互需要付出很高的代价. 特别是当数据库客户端增长时，这个代价是相当高的，并且消耗了很多资源。数据库连接池中的连接在应用服务器启动时被创建并在池中进行管理。一个连接请求由池中的数据库连接提供。当连接结束后，请求会被放回池中以供以后重用。

远程方法调用 (RMI)

78. 什么是RMI？

Java远程方法调用（RMI)是一个Java API，它执行的面向对象的等价远程过程调用（RPC）的方法，包括了直接传输序列化的Java类和分布式垃圾收集的支持。 远程方法调用（RMI），也可以看作是一个远程运行的对象上激活的方法的过程。RMI提供位置透明性，因为用户认为一个方法是在本地运行的对象上执行。 RMI Tips here.

79. 什么是RMI的体系结构的基本原理？

RMI的架构最重要的原则是将行为的定义和行为的实施分别对待。 RMI允许定义的行为和实现行为保持独立，并在独立的JVM中运行的代码。

80. RMI的体系结构层是什么？

RMI的结构主要分为以下几层：

- 桩(Stub)和框架(Skeleton)层：该层位于开发者视图的下面。该层是负责拦截客户端请求接口的方法并重定向这些请求到远程RMI服务上。
- 远程引用层：架构的第二层是处理从客户端到服务器的远程对象引用的解析。该层解析并管理从客户端到远程服务对象的引用。该连接是一对一(单播)连接的。
- 传输层：该层主要负责连接参与服务的两个JVM。它基于通过网络连接的机子的TCP/IP，提供了基本的连通性，以及一些防火墙的渗透策略。

81. 在RMI中远程接口的作用是什么？

远程接口用于识别那些不是来自本地机子接口但可以被调用的方法。所有对象都是必须直接或间接实现该接口的远程对象。实现该远程接口之前应该声明其远程接口，为每个远程对象定义构造方法，并在所有远程接口中为每个远程方法提供实现。

82. java.rmi.Naming 类扮演的角色？

java.rmi.Naming类提供了存储和获取已注册的远程对象. Naming类中的每个方法都需要一个URL格式的String作为参数的名称.

83. RMI中的绑定是什么意思？

绑定是关联或注册一个远程对象的名字的过程, 这个名字可以在以后用到, 用于查找与它绑定的远程对象. 远程对象可以通过Naming类中的bind或rebind方法与一个名字相关联.

84. Naming 类中的bind与rebind方法的区别？

bind方法的绑定主要用于将特定的名字绑定到一个远程对象, 但rebind方法的绑定用于将特定的名字重新绑定到一个新的远程对象. 如果这个名字已经绑定过了, 使用rebind这个绑定会被替换.

85. 运行RMI 程序的步骤?

为了使RMI程序正常运行需要以下步骤:

- 编译所有源文件.
- 用rmic生成stub.
- 启动rmiregistry.

- 启动RMIServer.
- 运行客户端程序.

86.RMI中stub的角色？

远程对象的stub作为远程对象在本地程序中的表示或代理. 调用者调用本地stub的一个方法, 这个方法会在远程对象上执行.当一个stub的方法被调用时, 它经历了以下步骤:

- 初始化与运行远程对象的远程JVM的连接.
- 将参数编码并传递给远程JVM.
- 等待方法调用与执行的结果.
- 解码返回值或异常(如果执行失败).
- 将返回值返回给调用者.

87. 什么是DGC？它是如何工作的？

DGC代表的是分布式垃圾收集。远程方法调用(RMI)使用的是DGC自动垃圾收集机制。由于RMI涉及到跨JVM的远程引用，垃圾回收就会相当困难。DGC使用相关的计数算法为远程对象提供自动存储管理。

88. 在RMI中使用RMISecurityManager的目的是什么？

在RMI应用程序中可以使用RMISecurityManager提供安全管理器来下载代码。如果安全管理器没有设置好，RMI的类加载器不会从远程端下载任何类。

89. 解释编组和解组。

当一个应用程序要通过网络来传送内存对象到另一台主机，或者保留它到存储器，内存表达法会将其转换到合适的格式。这个过程就叫做编组，而恢复操作就叫解组。

90. 解释序列化和反序列化。

Java提供了一个机制，是指一个对象可以被表示为字节序列，包括对象的数据，以及对象类型的信息和存储在对象中的数据类型。因此，序列化可以看做是平面化对象为了存储到磁盘中，方便后面读取和重新配置的一种方式。反序列化是一种从平面化状态到活跃状态的一种转换对象的逆过程。

Servlets

91. 什么是Servlet?

servlet是用来处理客户端请求并生成动态web内容的Java程序语言类。Servlets大多是用来处理或者存储HTML表单提交的数据，提供动态内容和管理那些不在HTTP无状态协议中的状态信息。

92. 解释一个Servlet的架构.

核心抽象概念肯定是所有servlet必须实现 javax.servlet.Servlet 接口。 每个 servlet 必须直接或者间接实现这个接口， 也可以继承于 javax.servlet.GenericServlet 或者javax.servlet.http.HTTPServlet。最后想提的是,每个 servlet 能够使用多线程服务多个请求。

93. 一个 Applet 和 一个 Servlet 区别是什么 ?

一个 Applet 是一个跑在客户机器的 网页浏览器 里面的 客户端java程序。相反，一个 servlet 是跑在网页服务器的服务的容器。一个 applet 能使用用户界面class， 而一个 servlet 不能够有一个用户界面。相反，一个servlet 等待客户端的 HTTP 请求并为每一个请求生成一个响应。

94. GenericServlet 和 HttpServlet 的区别是什么 ?

GenericServlet 是一个实现了Servlet 和 ServletConfig 接口的通用的协议无关的 servlet . 那些继承于GenericServlet 类的 servlet 将重写 service 方法 。最后想提的是， 为了给Web用户开发一个使用HTTP协议服务的HTTP servlet， 你的 servlet 必须改为继承于 HttpServlet 。查看Servlet的示例 。

95.解释一个Servlet的生命周期.

对每一个客户端的请求，这个Servlet引擎加载servlet和调用它的init方法，以便在servlet初始化。然后， Servlet对象处理所有从客户端来的后续请求，通过为每个请求单独调用服务的方法。最后，该servlet调用服务器的destroy方法。

96 .doGet()和doPost()之间的区别是什么？

doGet ： GET方法附加请求的URL的名称 - 值对。因此，存在客户端的请求字符数量的限制。此外，该请求的参数值为可见，因此，如果有敏感信息不能采用这种方式。
doPost： POST方法克服了GET请求的限制，将发送请求的值置于body里。此外，发送值的数量没有限制。最后，通过POST请求传递的敏感信息是不可见的

97. web应用是什么 ?

web应用是web服务的延伸. 主要有两种类型：面向视觉的和面向服务的. 面向视觉型的应用通过编辑语言来动态展示交互页面. 面向应用的则提供了后端的service.总的来说，就是一堆置于 server's URL明明空间下的servlets.

98. 什么是服务端包含 (SSI) ?

服务端包含是服务端的一种简单脚本语言,主要应用在Web方面，置于servlet标签中. 最常用的地方就是在网页中引入一个或多个文件. 当浏览器加载页面时，使用servlet产生的超文本替换其标签.

99. 什么是 Servlet 链?

Servlet 链是指将上一个servlet的结果传到下一个.第二个的结构又可以传到第三个. 最后的servlet负责将响应回复给客户端.

100. 如何知道请求 servlet的客户端信息？

ServletRequest类可以获取客户端的IP地址或主机名. getRemoteAddr()获取IP getRemoteHost()获取主机名. 示例如下 here.

101. Http response的结构是什么？

HTTP response 包括了三个部分:

- Status Code: 描述了这次回应的状况. 它可以用来检查这次请求是否成功完成. 一旦请求失败了, 这个status code可以用来寻找原因. 如果你的 servlet 没有返回一个status code, 默认就会返回成功的status code, HttpServletResponse.SC_OK.
- HTTP Headers: 它包含了response的更多信息.举个例子，headers可以反应response的访问date/time, 或者是用于将实体安全地传送到用户的编码形式。可以阅读 how to retrieve headers in Servlet here.
- Body: 它是response的具体内容. 可能包括HTML内容,比如图片。Body包括了紧接Header发送的HTTP事务消息数据字节。

102. 什么是cookie？session和cookie之间的区别是什么？

Cookie是Web服务器发送到浏览器的一小块信息，浏览器为每个Web服务器在本地文件中存储cookie。 在以后的请求里, 浏览器对特定的Web服务器，将request和所有特定的Web服务器的cookie一起发送。Session和Cookie之间的区别如下：

- Session无论在客户端浏览器的设置都可以工作。客户端可以选择禁用cookies。然而，Session仍然可以工作，因为客户端没有能力在服务器端禁用Session。
- Session和cookie也t有不同的信息存储量。 HTTP会话能够存储任何Java对象，而一个cookie只能保存String对象。

103.浏览器和servlet通过什么协议通信？

HTTP协议.

104. 什么是HTTP通道？

通道是指使用 HTTP或 HTTPS 封装其它的网络协议. HTTP包装了其他的网络通信协议. 其它协议通过HTTP来发送请求的过程就使用了通道.

105. sendRedirect和 forward 方法的区别?

sendRedirect创建一个新的请求, forward只是将请求转发.之前请求中包含的对象在redirect后将不可用, 因为产生了一个新的请求. 但是 forwarding不受此限制.通常来说,sendRedirect 比 forward 方法慢一些.

106. 什么是URL编码和解码？

URL编码就是替换其中的空格和特殊字符, 变成相应的Hex码.解码就是反向操作。

JSP

107. JSP页面是什么？

一个JSP页面是一个文本文档，包含了两种类型的文本：静态数据和JSP元素。静态数据可以以任何一种基于文本的格式表达，比如HTML或者XML。JSP是一种混合了静态内容与动态生成内容的技术。查看JSP例子

108. JSP请求是如何被处理的？

一个JSP请求的到来,浏览器首先以一个jsp扩展的文件名来请求一个页面。然后，Web Server读取到该请求，使用JSP编译器将JSP页面转换为一个servlet类。注意JSP文件只在该页面的第一次请求或者JSP文件已经改变时才编译。生成的servlet类被调用，去处理浏览器的请求。一旦请求处理完，servlet会向客户端返回一个相应。查看如何获取JSP请求中的参数

109. JSP的优势所在？

使用JSP技术有以下优势：

- JSP网页被动态的编译，因此开发者很容易更新当前代码。
- JSP网页可预编译。
- JSP网页能容易的组合成静态模版，包括HTML XML片段，且代码可形成动态内容。
- 开发者能提供自定义的JSP标签库，且使用类XML语法访问。
- 开发者可以在组件级别做逻辑上的改变，但是不能使用程序逻辑编辑个别的网页。

110. 什么是指令？在JSP中， 包括哪些不同类型的指令？

指令是JSP引擎所处理的命令，当网页被编译成小程序时，指令用于去设置页面级别的命令，从外部文件插入数据，指定自定义的标签库。指令被包括在<%@ 和%>内。指令的类型分为：

- 包含指令：用于包含文件，和当用页面合并文件内容。
- 页面指令：用于定义JSP页面的具体属性，比如错误页面和缓冲池。
- 标签库：用于声明页面中使用到的自定义标签库。

111. 什么是JSP的actions？

JSP的actions是使用XML语法结构来控制Servlet引擎的行为。JSP的actions是在当JSP页面请求的时候才执行。它们会动态插入一个文件中，再利用JavaBeans的组件，转发给用户到另一个页面，或者生成带Java插件的HTML页面。以下是一些可以操作的actions：

- jsp:include- 当JSP页面被请求的时候，包含了一个文件。
- jsp:useBean- 寻找或者实例化一个JavaBean。
- jsp:setProperty- 设置JavaBean的属性。
- jsp:getProperty- 得到JavaBean的属性。
- jsp:forward- 转发请求到新的页面。
- jsp:plugin- 生成特定浏览器的代码。

112. 什么是Scriptlet？

在Java Server Page(JSP)技术中，scriptlet是嵌入在JSP页面的其中一块Java代码。scriptlet是标签内的任何东西。在这些标签之间，用户可以添加任何有效的scriptlet。

113. 什么是声明？

声明类似于Java中的变量声明。声明是为表达式或者scriptlets后续的使用而声明的变量。添加一个声明，你必须在你的声明中使用序列。

114. 什么是表达式？

JSP表达式是把一个脚本语言表达式的值插入进来，转换成字符串，进入数据流之后再通过web服务器返回给客户端。表达式的定义是在<% = 和 %>标签之间。

115. 什么是隐式对象？他们是什么？

JSP隐式对象是指JSP容器在每页中提供给开发人员的Java对象。开发人员可以直接调用他们，而不需要任何显示声明。JSP隐式对象也被称为预定义变量。以下对象都是在JSP页面中隐式存在的：

- application
- page
- request
- response
- session
- exception
- out
- config
- pageContext

坚持看到这里了？哇!这是一篇很长的文章，详细地描述了在Java面试中各种类型的问题。