Java 程序员面试笔试真题库

呆萌钟　整理

目　　录

真题详解篇

面试笔试经验技巧篇

想找到一份程序员的工作，一点技术都没有显然是不行的，但是，只有技术也是不够的。面试笔试经验技巧篇主要针对程序员面试笔试中遇到的 13 个常见问题进行深度解析，并且结合实际情景，给出了一个较为合理的参考答案以供读者学习与应用，掌握这 13 个问题的解答精髓，对于求职者大有裨益。

经验技巧 1　如何巧妙地回答面试官的问题？

所谓“来者不善，善者不来”，程序员面试中，求职者不可避免地需要回答面试官各种刁钻、犀利的问题，回答面试官的问题千万不能简单地回答“是”或者“不是”，而应该具体分析“是”或者“不是”的理由。

回答面试官的问题是一门很深入的学问。那么，面对面试官提出的各类问题，如何才能条理清晰地回答呢？如何才能让自己的回答不至于撞上枪口呢？如何才能让自己的回答结果令面试官满意呢？

谈话是一种艺术，回答问题也是一种艺术，同样的话，不同的回答方式，往往也会产生出不同的效果，甚至是截然不同的效果。在此，编者提出以下几点建议，供读者参考。首先回答问题务必谦虚谨慎。既不能让面试官觉得自己很自卑，唯唯诺诺，也不能让面试官觉得自己清高自负，而应该通过问题的回答表现出自己自信从容、不卑不亢的一面。例如，当面试官提出“你在项目中起到了什么作用”的问题时，如果求职者回答：我完成了团队中最难的工作，此时就会给面试官一种居功自傲的感觉，而如果回答：我完成了文件系统的构建工作，这个工作被认为是整个项目中最具有挑战性的一部分内容，因为它几乎无法重用以前的框架，需要重新设计。这种回答不仅不傲慢，反而有理有据，更能打动面试官。

其次，回答面试官的问题时，不要什么都说，要适当地留有悬念。人一般都有猎奇的心理，面试官自然也不例外，而且，人们往往对好奇的事情更有兴趣、更加偏爱，也更加记忆深刻。所以，在回答面试官问题时，切记说关键点而非细节，说重点而非和盘托出，通过关键点，吸引面试官的注意力，等待他们继续“刨根问底”。例如，当面试官对你的简历中一个算法问题有兴趣，希望了解时，可以如下回答：我设计的这种查找算法，对于 80%以上的情况，都可以将时间复杂度从 O(n)降低到 O(log n)，如果您有兴趣，我可以详细给您分析具体的细节。

最后，回答问题要条理清晰、简单明了，最好使用“三段式”方式。所谓“三段式”，有点类似于中学作文中的写作风格，包括“场景/任务”“行动”和“结果”三部分内容。以面试官提的问题“你在团队建设中，遇到的最大挑战是什么”为例，第一步，分析场景/任务：在我参与的一个 ERP 项目中，我们团队一共四个人，除了我以外的其他三个人中，两个人能力很给力，人也比较好相处，但有一个人却不太好相处，每次我们小组讨论问题的时候，他都不太爱说话，也很少发言，分配给他的任务也很难完成。第二步，分析行动：为了提高团队的综合实力，我决定找个时间和他好好单独谈一谈。于是我利用周末时间，约他一起吃饭，吃饭的时候，顺便讨论了一下我们的项目，我询问了一些项目中他遇到的问题，通过他的回答，我发现他并不懒，也不糊涂，只是对项目不太了解，缺乏经验，缺乏自信而已，所以越来越孤立，越来越不愿意讨论问题。为了解决这个问题，我尝试着把问题细化到他可以完成的程度，从而建立起他的自信心。第三步，分析结果：他是小组中水平最弱的人，但是，慢慢地，他的技术变得越来越厉害了，也能够按时完成安排给他的工作了，人也越来越自信了，也越来越喜欢参与我们的讨论，并发表自己的看法，我们也都愿意与他一起合作了。“三段式”回答的一个最明显的好处就是条理清晰，既有描述，也有结果，有根有据，让面试官一目了然。

回答问题的技巧，是一门大的学问。求职者完全可以在平时的生活中加以练习，提高自己与人沟通的技能，等到面试时，自然就得心应手了。

经验技巧 2　如何回答技术性的问题？

程序员面试中，面试官会经常询问一些技术性的问题，有的问题可能比较简单，都是历年的笔试面试真题，求职者在平时的复习中会经常遇到，应对自然不在话下。但有的题目可能比较难，来源于 Google、Microsoft 等大企业的题库或是企业自己为了招聘需要设计的题库，求职者可能从来没见过或者从来都不能完整地、独立地想到解决方案，而这些题目往往又是企业比较关注的。

如何能够回答好这些技术性的问题呢？编者建议：会做的一定要拿满分，不会做的一定要拿部分分。即对于简单的题目，求职者要努力做到完全正确，毕竟这些题目，只要复习得当，完全回答正确一点问题都没有（编者认识的一个朋友据说把《编程之美》、《编程珠玑》、《程序员面试笔试宝典》上面的技术性题目与答案全都背得滚瓜烂熟了，后来找工作简直成了“offer 杀器”，完全就是一个 Bug，无解了）；对于难度比较大的题目，不要惊慌，也不要害怕，即使无法完全做出来，也要努力思考问题，哪怕是半成品也要写出来，至少要把自己的思路表达给面试官，让面试官知道你的想法，而不是完全回答不会或者放弃，因为面试官很多时候除了关注你的独立思考问题的能力以外，还会关注你技术能力的可塑性，观察求职者是否能够在别人的引导下去正确地解决问题，所以，对于你不会的问题，他们很有可能会循序渐进地启发你去思考，通过这个过程，让他们更加了解你。

一般而言，在回答技术性问题时，求职者大可不必胆战心惊，除非是没学过的新知识，否则，一般都可以采用以下六个步骤来分析解决。

（1）勇于提问

面试官提出的问题，有时候可能过于抽象，让求职者不知所措，或者无从下手，所以，对于面试中的疑惑，求职者要勇敢地提出来，多向面试官提问，把不明确或二义性的情况都问清楚。不用担心你的问题会让面试官烦恼，影响你的面试成绩，相反还对面试结果产生积极影响：一方面，提问可以让面试官知道你在思考，也可以给面试官一个心思缜密的好印象；另一方面，方便后续自己对问题的解答。

例如，面试官提出一个问题：设计一个高效的排序算法。求职者可能丈二和尚摸不到头脑，排序对象是链表还是数组？数据类型是整型、浮点型、字符型还是结构体类型？数据基本有序还是杂乱无序？数据量有多大，1000 以内还是百万以上个数？此时，求职者大可以将自己的疑问提出来，问题清楚了，解决方案也自然就出来了。

（2）高效设计

对于技术性问题，如何才能打动面试官？完成基本功能是必须的，仅此而已吗？显然不是，完成基本功能顶多只能算及格水平，要想达到优秀水平，至少还应该考虑更多的内容，以排序算法为例：时间是否高效？空间是否高效？数据量不大时也许没有问题，如果是海量数据呢？是否考虑了相关环节，例如数据的“增删改查”？是否考虑了代码的可扩展性、安全性、完整性以及鲁棒性？如果是网站设计，是否考虑了大规模数据访问的情况？是否需要考虑分布式系统架构？是否考虑了开源框架的使用？

（3）伪代码先行

有时候实际代码会比较复杂，上手就写很有可能会漏洞百出、条理混乱，所以，求职者可以首先征求面试官的同意，在编写实际代码前，写一个伪代码或者画好流程图，这样做往往会让思路更加清晰明了。

切记在写伪代码前要告诉面试官，他们很有可能对你产生误解，认为你只会纸上谈兵，实际编码能力却不行。只有征得了他们的允许，方可先写伪代码。

（4）控制节奏

如果是算法设计题，面试官都会给求职者一个时间限制用以完成设计，一般为 20min 左右。完成得太慢，会给面试官留下能力不行的印象，但完成得太快，如果不能保证百分百正确，也会给面试官留下毛手毛脚的印象，速度快当然是好事情，但只有速度，没有质量，速度快根本不会给面试加分。所以，编者建议，回答问题的节奏最好不要太慢，也不要太快，如果实在是完成得比较快，也不要急于提交给面试官，最好能够利用剩余的时间，认真仔细地检查一些边界情况、异常情况及极性情况等，看是否也能满足要求。

（5）规范编码

回答技术性问题时，多数都是纸上写代码，离开了编译器的帮助，求职者要想让面试官对自己的代码一看即懂，除了字迹要工整，不能眉飞色舞以外，最好是能够严格遵循编码规范：函数变量命名、换行缩进、语句嵌套和代码布局等，同时，代码设计应该具有完整性，保证代码能够完成基本功能、输入边界值能够得到正确地输出、对各种不合规范的非法输入能够做出合理的错误处理，否则，写出的代码即使无比高效，面试官也不一定看得懂或者看起来非常费劲，这些对面试成功都是非常不利的。

（6）精心测试

在软件界，有一句真理：任何软件都有 bug。但不能因为如此就纵容自己的代码，允许错误百出。尤其是在面试过程中，实现功能也许并不十分困难，困难的是在有限的时间内设计出的算法，各种异常是否都得到了有效的处理，各种边界值是否都在算法设计的范围内。

测试代码是让代码变得完备的高效方式之一，也是一名优秀程序员必备的素质之一。所以，在编写代码前，求职者最好能够了解一些基本的测试知识，做一些基本的单元测试、功能测试、边界测试以及异常测试。

在回答技术性问题时，注意在思考问题的时候，千万别一句话都不说，面试官面试的时间是有限的，他们希望在有限的时间内尽可能地去了解求职者，如果求职者坐在那里一句话不说，不仅会让面试官觉得求职者技术水平不行，思考问题能力以及沟通能力可能都存在问题。

其实，在面试时，求职者往往会存在一种思想误区，把技术性面试的结果看得太重要了。面试过程中的技术性问题，结果固然重要，但也并非最重要的内容，因为面试官看重的不仅仅是最终的结果，还包括求职者在解决问题的过程中体现出来的逻辑思维能力以及分析问题的能力。所以，求职者在与面试官的博弈中，要适当地提问，通过提问获取面试官的反馈信息，并抓住这些有用的信息进行辅助思考，从而博得面试官的欢心，进而提高面试的成功率。

经验技巧 3　如何回答非技术性问题？

评价一个人的能力，除了专业能力，还有一些非专业能力，如智力、沟通能力和反应能力等，所以在 IT 企业招聘过程的笔试面试环节中，并非所有的笔试内容都是 C/C++、数据结构与算法及操作系统等专业知识，也包括其他一些非技术类的知识，如智力题、推理题和作文题等。技术水平测试可以考查一个求职者的专业素养，而非技术类测试则更加强调求职者的综合素质，包括数学分析能力、反应能力、临场应变能力、思维灵活性、文字表达能力和性格特征等内容。考查的形式多种多样，但与公务员考查相似，主要包括行测（占大多数）、性格测试（大部分都有）、应用文和开放问题等内容。

每个人都有自己的答题技巧，答题方式也各不相同，以下是一些相对比较好的答题技巧（以行测为例）：

1）合理有效的时间管理。由于题目的难易不同，所以不要对所有题目都“绝对的公平”、都“一刀切”，要有轻重缓急，最好的做法是不按顺序回答。行测中有各种题型，

如数量关系、图形推理、应用题、资料分析和文字逻辑等，而不同的人擅长的题型是不一样的，因此应该首先回答自己最擅长的问题。例如，如果对数字比较敏感，那么就先答数量关系。

2）注意时间的把握。由于题量一般都比较大，可以先按照总时间/题数来计算每道题的平均答题时间，如 10s，如果看到某一道题 5s 后还没思路，则马上放弃。在做行测题目的时候，以在最短的时间内拿到最多分为目标。

3)平时多关注图表类题目,培养迅速抓住图表中各个数字要素间相互逻辑关系的能力。

4）做题要集中精力，只有集中精力、全神贯注，才能将自己的水平最大限度地发挥出来。

5）学会关键字查找，通过关键字查找，能够提高做题效率。

6）提高估算能力，有很多时候，估算能够极大地提高做题速度，同时保证正确率。

除了行测以外，一些企业非常相信个人性格对入职匹配的影响，所以都会引入相关的性格测试题用于测试求职者的性格特性，看其是否适合所投递的职位。大多数情况下，只要按照自己的真实想法选择就行了，不要弄巧成拙，因为测试是为了得出正确的结果，所以大多测试题前后都有相互验证的题目。如果求职者自作聪明，选择该职位可能要求的性格选项，则很可能导致测试前后不符，这样很容易让企业发现你是个不诚实的人，从而首先予以筛除。

经验技巧 4　如何回答快速估算类问题？

有些大企业的面试官，总喜欢使一些“阴招”“损招”，出一些快速估算类问题，对他们而言，这些问题只是手段，不是目的，能够得到一个满意的结果固然是他们所需要的，但更重要的是通过这些题目他们可以考查求职者的快速反应能力以及逻辑思维能力。由于求职者平时准备的时候可能对此类问题有所遗漏，一时很难想起解决的方案。而且，这些题目乍一看确实是毫无头绪，无从下手，完全就是坑求职者的，其实求职者只要从惊慌失措中冷静下来，稍加分析，也就那么回事。因为此类题目比较灵活，属于开放性试题，一般没有标准答案，只要弄清楚了回答要点，分析合理到位，具有说服力，能够自圆其说，就是正确答案，一点都不困难。

例如，面试官可能会问这样一个问题：“请你估算一下一家商场在促销时一天的营业额？”，求职者又不是统计局官员，如何能够得出一个准确的数据呢？求职者家又不是开商场的，如何能够得出一个准确的数据呢？即使求职者是商场的大当家，也不可能弄得清清楚楚明明白白吧？

难道此题就无解了吗？其实不然，本题只要能够分析出一个概数就行了，不一定要精确数据，而分析概数的前提就是做出各种假设。以该问题为例，可以尝试从以下思路入手：从商场规模、商铺规模入手，通过每平方米的租金，估算出商场的日租金，再根据商铺的成本构成,得到全商场日均交易额,再考虑促销时的销售额与平时销售额的倍数关系,乘以倍数，即可得到促销时一天的营业额。具体而言，包括以下估计数值：

1）以一家较大规模商场为例，商场一般按 6 层计算，每层大约长 100m，宽 100m，合计 60000m^2 的面积。

2）商铺规模占商场规模的一半左右，合计 30000m^2。

3）商铺租金约为 40 元/ m^2，估算出年租金为 40×30000×365=4.38 亿。

4）对商户而言，租金一般占销售额 20%左右，则年销售额为 4.38 亿×5=21.9 亿。计算平均日销售额为 21.9 亿/365=600 万。

5）促销时的日销售额一般是平时的 10 倍，所以大约为 600 万*10=6000 万。

此类题目涉及面比较广，例如：估算一下北京小吃店的数量？估算一下中国在过去一年

方便面的市场销售额是多少？估算一下长江的水的质量？估算一下一个行进在小雨中的人5min 内身上淋到的雨的质量？估算一下东方明珠电视塔的质量？估算一下中国去年一年一共用掉了多少块尿布？估算一下杭州的轮胎数量？但一般都是即兴发挥，不是哪道题记住答案就可以应付得了的。遇到此类问题，一步步抽丝剥茧，才是解决之道。

经验技巧 5　如何回答算法设计问题？

程序员面试中的很多算法设计问题，都是历年来各家企业的“炒现饭”，不管求职者以前对算法知识学习得是否扎实，理解得是否深入，只要面试前买本《程序员面试笔试宝典》（编者早前编写的一本书，由机械工业出版社出版），学习上一段时间，牢记于心，应付此类题目完全没有问题，但遗憾的是，很多世界级知名企业也深知这一点，如果纯粹是出一些毫无技术含量的题目，对于考前“突击手”而言，可能会占尽便宜，但对于那些技术好的人而言是非常不公平的。所以，为了把优秀的求职者与一般的求职者能够更好地区分开来，他们会年年推陈出新，越来越倾向于出一些有技术含量的“新”题，这些题目以及答案，不再是以前的陈谷子烂芝麻了，而是经过精心设计的好题。

在程序员面试中，算法的地位就如同是 GRE 或托福考试在出国留学中的地位一样，必须但不是最重要的，它只是众多考核方面中的一个而已，不一定就能决定求职者的生死。虽然如此，但并非说就不用去准备算法知识了，因为算法知识回答得好，必然会成为面试的加分项，对于求职成功，百利而无一害。那么如何应对此类题目呢？很显然，编者不可能将此类题目都在《程序员面试笔试宝典》中一一解答，一来由于内容众多，篇幅有限，二来也没必要，今年考过了，以后一般就不会再考了，不然还是没有区分度。编者以为，靠死记硬背肯定是行不通的，解答此类算法设计问题，需要求职者具有扎实的基本功以及良好的运用能力，编者无法左右求职者的个人基本功以及运用能力，因为这些能力需要求职者“十年磨一剑”地苦学，但编者可以提供一些比较好的答题方法和解题思路，以供求职者在面试时应对此类算法设计问题。“授之以鱼不如授之以渔”，岂不是更好？

（1）归纳法

此方法通过写出问题的一些特定的例子，分析总结其中一般的规律。具体而言就是通过列举少量的特殊情况，经过分析，最后找出一般的关系。例如，某人有一对兔子饲养在围墙中，如果它们每个月生一对兔子，且新生的兔子在第二个月后也是每个月生一对兔子，问一年后围墙中共有多少对兔子。

使用归纳法解答此题，首先想到的就是第一个月有多少对兔子，第一个月的时候，最初的一对兔子生下一对兔子，此时围墙内共有两对兔子。第二个月仍是最初的一对兔子生下一对兔子，共有 3 对兔子。到第三个月除最初的兔子新生一对兔子外，第一个月生的兔子也开始生兔子，因此共有 5 对兔子。通过举例，可以看出，从第二个月开始，每一个月兔子总数都是前两个月兔子总数之和，$U_{n+1}=U_n+U_{n-1}$，一年后，围墙中的兔子总数为 377 对。

此种方法比较抽象，也不可能对所有的情况进行列举，所以，得出的结论只是一种猜测，还需要进行证明。

（2）相似法

正如编者“年年岁岁花相似，岁岁年年仍单身”一样，此方法考虑解决问题的算法是相似的。如果面试官提出的问题与求职者以前用某个算法解决过的问题相似，此时此刻就可以触类旁通，尝试改进原有算法来解决这个新问题。而通常情况下，此种方法都会比较奏效。

例如，实现字符串的逆序打印，也许求职者从来就没遇到过此问题，但将字符串逆序肯定在求职准备的过程中是见过的。将字符串逆序的算法稍加处理，即可实现字符串的逆序打印。

（3）简化法

此方法首先将问题简单化，例如改变一下数据类型、空间大小等，然后尝试着将简化后的问题解决，一旦有了一个算法或者思路可以解决这个被“阉割过”的问题，再将问题还原，尝试着用此类方法解决原有问题。

例如，在海量日志数据中提取出某日访问 xxx 网站次数最多的那个 IP。很显然，由于数据量巨大，直接进行排序不可行，但如果数据规模不大时，采用直接排序不失为一种好的解决方法。那么如何将问题规模缩小呢？于是想到了Hash法，Hash往往可以缩小问题规模，然后在“阉割过”的数据里面使用常规排序算法即可找出此问题的答案。

（4）递归法

为了降低问题的复杂度，很多时候都会将问题逐层分解，最后归结为一些最简单的问题，这就是递归。此种方法，首先要能够解决最基本的情况，然后以此为基础，解决接下来的问题。

例如，在寻求全排列的时候，可能会感觉无从下手，但仔细推敲，会发现后一种排列组合往往是在前一种排列组合的基础上进行的重新排列，只要知道了前一种排列组合的各类组合情况，只需将最后一个元素插入到前面各种组合的排列里面，就实现了目标：即先截去字符串 s[1…n]中的最后一个字母，生成所有 s[1…n–1]的全排列，然后再将最后一个字母插入到每一个可插入的位置。

（5）分治法

任何一个可以用计算机求解的问题所需的计算时间都与其规模有关。问题的规模越小，越容易直接求解，解题所需的计算时间也越少。而分治法正是充分考虑到这一内容，将一个难以直接解决的大问题，分割成一些规模较小的相同问题，以便各个击破，分而治之。分治法一般包含以下三个步骤：

1）将问题的实例划分为几个较小的实例，最好具有相等的规模。

2）对这些较小的实例求解，而最常见的方法一般是递归。

3）如果有必要，合并这些较小问题的解，以得到原始问题的解。

分治法是程序员面试常考的算法之一，一般适用于二分查找、大整数相乘、求最大子数组和、找出伪币、金块问题、矩阵乘法、残缺棋盘、归并排序、快速排序、距离最近的点对、导线与开关等。

（6）Hash 法

很多面试笔试题目，都要求求职者给出的算法尽可能高效。什么样的算法是高效的？一般而言，时间复杂度越低的算法越高效。而要想达到时间复杂度的高效，很多时候就必须在空间上有所牺牲，用空间来换时间。而用空间换时间最有效的方式就是 Hash 法、大数组和位图法。当然，此类方法并非包治百病，有时，面试官也会对空间大小进行限制，那么此时，求职者只能再去思考其他的方法了。

其实，凡是涉及大规模数据处理的算法设计中，Hash 法就是最好的方法之一。

（7）轮询法

在设计每道面试笔试题时，往往会有一个载体，这个载体便是数据结构，例如数组、链表、二叉树或图等，当载体确定后，可用的算法自然而然地就会暴露出来。可问题是很多时候并不确定这个载体是什么。当无法确定这个载体时，一般也就很难想到合适的方法了。

编者建议，此时，求职者可以采用最原始的思考问题的方法——轮询法，在脑海中轮询各种可能的数据结构与算法，常考的数据结构与算法一共就那么几种（见表 1），即使不完全一样，也是由此衍生出来的或者相似的，总有一款适合考题的。

表 1　最常考的数据结构与算法知识点

数 据 结 构	算　　法	概　　念
链表	广度（深度）优先搜索	位操作

数组	递归	设计模式
二叉树	二分查找	内存管理（堆、栈等）
树	排序（归并排序、快速排序等）	
堆（大顶堆、小顶堆）	树的插入/删除/查找/遍历等	
栈	图论	
队列	Hash 法	
向量	分治法	
Hash 表	动态规划	

此种方法看似笨拙，其实实用，只要求职者对常见的数据结构与算法烂熟于心，一点都没有问题。

为了更好地理解这些方法，求职者可以在平时的准备过程中，应用此类方法去答题，做得多了，自然对各种方法也就熟能生巧了，面试的时候，再遇到此类问题，也就能够收放自如了。当然，千万不要相信有着张无忌般的运气，能够在一夜之间练成乾坤大挪移这一绝世神功，称霸武林，算法设计功力的练就是平时一点一滴的付出和思维的磨练。方法与技巧也许只是给面试打了一针“鸡血”、喂一口“大补丸”，不会让自己变得从容自信，真正的功力还是需要一个长期的积累过程的。

经验技巧 6　如何回答系统设计题？

应届生在面试的时候，偶尔也会遇到一些系统设计题，而这些题目往往只是测试一下求职者的知识面，或者测试求职者对系统架构方面的了解，一般不会涉及具体的编码工作。虽然如此，对于此类问题，很多人还是感觉难以应对，也不知道从何说起。

如何应对此类题目呢？在正式介绍基础知识之前，首先罗列几个常见的系统设计相关的面试笔试题，如下所示：

1）设计一个 DNS 的 Cache 结构，要求能够满足每秒 5000 次以上的查询，满足 IP 数据的快速插入，查询的速度要快（题目还给出了一系列的数据，比如站点数总共为 5000 万、IP 地址有 1000 万等）。

2）有 N 台机器，M 个文件，文件可以以任意方式存放到任意机器上，文件可任意分割成若干块。假设这 N 台机器的宕机率小于 1/3，想在宕机时可以从其他未宕机的机器中完整导出这 M 个文件，求最好的存放与分割策略。

3）假设有三十台服务器，每台服务器上面都存有上百亿条数据（有可能重复），如何找出这三十台机器中，根据某关键字，重复出现次数最多的前 100 条？要求使用 Hadoop 来实现。

4）设计一个系统，要求写速度尽可能快，并说明设计原理。

5）设计一个高并发系统，说明架构和关键技术要点。

6）有 25T 的 log(query->queryinfo)，log 在不断地增长，设计一个方案，给出一个 query 能快速返回 queryinfo。

以上所有问题中凡是不涉及高并发的，基本可以采用 Google 的三个技术解决，即 GFS、MapReduce 和 Bigtable，这三个技术被称为“Google 三驾马车”，Google 只公开了论文而未开源代码，开源界对此非常有兴趣，仿照这三篇论文实现了一系列软件，如 Hadoop、HBase、HDFS 及 Cassandra 等。

在 Google 这些技术还未出现之前，企业界在设计大规模分布式系统时，采用的架构往往是 database+sharding+cache，现在很多公司（比如 taobao、weibo.com）仍采用这种架构。在这种架构中，仍有很多问题值得去探讨。如采用什么数据库，是 SQL 界的 MySQL 还

是 NoSQL 界的 Redis/TFS，两者有何优劣？采用什么方式 sharding（数据分片），是水平分片还是垂直分片？据网上资料显示，weibo.com 和 taobao 图片存储中曾采用的架构是 Redis/MySQL/TFS+sharding+cache，该架构解释如下：前端 cache 是为了提高响应速度，后端数据库则用于数据永久存储，防止数据丢失，而 sharding 是为了在多台机器间分摊负载。最前端由大块大块的 cache 组成，要保证至少 99%（该数据在 weibo.com 架构中的是自己猜的，而 taobao 图片存储模块是真实的）的访问数据落在 cache 中，这样可以保证用户访问速度，减少后端数据库的压力。此外，为了保证前端 cache 中的数据与后端数据库中的数据一致，需要有一个中间件异步更新（为什么使用异步？理由简单：同步代价太高。异步有缺点，如何弥补?）数据，这个有些人可能比较清楚，新浪有个开源软件叫 Memcachedb（整合了 Berkeley DB 和 Memcached），正是完成此功能。另外，为了分摊负载压力和海量数据，会将用户微博信息经过分片后存放到不同节点上（称为“Sharding”）。

这种架构优点非常明显：简单，在数据量和用户量较小的时候完全可以胜任。但缺点是扩展性和容错性太差，维护成本非常高，尤其是数据量和用户量暴增之后，系统不能通过简单地增加机器解决该问题。

鉴于此，新的架构应运而生。新的架构仍然采用 Google 公司的架构模式与设计思想，以下将分别就此内容进行分析。

GFS 是一个可扩展的分布式文件系统，用于大型的、分布式的、对大量数据进行访问的应用。它运行于廉价的普通硬件上，提供容错功能。现在开源界有 HDFS（Hadoop Distributed File System），该文件系统虽然弥补了数据库+sharding 的很多缺点，但自身仍存在一些问题，比如：由于采用 master/slave 架构，因此存在单点故障问题；元数据信息全部存放在 master 端的内存中，因而不适合存储小文件，或者说如果存储大量小文件，那么存储的总数据量不会太大。

MapReduce 是针对分布式并行计算的一套编程模型。其最大的优点是：编程接口简单，自动备份（数据默认情况下会自动备三份），自动容错和隐藏跨机器间的通信。在 Hadoop 中，MapReduce 作为分布计算框架，而 HDFS 作为底层的分布式存储系统，但 MapReduce 不是与 HDFS 耦合在一起的，完全可以使用自己的分布式文件系统替换掉 HDFS。当前 MapReduce 有很多开源实现，如 Java 实现 Hadoop MapReduce，C++实现 Sector/sphere 等，甚至有些数据库厂商将 MapReduce 集成到数据库中了。

BigTable 俗称“大表”，是用来存储结构化数据的，编者觉得，BigTable 在开源界最火爆，其开源实现最多，包括 HBase、Cassandra 和 levelDB 等，使用也非常广泛。

除了 Google 的这“三驾马车”以外，还有其他一些技术可供学习与使用：

Dynamo：亚马逊的 key-value 模式的存储平台，可用性和扩展性都很好，采用 DHT（Distributed Hash Table）对数据分片，解决单点故障问题，在 Cassandra 中，也借鉴了该技术，在 BT 和电驴这两种下载引擎中，也采用了类似算法。

虚拟节点技术：该技术常用于分布式数据分片中。具体应用场景是：有一大块数据（可能 TB 级或者 PB 级），需按照某个字段（key）分片存储到几十（或者更多）台机器上，同时想尽量负载均衡且容易扩展。传统的做法是：Hash(key) mod N，这种方法最大的缺点是不容易扩展，即增加或者减少机器均会导致数据全部重分布，代价太大。于是新技术诞生了，其中一种是上面提到的 DHT，现在已经被很多大型系统采用，还有一种是对“Hash(key) mod N”的改进：假设要将数据分布到 20 台机器上，传统做法是 Hash(key) mod 20，而改进后，N 取值要远大于 20，比如是 20000000，然后采用额外一张表记录每个节点存储的 key 的模值，比如：

node1：0~1000000

node2：1000001~2000000

……

这样，当添加一个新的节点时，只需将每个节点上部分数据移动给新节点，同时修改一下该表即可。

Thrift：Thrift 是一个跨语言的 RPC 框架，分别解释“RPC”和“跨语言”如下：RPC 是远程过程调用，其使用方式与调用一个普通函数一样，但执行体发生在远程机器上；跨语言是指不同语言之间进行通信，比如 C/S 架构中，Server 端采用 C++编写，Client 端采用 PHP 编写，怎样让两者之间通信，Thrift 是一种很好的方式。

本篇最前面的几道题均可以映射到以上几个系统的某个模块中，如：

1）关于高并发系统设计，主要有以下几个关键技术点：缓存、索引、数据分片及锁粒度尽可能小。

2）题目 2 涉及现在通用的分布式文件系统的副本存放策略。一般是将大文件切分成小的 block（如 64MB）后，以 block 为单位存放三份到不同的节点上，这三份数据的位置需根据网络拓扑结构配置，一般而言，如果不考虑跨数据中心，可以这样存放：两个副本存放在同一个机架的不同节点上，而另外一个副本存放在另一个机架上，这样从效率和可靠性上，都是最优的（这个 Google 公布的文档中有专门的证明，有兴趣的可参阅一下）。如果考虑跨数据中心，可将两份存在一个数据中心的不同机架上，另一份放到另一个数据中心。

3）题目 4 涉及 BigTable 的模型。主要思想是将随机写转化为顺序写，进而大大提高写速度。具体是：由于磁盘物理结构的独特设计，其并发的随机写（主要是因为磁盘寻道时间长）非常慢，考虑到这一点，在 BigTable 模型中，首先会将并发写的大批数据放到一个内存表（称为“memtable”）中，当该表大到一定程度后，会顺序写到一个磁盘表（称为“SSTable”）中，这种写是顺序写，效率极高。此时可能有读者问，随机读可不可以这样优化？答案是：看情况。通常而言，如果读并发度不高，则不可以这么做，因为如果将多个读重新排列组合后再执行，系统的响应时间太慢，用户可能接受不了，而如果读并发度极高，也许可以采用类似机制。

经验技巧 7　如何解决求职中的时间冲突问题？

对于求职者而言，求职季就是一个赶场季，一天少则几家、十几家企业入校招聘，多则几十家、上百家企业招兵买马，企业多，选择项自然也多，这固然是一件好事情，但由于招聘企业实在是太多，自然而然会导致另外一个问题的发生：同一天企业扎堆，且都是自己心仪或欣赏的大牛企业的现象。如果不能够提前掌握企业的宣讲时间、地点，是很容易迟到或错过的。但有时候即使掌握了宣讲时间、笔试和面试时间，还是有可能错过，为什么呢？时间冲突，人不可能具有分身术，也不可能同一时间做两件不同的事情，所以，很多时候就必须有所取舍了。

到底该如何取舍呢？该如何应对这种时间冲突的问题呢？在此，编者将自己的一些想法和经验分享出来，以供读者参考：

1）如果多家心仪企业的校园宣讲时间发生冲突（前提是只宣讲，不笔试，否则请看后面的建议），此时最好的解决方法是和同学或朋友商量好，各去一家，然后大家进行信息共享。

2）如果多家心仪企业的笔试时间发生冲突，此时只能选择其一，毕竟企业的笔试时间都是考虑到了成百上千人的安排，需要提前安排考场、考务人员和阅卷人员等，不可能为了某一个人而轻易改变。所以，最好选择自己更有兴趣的企业参加笔试。

3）如果多家心仪企业的面试时间发生冲突，不要轻易放弃。对于面试官而言，面试任何人都是一样的，因为面试官谁都不认识，而面试时间也是灵活性比较大的，一般可以通过电话协商。求职者可以与相关工作人员（一般是企业的 HR）进行沟通，以某种理由（例如学校的事宜、导师的事宜或家庭的事宜等，前提是必须能够说服人，不要给出的理由连自己都说

服不了）让其调整时间，一般都能协调下来。但为了保证协调的成功率，一般要接到面试通知后第一时间联系相关工作人员变更时间，这样他们协调起来也更方便。

正如世界上没有能够包治百病的药物一样，以上这些建议在应用时，很多情况下也做不到全盘兼顾，当必须进行多选一的时候，求职者就要对此进行评估了，评估的项目可以包括：对企业的中意程度、获得 offer 的概率及去工作的可能性等。评估的结果往往具有很强的参考性，求职者依据评估结果做出的选择一般也会比较合理。

真　题　篇

真题篇主要列举了 18 套来自于顶级 IT 企业的面试笔试真题，这些企业是行业的标杆，代表了行业的最高水准，而他们所出的面试笔试真题不仅难易适中，覆盖面广（包括语言基础、链表、算法和海量数据处理等内容），而且具有非常好的区分度，代表性非常强，是历年来程序员面试笔试中的必考项或常考项，且越来越多的中小企业开始从中摘取部分题目或者直接全盘照搬以作为面试笔试题。

真题 1　某知名互联网下载服务提供商软件工程师笔试题

一、选择题

1．访问修饰符作用范围由大到小是（　　）。

A．private-protected-default-public　　B．public-protected-default-private

C．private-default-protected-public　　D．public-default-protected-private

2．在 Java 语言中，下面接口以键-值对的方式存储对象的是（　　）。

A．java.util.List　　B．java.util.Map

C．java.util.Collection　　D．java.util.Set

3．以下不是 Object 类的方法的是（　　）。

A．hashCode()　　B．finalize()

C．notify()　　D．hasNext()

4．有如下代码：

```
public class Test
{
    public void change(String str, char ch[])
    {
        str = "test ok";
        ch[0] = 'g';
    }
    public static void main(String args[])
    {
        String str = new String("good");
        char[] ch = { 'a', 'b', 'c' };
        Test ex = new Test();
        ex.change(str, ch);
        System.out.print(str + "and ");
        System.out.print(ch);
    }
}
```

上面程序的运行结果是（　　）。

A．good and abc　　B．good and gbc

C．test ok and abc　　D．test ok and gbc

二、填空题

1．Math.round(12.5)的返回值等于（　　），Math.round(-12.5)的返回值等于（　　）。

2．有如下程序：

```
String str1="hello world";
String str2="hello"+newString("world");
System.out.println (str1==str2);
```

那么程序的运行结果是（　　）。

3．在 Java 语言中，基本数据类型包括（　　）、字符类型（　　）、布尔类型 boolean 和数值类型（　　）。

4．字符串分为两大类：一类是字符串常量（　　）；另一类是字符串变量（　　）。

三、简答题

1．接口和抽象类有什么区别？
2．实现多线程的方法有哪几种？
3．利用递归方法求 6!
4．用 Java 语言实现一个观察者模式。
5．一个有 10 亿条记录的文本文件，已按照关键字排好序存储，请设计一个算法，可以从文件中快速查找指定关键字的记录。

真题 2　某知名社交平台软件工程师笔试题

一、单项选择题

1．二进制数 11101 转化为十进制数是（　　）。

A．23　　B．17　　C．26　　D．29

2．以下可以对对象加互斥锁的关键字是（　　）。

A．synchronized　　B．serialize　　C．volatile　　D．static

二、不定项选择题

1．下列关于类的构造方法的描述中，正确的是（　　）。

A．类中的构造方法不可省略
B．构造方法必须与类同名，但方法不能与 class 同名
C．构造方法在一个对象被 new 时执行
D．一个类只能定义一个构造方法

2．下列关于 Java 语言中 main 方法的描述中，正确的是（　　）。

A．Java 程序的 main 方法必须写在类里面
B．Java 程序中可以有多个 main 方法
C．Java 程序的 main 方法中，如果只有一条语句，可以不用大括号{}括起来
D．Java 程序中类名必须与文件名一样

3．在类声明中，声明一个类不能再被继承的关键字是（　　）。

A．private　　B．abstract　　C．final　　D．static

4．下面关于关键字 abstract 的描述中，正确的是（　　）。

A．关键字 abstract 可以修饰类或方法
B．final 类的方法都不能是 abstract，因为 final 类不能有子类
C．abstract 类不能实例化
D．abstract 类的子类必须实现其超类的所有 abstract 方法

5．以下不是合法标识符的是（　　）。

A．STR　　B．x3ab　　C．void　　D．abcd

6．以下关于类的描述中，正确的是（　　）。

A．只要没有定义不带参数的构造方法，JVM 都会为类生成一个默认构造方法
B．局部变量的作用范围仅仅在定义它的方法内，或者是在定义它的控制流块中
C．使用其他类的方法仅仅需要引用方法的名字即可
D．在类中定义的变量称为类的成员变量，在其他类中可以直接使用

7．有如下代码：

```
public class Test
{
    public static void main(String args[])
    {
        int i;
        i = 6;
```

```
            System.out.print(i);
            System.out.print(i++);
            System.out.print(i);
        }
}
```

以上程序的运行结果是（　　）。

A．666　　B．667　　C．677　　D．676

8．下列关于 Java 语言中关键字 super 的说法中，正确的是（　　）。

A．关键字 super 是在子类对象中指代其父类对象的引用

B．子类通过关键字 super 只能调用父类的属性，而不能调用父类的方法

C．子类通过关键字 super 只能调用父类的方法，而不能调用父类的属性

D．关键字 super 不仅可以指代子类的直接父类，还可以指代父类的父类

9．下面关于 String、StringBuilder 以及 StringBuffer 的描述中，正确的是（　　）。

A．对 String 对象的任何改变都不影响到原对象，相关的任何 change 操作都会生成新的对象

B．StringBuffer 是线程安全的

C．StringBuilder 是线程安全的

D．可以修改 StringBuilder 和 StringBuffer 的内容

10．以下不是基本数据类型的类型有（　　）。

A．int　　B．String　　C．Byte　　D．Float

11．JavaThread 中的方法 resume()负责恢复哪些线程的执行？（　　）

A．通过调用 wait()方法而停止运行的线程

B．通过调用 sleep()方法而停止运行的线程

C．通过调用 stop()方法而停止的线程

D．通过调用 suspend()方法而停止运行的线程

12．有如下代码：

```
public class Test
{
    public static int testException(int i)throws Exception
    {
        try
        {
            return i / 5;
        }
        catch (Exception e)
        {
            throw new Exception("exception in a aMethod");
        }
        finally{
            System.out.printf("finally");
        }
    }
    public static void main(String[] args)
    {
```

```
        try
        {
            testException(0);
        }
        catch (Exception ex)
        {
            System.out.printf("exception in main");
        }
        System.out.printf("finished");
    }
}
```

以上这段代码编译运行后，输出的结果是（　　）。

A．finallyexception in mainfinished　　B．finallyfinished

C．exception in mainfinally　　D．finallyexception in mainfinished

13．释放掉一个指定占据的内存空间的方法是（　　）。

A．调用 system.gc()方法　　B．调用 free()方法

C．赋值给该项对象的引用为 null　　D．程序员无法明确强制垃圾回收器运行

14．以下关于 Spring 框架的描述中，正确的是（　　）。

A．Spring 是“依赖注入”模式的实现

B．Spring 是一个轻量级 Java EE 的框架集合

C．使用 Spring 可以实现声明事务

D．Spring 提供了 AOP 方式的日志系统

15．堆的形状是一棵（　　）。

A．完全二叉树　　B．平衡二叉树

C．二叉排序树　　D．满二叉树

16．下列关于依赖注入的描述中，正确的是（　　）。

A．依赖注入提供使用接口编程

B．依赖注入使组件之间相互依赖，相互制约

C．依赖注入能够独立开发各组件，然后根据组件间关系进行组装

D．依赖注入指对象在使用时动态注入

17．以下关于 HashMap 与 HashTable 的说法中，正确的是（　　）。

A．迭代 HashMap 采用快速失败机制，而 HashTable 不是

B．HashTable 允许 null 值作为 key 和 value，而 HashMap 不可以

C．HashMap 不是同步的，而 HashTable 是同步的

D．两者都是用 key-value 方式获取数据

18．list 是一个 ArrayList 的对象，当将选项（　　）的代码填到//todo delete 处时，可以在 Iterator 遍历的过程中正确并安全地删除一个 list 中保存的对象。

```
Iterator it = list.iterator();
int index = 0;
while (it.hasNext())
{
    Object obj = it.next();
    if (needDelete(obj)) //needDelete 返回 boolean，决定是否要删除
    {
```

```
            //todo delete
        }
        index++;
}
```

A．it.remove()　　B．list.remove(index)

C．list.remove(obj)　　D．list.delete(index)

19．以下属于算法结构的是（　　）。

A．输入数据　　B．处理数据　　C．输出结果　　D．存储数据

20．已知某二叉树的后序遍历序列是 dabec，中序遍历序列是 debac，那么它的前序遍历序列是（　　）。

A．abcde　　B．dceab　　C．deabc　　D．cedba

21．算法的空间复杂度是指（　　）。

A．算法程序的长度　　B．算法程序中的指令条数

C．算法程序所占的存储空间　　D．算法执行过程中所需要的存储空间

22．二叉树是非线性数据结构，以下关于其存储结构的描述中，正确的是（　　）。

A．它不能用链式存储结构存储

B．它不能用顺序存储结构存储

C．顺序存储结构和链式存储结构都不能使用

D．顺序存储结构和链式存储结构都能存储

23．在一棵二叉树上，第 4 层的结点数最多是（　　）。

A．8　　B．16　　C．32　　D．64

24．设一组初始记录关键字序列（5，2，6，3，8），以第一个记录关键字 5 为基准进行一趟快速排序的结果为（　　）。

A．3，2，5，8，6　　B．2，3，5，8，6

C．3，2，5，6，8　　D．2，3，6，5，8

25．事务隔离级别是由（　　）实现的。

A．Hibernate　　B．Java 应用程序

C．数据库系统　　D．JDBC 驱动程序

26．设指针变量 p 指向双向链表中结点 A，指针变量 s 指向被插入的结点 X，则在结点 A 的后面插入结点 X 的操作序列为（　　）。

A．s->left=p;s->right=p->right;p->right=s;p->right->left=s

B．s->left=p;s->right=p->right;p->right->left=s;p->right=s

C．p->right=s;s->left=p;p->right->left=s;s->right=p->right

D．p->right=s;p->right->left=s;s->left=p;s->right=p->right

27．在排序方法中，从未排序序列中挑选元素，并将其依次插入已排序序列（初始时为空）的一端的方法，称为（　　）。

A．归并排序　　B．希尔排序　　C．插入排序　　D．选择排序

28．操作系统的功能是进程处理机管理、（　　）管理、（　　）管理、文件管理和作业管理等。

A．设备　　B．存储器　　C．硬件　　D．软件

29．下列中断属于强迫性中断的是（　　）。

A．掉电　　B．设备出错　　C．时间片到时　　D．执行 print 语句

30．进程调度是从（　　）选择一个进程投入运行。

A．就绪队列　　B．作业后备队列　　C．等待队列　　D．提交队列

31．“死锁”是针对（　　）的。

A．某个进程申请资源数超过了系统拥有的最大资源数

B．某个进程申请系统中不存在的资源

C．硬件故障

D．多个并发进程竞争独占型资源

32．某系统中有 11 台打印机，N 个进程共享打印机资源，每个进程要求 3 台，当 N 的取值不超过（　　）时，系统不会发生死锁。

A．3　　B．5　　C．8　　D．7

33．IP 协议属于（　　）。

A．网络互联层　B．应用层　　C．数据链路层　D．传输层

34．将网络物理地址转换为 IP 地址的协议是（　　）。

A．IP　　B．ICMP　　C．ARP　　D．RARP

35．对于 IP 地址 130.63.160.2，掩码为 255.255.255.0，子网号为（　　）。

A．160.2　　B．160　　C．63.160　　D．63.160.2

36．对于 IP 地址 200.5.6.4，属于（　　）类地址。

A．A　　B．B　　C．C　　D．D

37．一个广域网和一个局域网相连，需要的设备是（　　）。

A．NIC　　B．网关　　C．集线器　　D．路由器

三、问答题

1．List<? extends T>和 List<? super T>之间有什么区别？

2．给出两种单例模式的实现方法，并说明这两种方法的优缺点。

3．描述 Java 语言中抽象基类和接口各自主要的使用场景。

4．int 和 Integer 的区别是什么？

5．已知两个链表 head1 和 head2 各自有序，请把它们合并成一个依然有序的链表。结果链表要包含 head1 和 head2 的所有结点，即结点值相同。

6．给定 a、b 两个文件，各存放 50 亿个 url，每个 url 各占 64B，内存限制是 4GB，请找出文件 a 与文件 b 中共同的 url。

真题 3　某知名安全软件服务提供商软件工程师笔试题

一、不定项选择题

1．“hello” instanceof Object 的返回值是（　　）。

A．“abcd”　　B．true　　C．false　　D．String

2．下面有关方法覆盖的描述中，不正确的是（　　）。

A．覆盖的方法一定不能是 private 的

B．要求覆盖和被覆盖的方法必须具有相同的访问权限

C．覆盖的方法不能比被覆盖的方法抛出更多的异常

D．要求覆盖和被覆盖的方法有相同的名字、参数列以及返回值

3．下面说法正确的是（　　）。

A．如果源代码中有 package 语句，则该语句必须被放在代码的第一行（不考虑注释和空格）

B．如果源代码中有 main()方法，则该方法必须被放在代码的第一行

C．如果源代码中有 import 语句，则该语句必须被放在代码的第一行（不考虑注释和空格）

D．如果某文件的源代码中定义了一个 public 的接口，则接口名和文件名可以不同

4．下面变量名中合法的有（　　）。

A．2var　　B．var2　　C．_var　　D．_1_

E．$var　　F．#var

5．一个 Java 程序运行从上到下的环境次序是（　　）。

A．JRE/JVM、操作系统、Java 程序、硬件

B．Java 程序、JRE/JVM、硬件、操作系统

C．Java 程序、JRE/JVM、操作系统、硬件

D．Java 程序、操作系统、JRE/JVM、硬件

6．下面关键字中，可以用来修饰接口中的变量的是（　　）。

A．static　　B．private　　C．synchronized　　D．protected

7．有如下代码：

String s=“xbcde”;

System.out.println(s.charA.t(4));

以下针对上述代码段的描述中，正确的是（　　）。

A．输出字符 e

B．什么都没有，抛出 ArrayIndexOutOfBoundsException

C．输出字符 d

D．代码编译不成功，因为 charA.t()方法不属于 String 类

8．下面创建 Map 集合的方式中，正确的是（　　）。

A．Map m=new Map(new Collection())　　B．Map m=new Map(10, 2,40)

C．Map m=new Map()　　D．Map 是接口，所以不能实例化

9．以下关于被访问控制符 protected 修饰的成员变量的描述中，正确的是（　　）。

A．可以被三种类所引用：该类自身、与它在同一个包中的其他类、在其他包中的该类的子类

B．只能被该类自身所访问和修改

C．可以被两种类访问和引用：该类本身、该类的所有子类

D．只能被同一个包中的类访问

10．为了区分类中重载的同名的不同方法，要求（　　）。

A．采用不同的形式参数列表　　B．采用不同的返回值类型

C．调用时用类名或者对象名作前缀　　D．采用不同的参数名

11．下列对于构造方法的描述中，正确的是（　　）。

A．构造方法必须用 void 声明返回类型

B．构造方法名必须与类名相同

C．构造方法可以被程序调用

D．如果编程人员没在类中定义构造方法，程序将报错

12．下列有关继承的描述中，正确的是（　　）。

A．子类能继承父类的非私有方法和属性

B．子类能继承父类的所有方法和属性

C．子类只能继承父类的公有方法和属性

D．子类能继承父类的方法，而不是属性

13．下面有关子类继承父类构造方法的描述中，正确的是（　　）。

A．创建子类的对象时，先调用子类自己的构造方法，然后调用父类的构造方法

B．子类会继承父类的构造方法

C．子类必须通过关键字 super 调用父类的构造方法

D．子类无法继承父类的构造方法

14．下列关于 Java 语言基础知识的描述中，正确的是（　　）。

A．类是方法和变量的集合体　　B．抽象类或接口可以被实例化

C．数组是无序数据的集合　　D．类成员数据必须是公有的

15．有如下代码：

```
public class Test
{
        public static void main(String[] args)
        {
            class A
            {
                public int i=3;
            }
            Object o = (Object)new A();
            A a = (A)o;
            System.out.println("i = "+ a.i);
        }
}
```

上述程序运行后的结果是（　　）。

A．i=3　　B．编译失败

C．运行结果为 ClassCastException　　D．i=0

二、填空题

1．用于声明一个类为抽象类的关键字是（　　），用于将一个类修饰为最终类的关键字是（　　）。

2．构造方法、成员变量初始化以及静态成员变量初始化三者的先后顺序是（　　）。

3．在 Java 语言的基本数据类型中，字符型、整型分别占用字节数为（　　）、（　　）。

4．一般有两种用于创建线程对象的方法，分别是（　　）与（　　）。

5．Java 语言提供了两种用于多态的机制，分别是（　　）与（　　）。

三、问答题

1．接口能否继承接口？抽象类是否可实现（implements）接口？抽象类是否可继承实体类？

2．面向对象的特征有哪些方面？

3．String 和 StringBuffer 有什么区别？

4．final、finally 和 finalize 的区别是什么？

5．ArrayList、Vector 和 LinkedList 有什么特点？HashMap 和 HashTable 有什么区别？

四、附加题

1．编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。但是要保证汉字不被截半个，例如“人 ABC”4，应该截为“人 AB”，输入“人 ABC 们 DEF”，6，应该输出为“人 ABC”而不是“人 ABC+们的半个”。

2．排序有哪几种方法？用 Java 语言实现一个插入排序？

真题 4　某知名互联网金融企业软件工程师笔试题

一、单项选择题

1．下列描述中，正确的是（　　）。

A．Java 程序经编译后会产生 Machine Code（机器码）

B．Java 程序经编译后会产生 Byte Code（字节码）

C．Java 程序经编译后会产生 DLL（动态链接库）

D．以上描述都不正确

2．Java 语言是从（　　）语言改进重新设计的。

A．BASIC　　B．C++　　C．Pascal　　D．Ada

3．下列关于类的描述中，正确的是（　　）。

A．类中的构造方法不可省略

B．一个类只能定义一个构造方法

C．new 一个对象的时候构造方法会被调用

D．构造方法必须与类同名，但普通方法不能与类同名

4．下列选项中，提供了 Java 存取数据库能力的包是（　　）。

A．java.sql　　B．java.swing

C．java.io　　D．java.awt

5．下列运算符合法的是（　　）。

A．&&　　B．<>　　C．while　　D．:=

6．有如下代码：

```
a=0;
c=0;
do{
--c;
a=a-1;
}while(a>0);
```

当执行完以上代码后，变量 c 的值是（　　）。

A．-2　　B．1　　C．-1　　D．死循环

7．下列关于 abstract 的描述中，正确的是（　　）。

A．abstract 修饰符可修饰属性、方法和类

B．抽象方法的方法体必须用一对大括号包住

C．抽象方法的方法体（大括号）可有可无

D．声明抽象方法不可写出大括号

8．下列关于形式参数的描述中，正确的是（　　）。

A．形式参数可被视为局部变量

B．形式参数不可以是对象

C．形式参数为方法被调用时真正被传递的参数

D．形式参数可被字段修饰符修饰

9．下列关于实例方法的描述中，正确的是（　　）。

A．实例方法可直接调用超类的类方法

B．实例方法可直接调用超类的实例方法

C．实例方法可直接调用其他类的实例方法

D．实例方法可直接调用本类的类方法

10．下列关于 Java 语言的描述中，正确的是（　　）。

A．Java 语言容许单独的过程与函数存在

B．Java 语言容许单独的方法存在

C．Java 语言中的方法属于类中的成员

D．Java 语言中的方法必定隶属于某一类（对象）

二、多项选择题

1．下列关于 Java 语言的编译过程的描述中，正确的有（　　）。

A．环境变量可在编译 source code 时指定
B．在编译程序时，所能指定的环境变量不包括 class path
C．javac 一次可同时编译数个 Java 源文件
D．javac.exe 能指定编译结果要置于哪个目录（directory）

2．下列标识符中，不合法的有（　　）。
A．if　　B．$Usdollars
C．12345　　D．t.txt

3．下列关于数组的描述中，错误的有（　　）。
A．数组是一种对象　　B．数组是一种原生类
C．int arr=[]={1,2,3,4}　　D．数组的大小可以随意改变

4．以下不能用来修饰接口的有（　　）。
A．private　　B．public　　C．abstract　　D．static

5．下列关于按值传递与按引用传递的描述中，正确的是（　　）。
A．按值传递不会改变实际参数的数值
B．按引用传递能改变实际参数的参考地址
C．按引用传递能改变实际参数的内容
D．按引用传递不能改变实际参数的参考地址

6．下列关于类方法的调用的描述中，错误的有（　　）。
A．在类方法中可用 this 来调用本类的类方法
B．在类方法中调用本类的类方法时可直接调用
C．在类方法中只能调用本类中的类方法
D．在类方法中绝对不能调用实例方法

7．下列关于 Java 语言基础知识的描述中，错误的有（　　）。
A．能被 java.exe 成功运行的 java class 文件必须有 main()方法
B．J2SDK 就是 Java API
C．Appletviewer.exe 可利用 jar 选项运行.jar 文件
D．能被 Appletviewer 成功运行的 java class 文件必须有 main()方法

三、判断题

1．Unicode 是用 16 位来表示一个字的。（　　）
2．Java 程序中的起始类名称必须与存放该类的文件名相同。（　　）
3．原生类中的数据类型均可任意转换。（　　）

四、程序题

1．写出下列程序的运行结果

```
public class Test
{
    public static void main(String[] args)
    {
        String a = "hello";
        change(a);
        System.out.println(a);
    }
    public static void change(String name)
    {
        name="world";
```

```
    }
}
```

2．写出下面程序运行的结果

```
public class Test
{
    static boolean f(char c)
    {
        System.out.print(c);
        return true;
    }
    public static void main(String[] argv)
    {
            int i = 0;
            for (f('A'); f('B') && (i < 2); f('C'))
            {
                i++;
                f('D');
            }
     }
 }
```

五、简答题

1．HashMap 和 HashTable 的区别是什么？

2．&和&&的区别是什么？

3．Collection 和 Collections 的区别是什么？

4．abstract class 和 interface 的区别是什么？

5．Final、finally 和 finalize 的区别是什么？

六、加分题

1．什么是设计模式？有哪些常见的设计模式？

2．请简要介绍 Spring MVC、IoC 和 AOP。

真题 5　某知名搜索引擎提供商软件工程师笔试题

一、单选题

1．下列关于实例方法的描述中，正确的是（　　）。

A．实例方法可直接调用超类的类方法

B．实例方法可直接调用超类的实例方法

C．实例方法可直接调用其他类的实例方法

D．实例方法可直接调用本类的类方法

2．下列关于抽象方法的描述中，正确的是（　　）。

A．抽象方法的 body 部分必须用一对大括号{ }包住

B．abstract 修饰符可修饰字段、方法和类

C．声明抽象方法，大括号可有可无

D．声明抽象方法不可写出大括号

3．有如下代码：

```
public class Test
{
```

```
    public int f()
    {
        static int i = 0;
        i++;
        return i;
    }
    public static void main(String args[])
    {
        Test test = new Test();
        test.f();
        int j = test.f();
        System.out.println(j);
    }
}
```

上述代码的输出结果是（　　）。

A．0　　B．2　　C．1　　D．编译失败

4．有如下代码：

```
class Super
{
    public Integer getLenght()
    {
        return new Integer(4);
    }
}

public class Sub extends Super
{
    public Long getLenght()
    {
        return new Long(5);
    }

    public static void main(String[] args)
    {
         Super super = new Super();
         Sub sub = new Sub();
          System.out.println(super.getLenght().toString() + "," +sub.getLenght());
    }
}
```

上述代码的输出结果是（　　）。

A．4,5　　B．4,4　　C．5,4　　D．编译失败

5．Servlet 处理请求的方式为（　　）。

A．以程序的方式　　B．以进程的方式

C．以线程的方式　　D．以响应的方式

6．在 JDBC 中，用于表示数据库连接的对象是（　　）。

A．Statement　　B．Connection　　C．PreparedStatement　D．DriverManager

7．在 Java 语言中，用于调用存储过程的对象是（　　）。

A．DriverManager　B．ResultSet　　C．CallableStatemet　D．PreparedStatement

8．下面关于垃圾回收的描述中，正确的是（　　）。

A．对象空间被回收掉之后，会执行该对象的 finalize 方法

B．一个对象一旦成为垃圾，就立刻被回收

C．finalize 方法和 C++语言的析构函数完全是一回事

D．一个对象成为垃圾是因为不再有引用指着它，但是线程并非如此

9．按照 MVC 设计模式，JSP 用于实现（　　）。

A．Controller（控制器）　　B．View（视图）

C．Model（模型）　　D．Database（数据库）

10．有如下代码：

```
10) public Object m() {
11) Object o = new Float(3.1f);
12) Object [] oa = new Object[1];
13) oa[0] = o;
14) o = null;
15) oa[0] = null;
16) print 'return 0';
17) }
```

当 Float 对象在第 11 行被创建后，（　　）能够被垃圾回收。

A．14 行以后　　B．13 行以后　　C．15 行以后　　D．16 行以后

11．有如下代码：

```
class Base
{
    Base()
    {
        System.out.print("Base");
    }
}

public class Alpha extends Base
{
    public static void main(String[] args)
    {
        new Alpha();
        new Base();
    }
}
```

上述代码的输出结果是（　　）。

A．Base　　B．BaseBase　　C．运行失败　　D．编译失败

12．在 J2EE 中，属于 Web 层的组件有（　　）。

A．Servlet　　B．HTML　　C．Applet　　D．EJB

13．以下关于异常的描述中，正确的是（　　）。

A．如果一个方法声明将抛出某个异常，它就必须真的抛出那个异常

B．一旦出现异常，程序运行就终止了

C．在 catch 子句中匹配异常是一种精确匹配

D．可能抛出系统异常的方法是不需要声明异常的

14．有如下代码：

```
public class Test
{
    public static void main(String[] args)
    {
        try
        {
            return;
        }
        finally
        {
            System.out.println("Finally");
        }
    }
}
```

上述代码的输出结果是（　　）。

A．Finally　　B．编译失败

C．运行时抛出异常　　D．代码正常运行但没有任何输出

15．在 JSP 指令中，isELIgnored="boolean"的意思是（　　）。

A．决定该页面是否是一个错误处理页面　B．决定是否实现 servlet 的单线程模式

C．决定是否支持 EL 表示　　D．没有具体的含义

16．以下关于 Java 语言中的引用的描述中，正确的是（　　）。

A．引用实际上就是指针　　B．引用本身是 Primitive

C．一个对象只能被一个引用所指引　　D．引用就是对象本身

17．以下关于 import java.util 包的描述中，错误的是（　　）。

A．Vector 类放在/java/util/目录下　　B．Vector 类属于 java.util 包

C．Vector 类放在 java.util 文件中　　D．Vector 类是 Sun 公司的产品

18．下列属于容器型构件的是（　　）。

A．JButton　　B．JEdit　　C．JPanel　　D．JTextField

19．在一个线程中，sleep(100)方法将使得该线程在（　　）后获得对 CPU 的控制（假设睡眠过程中不会有其他事件唤醒该线程）。

A．正好 100ms　　B．100ms 不到　　C．≥100ms　　D．不一定

20．下面不是 Java 语言关键字的是（　　）。

A．integer　　B．float　　C．double　　D．default

21．在 WEB-INF 目录下，必须存放的文件为（　　）。

A．class 文件　　B．web.xml　　C．html 文件　　D．jar 文件

22．表达式 4&7 的运算结果是（　　）。

A．4　　B．1　　C．6　　D．7

23．有如下代码：

long temp=(int)3.9;
temp%=2;
那么，变量 temp 的最终值是（　　）。

A．0　　B．1　　C．2　　D．4

24．以下可以替换 URL 中的 session ID 的方法是（　　）。

A．HttpServletRequest 接口的 encodeURL 方法

B．HttpServletResponse 接口的 encodeURL 方法

C．HttpServletResponse 接口的 rewriteURL 方法

D．HttpServletRequest 接口的 rewriteURL 方法

25．每个使用 Swing 构件的程序必须有一个（　　）。

A．标签　　B．按钮　　C．菜单　　D．容器

26．下列标识符命名原则中，正确的是（　　）。

A．变量和方法名的首写字母大写　　B．类名的首字母小写

C．接口名的首写字母小写　　D．常量完全大写

27．类 Test 定义如下：

```
1) public class Test{
2)   public float f(float a，float b){ return 0;}
3)
4) }
```

将选项（　　）中代码插入第 3 行是不合法的。

A．public float f (float a，　float b，float c) { return 0;}

B．public float f (float c，float d) { return 0;}

C．public int f (int a，　int b) { return 0;}

D．private float f (int a，int b，float c) { return 0;}

28．以下描述中，能够创建一个数组实例的是（　　）。

A．int[] arr = new int [10];　　B．float fa = new float [10];

C．char[] ca = “hello”;　　D．int ia[][] = {1, 2, 3} {4, 5, 6};

29．以 public 修饰的类如下所示：public class Car{…}，则类 Car（　　）。

A．可被其他程序包中的类使用　　B．不能被其他类继承

C．不能被任意其他类使用　　D．仅能被本程序包中的类使用

30．Java 程序的执行过程中用到一套 JDK 工具，其中，java.exe 是指（　　）。

A．Java 编译器　　B．Java 解释器

C．Java 文档生成器　　D．Java 类分解器

31．下列关于构造方法的描述中，错误的是（　　）。

A．Java 语言规定构造方法没有返回值，但不用 void 声明

B．Java 语言规定构造方法名与类名必须相同

C．Java 语言规定构造方法不可以重载

D．Java 语言规定构造方法不能直接被调用

32．构造方法调用的时间是（　　）。

A．定义类时　　B．创建对象时

C．使用对象的变量时　　D．调用对象方法时

33．以下关于关键字 break 的描述中，正确的是（　　）。

A．只中断最外层的循环　　B．只中断最内层的循环

C．借助于标号，可以实现任何外层循环中断 D．只中断某一层的循环

34．在 Java 语言中，下面可以用作正确的变量名称的是（　　）。

A．1x　　B．age　　C．extends　　D．implements

35．在 JavaScript 中，以下验证一个数据是否是数字的描述中，正确的是（　　）。

A．int I = value 若报错就不是数字

B．如果用 Integer.parseInt(value)有误就不是数字

C．没有方法验证

D．利用 isNaN(value) 返回的 boolean 进行判断

36．以下不能作 JSP 的服务器的是（　　）。

A．JBoss　　B．BEA WebLogic　　C．Tomcat　　D．PWS

37．以下不是 JSP 操作指令的是（　　）。

A．setProperty　　B．include　　C．forward　　D．import

38．下面不是 Java 类访问控制关键字的是（　　）。

A．private　　B．protected　　C．this　　D．public

39．如果希望控件在界面上按表格行分列排列，应使用的布局管理器是（　　）。

A．BoxLayout　　B．GridLayout　　C．FlowLouLayout　　D．BorderLayout

40．在配置 tomcat 虚拟目录时，需要打开的文件是（　　）。

A．web.xml　　B．index.jsp　　C．server.xml　　D．以上都不是

41．下面不是表单标记的是（　　）。

A．RADIO　　B．INPUT　　C．CHECKBOX　　D．TR

42．下面不是 response 对象的方法的是（　　）。

A．addCookie(Cookie cookie)

B．setHeader(String headername,String headervalue)

C．getParameter(String str)

D．sendError(int errorcode)

43．以下是编写 Servlet 必须导入的包的是（　　）。

A．java.sql.*　　B．java.servlet.*　　C．java.util.*　　D．java.io.*

44．下面不属于 SQL 语句的子类的是（　　）。

A．数据查询语言（DQL）　　B．数据定义语言（DDL）

C．事务控制语言（TCL）　　D．数据插入语言（DIL）

45．有如下代码：

```
public class Outer
{
    public void    someOuterMethod()
    {
        // Line 3
    }
    public class Inner{}
    public static void    main(String[]argv)
    {
        Outer o = new Outer();
        // Line 8
    }
}
```

内部类里面实例化了一个实例的是（　　）。

A．new Inner(); // At line 3　　B．new Inner(); // At line 8
C．new Outer.Inner(); // At line 8　　D．new o.Inner(); // At line 8

二、多选题

1．以下声明中，能够防止方法覆盖的有（　　）。
A．final void f() {}　　B．void final f() {}
C．static void f() {}　　D．static final void f() {}
E．final abstract void f() {}

2．下列属于 JSP 中注释的有（　　）。
A．<!-- 与 -->　　B．/　　C．/** 与 **/　　D．<%-- 与 --%>

3．按照学生平均成绩（avg_grade）将 students 表中的数据检索出来，下面 SQL 语句正确的是（　　）。
A．SELECT * FROM students ORDER BY avg_grade
B．SELECT * FROM students GROUP BY avg_grade ASC
C．SELECT * FROM students GROUP BY avg_grade DESC
D．SELECT * FROM students ORDER by avg_grade asc

4．下列是 JSP 作用域的通信对象的有（　　）。
A．application　　B．session　　C．pageContext　　D．cookie

5．在接口中，以下定义正确的是（　　）。
A．void f();　　B．public double f();
C．public final double f();　　D．static void f(double d1);
E．protected void f(double d1);

6．下面语句中，正确地声明一个整型的二维数组的有（　　）。
A．int　arr[][] = new　int[][];　　B．int　arr[20][10] = new　int[][];
C．char　a[][] = new　char[10][10];　　D．int　[][]a = new　int[10][20];
E．int　[]a[] = new　int[10][10];

7．下面不是 Java 语言的原始数据类型的有（　　）。
A．int　　B．Boolean　　C．Double　　D．char

8．下面能够生成 5 个空字符串的语句有（　　）。
A．String　a[]=new　String[5]；for(int　i=0；i<5；a[i++]=“”)；
B．String　a[]={“”，“”，“”，“”，“”}；
C．String　a[5]；
D．String　[]a=new　String[5]；for(int　i=0；i<5；a[i++]=null)；

9．以下关于数组的描述中，错误的有（　　）。
A．数组是一种对象　　B．数组属于一种原生类
C．int number=[]={1,2,3,4,5}　　D．数组的大小可以随意改变

10．不能用来修饰 interface 的有（　　）。
A．private　　B．public　　C．final　　D．static

11．下列关于类方法的描述中，错误的有（　　）。
A．在类方法中可用 this 来调用本类的类方法
B．在类方法中调用本类的类方法时可直接调用
C．在类方法中绝对不能调用实例方法
D．在类方法中只能调用本类中的类方法

12．有如下代码：

```
class A {
```

A() { }
}
class B extends A {
}
关于上述代码，以下描述正确的是（　　）。
A．B 类的构造方法一定是 public　B．B 类的构造方法应该是没有参数
C．B 类的构造方法应该调用 this()　D．B 类的构造方法应该调用 super()
13．下列标识符中，不合法的有（　　）。
A．if　B．$aa　C．12　D．a.txt
14．以下能使用 throw 抛出异常的有（　　）。
A．Throwable　B．Event　C．Object　D．Error
E．Exception　F．RuntimeException
15．在 javax.Servlet 的包中，属于类的是（　　）。
A．Servlet　B．ServletException
C．GenericServlet　D．ServletContext
16．有如下代码：
```
public class X {
    public X f() { return this;}
}
public class Y extends X {
}
```
以下能添加到 Y 类的定义中的方法是（　　）。
A．public void f() {}　B．private void f() {}
C．public void f(String s) {}　D．private Y f() { return null; }
E．public X f() { return new Y(); }
17．有如下代码<% int i = Integer.parseInt(request.getParemeter(“value”)) %>，下面描述正确的有（　　）。
A．不会有错
C．当 value＝＂＂时会报错
B．当 value 与 int 类型不匹配时会报错
D．为了安全起见应该将该段代码放在 try{}和 catch(){}之间
18．以下哪两个是等价的？（　　）
A．<%= YoshiBean.size%>
B．<%= YoshiBean.getSize()%>
C．<%= YoshiBean.getProperty("size")%
D．<JSP:getProperty id="YoshiBean" param="size"/>
E．<jsp:getProperty id="YoshiBean" property="size"/>
F．<jsp:getProperty name="YoshiBean" param="size"/>
G．<jsp:getProperty name="YoshiBean" property="size"/>
19．以下关于构造方法的描述中，正确的有（　　）。
A．默认构造方法初始化方法变量
B．默认构造方法调用其父类的无参构造器
C．默认构造方法有和它所在类相同的访问修饰符
D．如果一个类没有无参构造方法，编译器会为它创建一个默认构造方法

E．只有当一个类没有任何构造方法时，编译器才会为它创建一个默认构造方法

20．在 Servlet 的生命周期中，容器只调用一次的方法是（　　）。

A．getServletConfig　　B．service　　C．init　　D．destroy

三、简答题

1．关键字 static 的作用是什么？

2．JSP 和 Servlet 有哪些相同点和不同点？它们之间的联系是什么？

3．switch 是否能作用在 byte 上？是否能作用在 long 上？是否能作用在 String 上？

4．数据连接池的工作机制是什么?

5．多线程同步有几种实现方法？

6．HTML 的 Form 和 XForm 的区别是什么？

7．forward 和 redirect 的区别是什么？

8．Overload 和 Override 的区别是什么？Overload 的方法是否可以改变返回值的类型？

9．下面的 Java 代码保存在 B.java 文件中是否合法？

```
class A
{
    publicstaticvoid main(String args[])
    {
        System.out.println("Hello World");
    }
}
```

10．Servlet 和 CGI 的区别是什么？

四、编程题

1．对数组进行顺序排列。

2．用 Java 语言写一段访问 Oracle 数据库的程序，并实现数据查询。

3．请给出单例模式的实现代码。

4．用循环控制语句打印输出：1+3+5+…+99 的结果。

真题详解篇

真题详解篇主要针对 18 套真题进行深度剖析，在写法上，庖丁解牛，针对每一道题目都有非常详细的解答。授之以鱼的同时还授之以渔，不仅告诉答案，还告诉读者以后再遇到同类型题目时该如何解答。读者学完基础知识后，可以抽出一两个小时的时间来完成本书中的习题，找出自己的知识盲区，查漏补缺，为自己加油、补课。

真题详解 1　某知名互联网下载服务提供商软件工程师笔试题

一、选择题

1．答案：B。

分析：本题考察的是 Java 语言中访问修饰符作用范围的知识。

在 Java 语言中，类的权限访问修饰符有以下几种：private、default（package）、protected 和 public。下面具体对这几个权限访问修饰符进行介绍。

（1）私有权限（private）

private 可以修饰数据成员、构造方法及方法成员，不可以修饰类（此处指外部类，不考虑内部类）。被 private 修饰的成员，只能在定义它们的类中使用，在其他类中不能调用。

（2）默认权限（default）

类、数据成员、构造方法和方法成员等都能够使用默认权限，即不被 private、protected 和 public 修饰。默认权限即同包权限，同包权限的元素只能在定义它们的类中以及同包的类中被调用。

（3）受保护权限（protected）

protected 可以修饰数据成员、构造方法和方法成员，不可以修饰类（此处指外部类，不考虑内部类）。被 protected 修饰的成员，能在定义它们的类中以及同包的类中被调用。如果有不同包的类想调用它们，那么这个类必须是它的子类。

（4）公共权限（public）

public 可以修饰类、数据成员、构造方法及方法成员。被 public 修饰的成员，可以在任何一个类中被调用，不管同包或不同包，是权限最大的一个修饰符。

以上几种修饰符的使用范围见表 2（表中√表示可访问，×表示不可访问）。

表 2　修饰符的使用范围

范　　围	private	default	protected	public
同一类	√	√	√	√
同一包中的类	×	√	√	√
同一包中的类、不同包中的子类	×	×	√	√
所有	×	×	×	√

通过表 2 可知，访问修饰符的作用范围由大到小依次是 public、protected、default 和 private。所以，选项 B 正确。

所以，本题的答案为 B。

2．答案：B。

分析：本题考察的是 Java 语言中基本数据结构的知识。

对于选项 A，List 中保存了相同类型的多个元素，元素是按照存入的顺序存储的，元素可以重复。所以，选项 A 错误。

对于选项 B，Map 是以键-值对的方式来存储对象的，并且键不允许重复。所以，选项 B 正确。

对于选项 C，java.util.Collection 是一个集合接口，它提供了对集合对象进行基本操作的通用接口方法。而 Set 与 List 是它的两个具体的接口，由于 Set 与 List 都不是以键-值对的方式来存储对象的，因此，Collection 接口也不是。所以，选项 C 错误。

对于选项 D，Set 中也保存了相同类型的多个元素，元素是不能重复的。所以，选项 D 错误。

各接口的区别见表 3。

表 3　各接口的区别

类　　型		是否有序	是否允许重复	是否线程同步
Collection		否	是	—
List	ArrayList	否	是	否
	Vector			是
	LinkedList			否
Set	HashSet	否	否	否
	TreeSet	是		否
Map	HashMap	否	<key, value>，key 不允许重复	否
	TreeMap	是		否
	Hashtable	否		是

所以，本题的答案为 B。

3．答案：D。

分析：本题考察的是 Object 类的知识。

Object 类是类层次结构的根，在 Java 语言中，所有的类从根本上而言都继承自这个类。而且，Object 类是 Java 语言中唯一没有父类的类，而其他所有的类，包括标准容器类，例如数组，都继承了 Object 类。

具体而言，Object 类的方法见表 4。

表 4　Object 类的方法

方　法　名	返　回 类 型	方 法 描 述
clone()	Object	创建并返回此对象的一个副本
equals(Object obj)	boolean	判断 obj 对象是否与此对象相等
finalize()	void	当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法
getClass()	Class<?>	返回此 Object 的运行时类
hashCode()	int	返回该对象的哈希码值
notify()	void	唤醒在此对象监视器上等待的单个线程
notifyAll()	void	唤醒在此对象监视器上等待的所有线程
toString()	String	返回该对象的字符串表示
wait()	void	在其他线程调用此对象的 notify()方法或 notifyAll()方法前，使当前线程等待
wait(long timeout)	void	在其他线程调用此对象的 notify()方法或 notifyAll()方法，或者超过指定的时间量前，使当前线程等待
wait(long timeout, int nanos)	void	在其他线程调用此对象的 notify()方法或 notifyAll()方法，或者其他某个线程中断当前线程，或

		者已超过某个实际时间量前，使当前线程等待

由此可见，Object 类没有 hasNext()方法。所以，选项 D 正确。

所以，本题的答案为 D。

4．答案：B。

分析：本题考察的是 Java 语言中传参方式以及不可变类的知识。

不可变类（Immutable class）是指当创建了这个类的实例后，就不允许修改它的值了，也就是说，一个对象一旦被创建出来，在其整个生命周期中，它的成员变量就不能被修改了。它有点类似于常量（const），即只允许其他程序进行读操作，而不允许其他程序进行修改操作。

在 Java 类库中，所有基本类型的包装类都是不可变类，例如 Integer、Float 等。此外，String 也是不可变类。可能有人会有疑问，既然 String 是不可变类，为什么还可以写出如下代码来修改 String 类型的值呢？

```
public class Test
{
    public static void main(String[] args)
  {
        String s="Hello";
        s+=" world";
        System.out.println(s);
    }
}
```

程序的运行结果为

```
Hello world
```

表面上看，s+="world"的作用好像是修改 String 类型对象 s 的值，其实不是，String s="Hello"语句声明了一个可以指向 String 类型对象的引用，这个引用的名字为 s，它指向了一个字符串常量"Hello"。s+="world"并没有改变 s 所指向的字符串的值（由于"Hello"是 String 类型的对象，而 String 又是不可变量），在这行代码运行后，s 指向了另外一个 String 类型的对象，该对象的内容为"Hello world"。原来的那个字符串常量"Hello"还存在于内存中，并没有被改变。

在 Java 语言中，除了 8 种原始的数据类型（分别为 byte、short、int、long、float、double、char 和 boolean）外，其他的类型都是对象，在方法调用的时候，传递的都是引用。引用从本质上来讲也是按值传递，只不过传递的这个值是对象的引用而已，因此，在方法调用的时候，对形参引用所指对象属性值的修改对实参可见。但是对引用值本身的修改对实参是不可见的。

回到本题中来，在调用 change 方法的时候，change 方法的形参 str 实际上是实参 str（main 方法中的 str）的一个副本，由于 String 是不可变量，因此，无法通过 str 来修改这个字符串的内容，执行语句 str="test ok"的结果是使形参的 str 指向了另外一个常量字符串（可以理解为修改了形参的值，而不是修改了形参所指向对象的值），但是这个修改对实参是不可见的，调用 change 方法结束后对象的 main 方法中 str 的值还是"good"，而 change 方法的形参 ch 也是实参 ch（main 方法中的 ch 值）的一个副本，但是在这个方法中通过形参 ch 修改了实参 ch 所指向对象的值，即数组元素的值，形参 ch 与实参 ch 指向的是同一个对象，因此，通过形参对这个对象值的修改对实参是可见的，所以，当调用 ex.change 方法后，main 方法中 ch 指向的数组的值变为{‘g’，‘b’，‘c’}，因为该方法只是改变了 ch[0]的值而已，所以，程序最终输出为字符串"good and gbc"。所以，选项 B 正确。

所以，本题的答案为 B。

二、填空题

1．答案：13，-12。

分析：本题考察的是 Math 类中 round 方法的使用。

Math 类主要提供了下面 5 个与取整相关的方法：

1）static double ceil(double a)：返回大于等于 a 的最小整数。

2）static double floor(double a)：返回小于等于 a 的最大整数。

3）static double rint(double a)：四舍五入方法，返回与 a 的值最相近的整数，为 double 类型。

4）static long round(double a)：四舍五入方法，返回与 a 的值最相近的长整型数。

5）static int round(float a)：四舍五入方法，返回与 a 的值最相近的整型数。

对于本题而言，round 是一个四舍五入的方法，12.5 的小数部分为 0.5，当对其执行 Math.round()操作时，结果需要四舍五入，所以，结果为 13；-12.5 的小数部分也为 0.5，当对其执行 Math.round()操作时，结果也需要四舍五入，由于-12>-13，因此，结果为-12。

2．答案：false。

分析：本题考察的是字符串知识。

在 Java 语言中，除了 8 种基本的数据类型外，其他的类型都为引用类型，因此，语句 str1==str2 的功能是比较 str1 与 str2 这两个字符串的地址是否相同，显然，str1 存储在常量区，而 str2 中的“world”是在堆空间上申请的另外一块存储空间，因此，二者必然有不同的存储地址。因此，程序的运行结果为 false。

3．答案：浮点型 float、double，char，byte、short、int、long。

分析：本题考察的是 Java 数据类型的知识。

Java 语言中只有 8 种基本数据类型，分别为 byte、short、int、long、float、double、char 和 boolean。在方法调用传参时，这 8 种基本数据类型都是按值传递的，除此之外，所有的数据类型都是按引用传递的。

由以上分析可知，本题的答案为：浮点型 float、double，字符类型 char，布尔类型 boolean，数值类型 byte、short、int、long。

4．答案：String，StringBuffer。

分析：本题考察的是对 Java 字符串的理解。

在 Java 语言中，String 是不可变类，也就是说，String 对象一旦被创建，其值将不能被改变，而 StringBuffer 是可变类，当对象被创建后，仍然可以对其值进行修改。由于 String 是不可变类，因此，适合在需要被共享的场合中使用，而当一个字符串经常需要被修改时，最好使用 StringBuffer 来实现。如果使用 String 来保存一个经常被修改的字符串，在字符串被修改的时候会比 StringBuffer 多了很多附加的操作，同时会生成很多无用的对象，由于这些无用的对象会被垃圾回收器回收，所以，会影响程序的性能。在规模小的项目中这种影响很小，但是在一个规模大的项目中，这会给程序的运行效率带来很大的负面影响。

三、简答题

1．答案：接口（interface）和抽象类（abstract class）是支持抽象类定义的两种机制（注意，该句中前后两个抽象类的意义不一样，前者表示的是一个实体，后者表示的是一个概念）。两者具有很大的相似性，甚至有时候是可以互换的。但同时，两者也存在很大的区别。

具体而言，接口是公开的，里面不能有私有的方法或变量，是用于让别人使用的，而抽象类是可以有私有方法或私有变量的，如果一个类中包含抽象方法，那么这个类就是抽象类。在 Java 语言中，可以通过把类或者类中的某些方法声明为 abstract（abstract 只能用来修饰类或者方法，不能用来修饰属性）来表示一个类是抽象类。接口就是指一个方法的集合，接口

中的所有方法都没有方法体，在 Java 语言中，接口是通过关键字 interface 来实现的。

包含一个或多个抽象方法的类就必须被声明为抽象类，抽象类可以声明方法的存在而不去实现它，被声明为抽象的方法不能包含方法体。在抽象类的子类中，实现方法必须含有相同的或者更低的访问级别（public->protected->private）。抽象类在使用的过程中不能被实例化，但是可以创建一个对象使其指向具体子类的一个实例。抽象类的子类为父类中所有的抽象方法提供具体的实现，否则，它们也是抽象类。接口可以被看作是抽象类的变体，接口中所有的方法都是抽象的，可以通过接口来间接地实现多重继承。接口中的成员变量都是 static final 类型，由于抽象类可以包含部分方法的实现，所以，在一些场合下抽象类比接口存在更多的优势。

接口与抽象类的相同点如下：

1）都不能被实例化。

2）接口的实现类或抽象类的子类都只有实现了接口或抽象类中的方法后才能被实例化。

接口与抽象类的不同点如下：

1）接口只有定义，不能有方法的实现，而抽象类可以有定义与实现，即其方法可以在抽象类中被实现。

2）实现接口的关键字为 implements，继承抽象类的关键字为 extends。一个类可以实现多个接口，但一个类只能继承一个抽象类，因此，使用接口可以间接地达到多重继承的目的。

3）接口强调特定功能的实现，其设计理念是“has-a”关系，而抽象类强调所属关系，其设计理念为“is-a”关系。

4）接口中定义的成员变量默认为 public static final，只能够有静态的不能被修改的数据成员，而且，必须给其赋初值，其所有的成员方法都是 public、abstract 的，而且只能被这两个关键字修饰。而抽象类可以有自己的数据成员变量，也可以有非抽象的成员方法，而且，抽象类中的成员变量默认为 default，当然也可以被定义为 private、protected 和 public，这些成员变量可以在子类中被重新定义，也可以被重新赋值，抽象类中的抽象方法（其前有 abstract 修饰）不能用 private、static、synchronized 和 native 等访问修饰符修饰，同时方法必须以分号结尾，并且不带花括号{}。所以，当功能需要累积时，使用抽象类；不需要累积时，使用　　接口。

5）接口被运用于实现比较常用的功能，便于日后维护或者添加删除方法，而抽象类更倾向于充当公共类的角色，不适用于日后重新对里面的代码进行修改。

2．答案：Java 虚拟机（Java Virtual Machine，JVM，是运行所有 Java 程序的抽象计算机，是 Java 语言的运行环境）允许应用程序并发地运行多个线程。在 Java 语言中，多线程的实现一般有以下三种方法：

1）实现 Runnable 接口，并实现该接口的 run()方法。

以下是主要步骤：

① 自定义类并实现 Runnable 接口，实现 run()方法。

② 创建 Thread 对象，用实现 Runnable 接口的对象作为参数实例化该 Thread 对象。

③ 调用 Thread 的 start()方法。

```
class MyThread implements Runnable
{ //创建线程类
    public void run()
    {
        System.out.println("Thread body");
```

```
        }
    }
    public class Test
    {
        public static void main(String[] args)
        {
            MyThread thread=new MyThread();
            Thread t=new Thread(thread);
            t.start();  //开启线程
        }
    }
```

2）继承 Thread 类，重写 run 方法。

Thread 本质上也是实现了 Runnable 接口的一个实例，它代表一个线程的实例，并且，启动线程的唯一方法就是通过 Thread 类的 start()方法。start()方法是一个 native（本地）方法，它将启动一个新线程，并执行 run()方法（Thread 中提供的 run()方法是一个空方法）。这种方式通过自定义类直接 extends Thread，并重写 run()方法，就可以启动新线程并执行自己定义的 run()方法。需要注意的是，当 start()方法调用后并不是立即执行多线程代码，而是使得该线程变为可运行态（Runnable），什么时候运行多线程代码是由操作系统决定的。

下例给出了 Thread 的使用方法。

```
    class MyThread extends Thread
    { //创建线程类
        public void run()
        {
            System.out.println("Thread body"); //线程的方法体
        }
    }
    public class Test
    {
        public static void main(String[] args)
        {
            MyThread thread=new MyThread();
            thread.start();      //开启线程
        }
    }
```

3）实现 Callable 接口，重写 call()方法。

Callable 对象实际是属于 Executor 框架中的功能类，Callable 接口与 Runnable 接口类似，但是提供了比 Runnable 更强大的功能，主要表现为以下三点：

① Callable 可以在任务结束后提供一个返回值，Runnable 无法提供这个功能。

② Callable 中的 call()方法可以抛出异常，而 Runnable 的 run()方法不能抛出异常。

③ 运行 Callable 可以拿到一个 Future 对象，Future 对象表示异步计算的结果。它提供了检查计算是否完成的方法。由于线程属于异步计算模型，所以无法从其他线程中得到方法的返回值，在这种情况下，就可以使用 Future 来监视目标线程调用 call()方法的情况，当调用 Future 的 get()方法以获取结果时，当前线程就会阻塞，直到 call()方法结束返回结果。

示例代码如下所示：

```
import java.util.concurrent.*;
public class CallableAndFuture
{
    // 创建线程类
    public static class CallableTest implements Callable<String>
    {
        public String call() throws Exception
        {
            return "Hello World!";
        }
    }
    public static void main(String[] args)
    {
        ExecutorService threadPool = Executors.newSingleThreadExecutor();
        // 启动线程
        Future<String> future = threadPool.submit(new CallableTest());
        try
        {
            System.out.println("waiting thread to finish");
            System.out.println(future.get()); // 等待线程结束，并获取返回结果
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

上述程序的输出结果为

```
waiting thread to finish
Hello World!
```

在以上三种方式中，前两种方式线程执行完后都没有返回值，只有最后一种是带返回值的。当需要实现多线程时，一般推荐实现 Runnable 接口的方式，原因如下：首先，Thread 类定义了多种方法可以被派生类使用或重写，但是只有 run 方法是必须被重写的，在 run 方法中实现这个线程的主要功能。这当然是实现 Runnable 接口所需的同样的方法。而且，很多 Java 开发人员认为，一个类仅在它们需要被加强或修改时才会被继承。因此，如果没有必要重写 Thread 类中的其他方法，那么通过继承 Thread 的实现方式与实现 Runnable 接口的效果相同，在这种情况下最好通过实现 Runnable 接口的方式来创建线程。

3．答案：本题考察的是递归知识。

使用递归时，关键问题是要明白递归表达式的含义以及递归的终止条件。

实现代码如下：

```
public class Test
{
    public static long fac(int n)
    {
```

```
            if(n > 1)
                return (n * fac(n - 1));
            else
                return 1;
        }
        public static void main(String args[])
        {
            System.out.println(fac(6));
        }
    }
```

程序的运行结果为

720

4．答案：观察者模式（也被称为发布/订阅模式）提供了避免组件之间紧密耦合的另一种方法，它将观察者和被观察的对象分离开。在该模式中，一个对象通过添加一个方法（该方法允许另一个对象，即观察者注册自己）使本身变得可观察。当可观察的对象更改时，它会将消息发送到已注册的观察者。这些观察者收到消息后所执行的操作与可观察的对象无关，这种模式使得对象可以相互对话，而不必了解原因。Java 语言与 C#语言的事件处理机制就是采用的此种设计模式。

例如，用户界面（同一个数据可以有多种不同的显示方式）可以作为观察者，业务数据是被观察者，当数据有变化后会通知界面，界面收到通知后，会根据自己的显示方式修改界面的显示。面向对象设计的一个原则是：系统中的每个类将重点放在某一个功能上，而不是其他方面。一个对象只做一件事情，并且将它做好。观察者模式在模块之间划定了清晰的界限，提高了应用程序的可维护性和重用性。设计类图如图 1 所示。

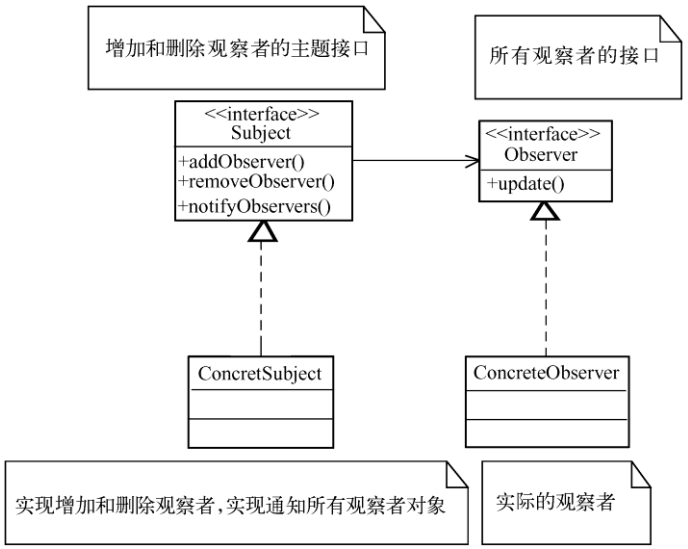


图 1　设计类图

下面给出一个观察者模式的示例代码，代码的主要功能是实现天气预报，同样的温度信息可以有多种不同的展示方式：

```
import java.util.ArrayList;
interface Subject
{
    public void registerObserver(Observer o);
```

```
        public void removeObserver(Observer o);
        public void notifyObservers();
    }

    class Whether implements Subject
    {
        private ArrayList<Observer>observers=new ArrayList<Observer>();
        private float temperature;
        @Override
        public void notifyObservers() {
            for(int i=0;i<this.observers.size();i++)
            {
                this.observers.get(i).update(temperature);
            }
        }
        @Override
        public void registerObserver(Observer o) {
            this.observers.add(o);
        }
        @Override
        public void removeObserver(Observer o) {
            this.observers.remove(o);
        }
        public void whetherChange() {
            this.notifyObservers();
        }
        public float getTemperature(){
            return temperature;
        }
        public void setTemperature(float temperature) {
            this.temperature = temperature;
            notifyObservers();
        }
    }

    interface Observer
    {
        //更新温度
        public void update(float temp);
    }

    class WhetherDisplay1 implements Observer
    {
        private float temprature;
```

```
    public WhetherDisplay1(Subject whether){
        whether.registerObserver(this);
    }
    @Override
    public void update(float temp) {
        this.temprature=temp;
        display();
    }
    public void display(){
        System.out.println("display1****:"+this.temprature);
    }
}

class WhetherDisplay2 implements Observer
{
    private float temprature;
    public WhetherDisplay2(Subject whether)
    {
        whether.registerObserver(this);
    }
    @Override
    public void update(float temp) {
        this.temprature=temp;
        display();
    }

    public void display()
    {
        System.out.println("display2----:"+this.temprature);
    }
}

public class Test
{
    public static void main(String[] args)
    {
        Whether whether=new Whether();
        WhetherDisplay1 d1=new WhetherDisplay1(whether);
        WhetherDisplay2 d2=new WhetherDisplay2(whether);
        whether.setTemperature(27);
        whether.setTemperature(26);
    }
}
```

5．答案：10 亿条记录对应的数据量在 GB 量级，对于普通的计算机来讲，没有这么大

的内存空间供使用，因此，无法一次把这些数据信息全部都读到内存中进行处理，需要对问题进行分解，例如把数据分成 100 份，每一份就是 10MB 量级，基本上放入内存无压力了。

把这 10 亿记录，均分为 100 份，把每份的第一条记录关键字和此记录对应的文件偏移量先扫入内存（类似索引），这里需要磁盘随机 IO100 次。

这样可以马上定位出指定关键字所在的记录块，把相应的记录块拿到内存，二分查找即可。

真题详解 2　某知名社交平台软件工程师笔试题

一、单项选择题

1．答案：D。

分析：本题考察的是进制转换的知识。

本题中，二进制数 11101 对应的十进制数表示为 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 1*2^4 = 29，所以，选项 D 正确。除了人工转换外，在 Java 语言中，也可以用如下方法将一个二进制数转换为十进制数：Integer.valueOf("11101",2) 。

所以，本题的答案为 D。

2．答案：A。

分析：本题考察的是 Java 关键字的知识。

对于选项 A，synchronized（同步的）是 Java 语言的关键字，主要用来给对象和方法或者代码块加锁，当它锁定一个方法或者一个代码块时，同一时刻最多只有一个线程执行这段代码。当两个并发线程访问同一个对象中的这个加锁同步代码块时，同一时间只能有一个线程执行。所以，选项 A 正确。

对于选项 B，serialize 是序列化的意思，所谓对象的序列化指的是把对象转换为字节序列的过程，所谓对象的反序列化指的是把字节序列恢复为对象的过程。通常，对象的序列化主要有以下两种用途：①将对象的字节序列永久地保存到硬盘上，通常存放在一个文件中；②在网络上传送对象的字节序列。在 Java 语言中，序列化通过 Serializable 接口来实现。所以，选项 B 不正确。

对于选项 C，在由 Java 语言编写的程序中，有时候为了提高程序的运行效率，编译器会做一些优化操作，把经常被访问的变量缓存起来，程序在读取这个变量的时候有可能会直接从寄存器中读取这个值，而不会去内存中读取。这样做的一个好处是提高了程序的运行效率，但当遇到多线程编程时，变量的值可能被其他线程改变了，而该缓存的值不会做相应的改变，从而造成应用程序读取的值和实际的变量值不一致。关键字 volatile 正好能够解决这一问题，被关键字 volatile 修饰的变量编译器不会做优化，每次都会从内存中读取。所以，选项 C 不　　正确。

对于选项 D，关键字 static 主要有以下两种作用：第一，为某特定数据类型或对象分配单一的存储空间，而与创建对象的个数无关。第二，希望某个方法或属性与类而不是对象关联在一起，也就是说，在不创建对象的情况下就可以通过类来直接调用方法或使用类的属性。总之，被 static 修饰的属性（方法）是类的属性（方法），不属于任何对象。所以，选项 D 不正确。

所以，本题的答案为 A。

二、不定项选择题

1．答案：C。

分析：本题考察的是 Java 语言构造方法的知识。

对于选项 A，类中的构造方法是可以省略的，当省略的时候，编译器会提供一个默认的构造方法以供使用。因此，选项 A 错误。

对于选项 B，构造方法必须与类名相同，但是方法名也可以与类名相同。如下例所示：

```
public class Test{
    public Test(){
        System.out.println("construct");
    }
    public void Test(){
        System.out.println("call Test");
    }
    public static void main(String[] args) {
        Test a = new Test(); //调用构造方法
        a.Test();   //调用 Test 方法
    }
}
```

程序的运行结果为

```
construct
call Test
```

因此，选项 B 错误。

对于选项 C，当一个对象被 new 的时候必定会调用构造方法。因此，选项 C 正确。

对于选项 D，由于构造方法也可以重载，所以，一个类可以定义多个构造方法。因此，选项 D 错误。

所以，本题的答案为 C。

2．答案：A、B。

分析：本题考察的是 Java 基本语法的知识。

在 Java 语言中，main 方法是程序的入口方法，一个程序要想运行必须要有 main 方法，但是只有满足特定条件的 main 方法才能作为程序的入口方法。对于本题而言：

对于选项 A，由于 Java 语言是纯面向对象语言，所以，所有的属性与方法都必须定义在类里面，而且，main 方法也不例外。因此，选项 A 正确。

对于选项 B，Java 程序可以定义多个 main 方法，但是只有 public static void main(String[] args)方法才是 Java 程序的入口方法，其他 main 方法都不是，并且这个入口方法必须被定义在类名与文件名相同的被 public 修饰的类中，如下例所示（Test.java）：

```
class T{
    public static void main(String[] args) {
        System.out.println("T main");
    }
}
public class Test {
    // 程序入口方法
    public static void main(String[] args) {
        System.out.println("Test main");
    }
}
```

程序的运行结果为

```
Test main
```

从上例可以看出，这个程序中定义了多个 main 方法，但是只有满足特定条件的 main 方法才能作为程序的入口方法。因此，选项 B 正确。

对于选项 C，在 Java 语言中，不管方法体里有几条语句，所有的方法体都必须用大括号{}括起来。因此，选项 C 错误。

对于选项 D，在 Java 语言中，一个文件内部可以有多个类的存在，但只有被 public 修饰的类的名字与文件的名字相同，其他类的名字可以根据需求随意起名字。因此，选项 D 错误。

所以，本题的答案为 A、B。

3．答案：C。

分析：本题考察的是 Java 关键字的知识。

对于选项 A，关键字 private 是一个作用域修饰符，被关键字 private 修饰过的变量或方法只有当前类或对象具有访问权限。所以，选项 A 不正确。

对于选项 B，在 Java 语言中，可以通过把类或者类中的某些方法声明为 abstract 来表示一个类是抽象类。所以，选项 B 不正确。

对于选项 C，被 final 修饰的变量为常量，当一个方法被声明为 final 时，该方法不允许任何子类重写，当一个类被声明为 final 时，此类不能被继承，所有方法都不能被重写。所以，选项 C 正确。

对于选项 D，关键字 static 主要有两种作用：第一，为某特定数据类型或对象分配单一的存储空间，而与创建对象的个数无关；第二，希望某个方法或属性与类而不是对象关联在一起，也就是说，在不创建对象的情况下就可以通过类来直接调用方法或使用类的属性。即被 static 修饰的属性（方法）是类的属性（方法），不属于任何对象。所以，选项 D 不正确。

所以，本题的答案为 C。

4．答案：A、B、C。

分析：本题考察的是抽象类的知识。

在 Java 语言中，关键字 abstract 主要用来定义抽象类或抽象方法。

对于选项 A，在 Java 语言中，关键字 abstract 只能修饰类或方法，代表抽象方法或抽象类。所以，选项 A 正确。

对于选项 B，在 Java 语言中，关键字 final 可以用来修饰类、方法和变量（包括成员变量和局部变量）。具体而言，final 具有以下性质：①当用 final 修饰一个类时，表明这个类不能被继承；②当 final 修饰方法时，这个方法不能被子类重写；③当 final 修饰变量时，如果是基本数据类型的变量，则其数值一旦被初始化之后便不能更改，如果是引用类型的变量，则在对其初始化之后便不能再让其指向另一个对象。所以，被 final 修饰的方法是不能被继承的，因此，该方法不能为 abstract。所以，选项 B 正确。

对于选项 C，由于抽象类中存在没有方法体的方法，因此，不能被实例化。所以，选项 C 正确。

对于选项 D，子类可以实现超类所有的方法（这个子类能被实例化），子类也可以是一个抽象类，此时这个子类可以实现超类的 abstract 方法，也可以不实现。所以，选项 D 错误。

所以，本题的答案为 A、B、C。

5．答案：C。

分析：本题考察的是标识符的知识。

在 Java 语言中，变量名、方法名和数组名统称为标识符，Java 语言规定标识符只能由字母（a～z，A～Z）、数字（0～9）、下划线（_）和$组成，并且标识符的第一个字符必须是字母、下划线或$。而且，标识符也不能包含空白字符（换行符、空格和制表符）。此外，Java 语言的关键字也不能作为标识符来使用。

以上这四个选项都符合变量的命名规则，但是，选项 C 中的 void 是 Java 语言的关键字，因此，它不能被用作标识符使用。所以，选项 C 不正确。

所以，本题的答案为 C。

6．答案：B。

分析：本题考察的是 Java 语言知识。

对于选项 A，只要类中定义了构造方法（不管有没有参数），JVM 都不会提供默认构造方法了。因此，选项 A 错误。

对于选项 B，变量的作用域指的是可以使用此变量的名称来引用它的程序区域，变量声明在程序中的位置决定了该变量的作用域。局部变量在方法或方法的一个代码块中声明，它的作用域为它所在的代码块（代码块是整个方法或方法中的某块代码，即以括号{}包括的代码）。 所以，局部变量仅在定义它的方法或块内可见。因此，选项 B 正确。

对于选项 C，在使用其他类的方法的时候，需要用“类名.方法”来引用，不能直接通过方法名字引用。因此，选项 C 错误。

对于选项 D，只有被 static 修饰的变量才是类的成员变量，但是当类中的变量被 private 修饰时，其他类是无法直接使用的。因此，选项 D 错误。

所以，本题的答案为 B。

7．答案：B。

分析：本题考察的是前置自增运算符++和后置自增运算符++的知识。

在编程的时候，经常会用到变量的自增或自减操作，尤其在循环语句中用得最多。以自增为例，有两种自增方式：前置与后置，即++i 和 i++，它们的不同点在于后置的 i++是在程序执行完毕后自增，而前置的++i 是在程序开始执行前进行自增。

对于本题而言，整型变量 i 被初始化为 6，因此，第一个输出为 6，第二个输出 i++，因为这是后置++操作，因此，输出结果为 6，输出后 i 的值变为 7，故最后一个输出操作的值为 7。所以，选项 B 正确。

所以，本题的答案为 B。

8．答案：A。

分析：本题考察的是 super 关键字的知识。

在 Java 语言中，关键字 this 指的是对当前对象的引用，关键字 super 指的是当前对象里面的父对象的引用，当引用当前对象的某个方法或某个成员时，通常会使用 this，而通过 super 可以调用父类的构造方法、父类的方法和属性。如下例所示：

```
class Base
{
    public int status=0;
    Base(int status)
    {
        this.status=status;
    }
    public void print()
    {
        System.out.println("base");
    }
}
class Sub extends Base
{
    public int status;
    Sub(int status)
```

```
        {
            super(status-1);
            this.status=status;
        }
        public void printSub()
        {
            System.out.println("sub");
            System.out.println("status="+status);
        }
        public void printBase()
        {
            super.print();
            System.out.println("status="+super.status);
        }
    }
    public class Test
    {
        public static void main(String args[])
        {
            Sub s=new Sub(2);
            s.printBase();
            s.printSub();
        }
    }
```

程序的运行结果为

```
base
status=1
sub
status=2
```

通过以上分析可知，选项 A 正确，选项 B 和选项 C 错误，对于选项 D，super 只能表示父类的引用，不能表示父类的父类。因此，选项 D 错误。

所以，本题的答案为 A。

9．答案：A、B、D。

分析：本题考察的是 Java 语言中字符串相关知识。

在 Java 语言中，String 是不可变类，也就是说，String 对象一旦被创建，字符串的内容将不能被改变，而 StringBuffer 是可变类，当对象被创建后，仍然可以对字符串的内容进行修改。由于 String 是不可变类，因此，它适合在需要被共享的场合中使用，而当一个字符串经常需要被修改时，最好使用 StringBuffer 来实现。如果使用 String 来保存一个经常被修改的字符串，在字符串被修改时，会比 StringBuffer 多很多附加的操作，同时生成很多无用的对象，由于这些无用的对象会被垃圾回收器回收，所以会影响程序的性能。在规模小的项目中这个影响很小，但是在一个规模大的项目中，这会对程序的运行效率带来很大的影响。

String 字符串修改实现的原理如下：当使用 String 类型来对字符串进行修改时，其实现方法是首先创建一个 StringBuffer，然后调用 StringBuffer 的 append 方法，最后调用 StringBuffer 的 toString 方法把结果返回。举例如下：

String s=“Hello”;

s+=“World”;

以上代码与下述代码等价：

String s=“Hello”;

StringBuffer sb=new StringBuffer(s);

s.append(“World”);

s=sb.toString();

由此可以看出，上述过程比使用 StringBuffer 多了一些附加的操作，同时也生成了一些临时的对象，这样会导致程序的执行效率降低。

StringBuilder 也是可以被修改的字符串，它与 StringBuffer 类似，都是字符串缓冲区，但 StringBuilder 不是线程安全的，如果只是在单线程中使用字符串缓冲区，那么 StringBuilder 的效率会更高些。因此，在只有单线程访问的时候，可以使用 StringBuilder，当有多个线程访问时，最好使用线程安全的 StringBuffer。因为 StringBuffer 必要时可以对这些方法进行同步，所以，任意特定实例上的所有操作就好像是以串行顺序发生的，该顺序与所涉及的每个线程进行的方法调用顺序一致。

在执行效率方面，StringBuilder 最高，StringBuffer 次之，String 最低，鉴于这一情况，一般而言，如果要操作的数据量比较小，可以使用 String 类，如果是在单线程下操作大量数据，优先使用 StringBuilder 类，如果是在多线程下操作大量数据，优先考虑 StringBuffer 类。

从以上分析可知，StringBuilder 不是线程安全的。所以，选项 A、选项 B 与选项 D 正确。

所以，本题的答案为 A、B、D。

10．答案：B、C、D。

分析：本题考察的是 Java 语言中基本类型知识。

Java 语言一共提供了 8 种原始的数据类型（byte、short、int、long、float、double、char 和 boolean），这些数据类型不是对象，而是 Java 语言中不同于类的特殊类型，这些基本类型的数据变量在声明之后就会立刻在栈上分配内存空间。此外，Java 语言还提供了这些原始数据类型的包装类（字符类型 Character，布尔类型 Boolean，数值类型 Byte、Short、Integer、Long、Float、Double）。

本题中，Byte、Float 是包装类类型，String 是存储字符串的类，只有 int 是基本数据类型。所以，选项 B、选项 C 与选项 D 正确。

所以，本题的答案为 B、C、D。

11．答案：D。

分析：本题考察的是线程相关方法的知识。

对于选项 A，wait()方法是一种使线程暂停执行的方法，例如，当线程交互时，如果线程对一个同步对象发出了一个 wait()调用请求，那么该线程会暂停执行，被调对象进入等待状态，直到被唤醒（通常使用 notify 方法唤醒）或等待时间超时。所以，选项 A 错误。

对于选项 B，sleep()方法的作用是使当前运行的线程休眠指定的时间。所以，选项 B 错误。

对于选项 C，可以使用 stop()方法来终止线程的执行。当使用 Thread.stop()方法来终止线程时，它会释放已经锁定的所有的监视资源。如果当前任何一个受这些监视资源保护的对象处于一个不一致的状态，其他的线程将会看到这个不一致的状态，这可能会导致程序执行的不确定性，并且这种问题很难被定位。因此，不推荐使用。所以，选项 C 错误。

对于选项 D，suspend()方法就是将一个线程挂起（暂停），并且不会自动恢复，必须通过调用对应的 resume()方法，才能使得线程重新进入可执行状态。所以，选项 D 正确。

所以，本题的答案为 D。

12．答案：B。

分析：本题考察的是 Java 语言中异常处理的知识。

在 Java 语言的异常处理中，finally 语句块的作用就是为了保证无论出现什么情况，finally 语句块里的代码一定会被执行。由于当程序执行到 return 语句的时候就意味着结束对当前方法的调用并跳出这个方法体，因此，任何语句要执行都只能在 return 语句前执行（除非碰到 exit 方法），所以，finally 块里的代码也是在 return 前执行的。此外，如果 try-finally 或者 catch-finally 中都有 return 语句，则 finally 块中的 return 语句将会覆盖别处的 return 语句，最终返回到调用者那里的是 finally 中 return 的值。

对于本题而言，在调用 testException 方法时不会抛出异常，虽然 testException 方法体内调用 return 返回这个方法，但是 Java 虚拟机要保证 finally 块的代码必须执行，因此，在调用 testException 方法的时候会输出 finally，接着方法调用结束后，在 main 方法中会输出 finished。因此，选项 B 正确。

所以，本题的答案为 B。

13．答案：D。

分析：本题考察的是 Java 垃圾回收知识。

在 Java 语言中，GC（Garbage Collection，垃圾回收）是一个非常重要的概念，它的主要作用是回收程序中不再使用的内存。在使用 C/C++语言进行程序开发的时候，开发人员必须非常仔细地管理好内存的分配与释放，如果忘记或者错误地释放内存往往会导致程序运行不正确甚至是程序的崩溃。为了减轻开发人员的工作，同时增加系统的安全性与稳定性，Java 语言提供了垃圾回收器来自动检测对象的作用域，实现自动地将不再被使用的存储空间释放掉。

在 Java 语言中，释放掉占据的内存空间是由 GC 完成的，程序员无法直接强制释放存储空间，当一个对象不被使用的时候，GC 会将该对象标记为垃圾，会在后面一个不确定的时间内回收垃圾（程序员无法控制这个时间）。

给对象引用赋值为 null，并且该对象无其他引用，GC 会标记该对象为垃圾，会在后面一个不确定的时间内回收垃圾。所谓不确定是指什么时间回收，程序员无法控制。

本题中，对于选项 A，开发人员可以通过调用 System.gc()方法来通知垃圾回收器运行，但是 JVM 也并不能保证垃圾回收器马上就会运行。因此，选项 A 错误。

对于选项 B，Java 语言没有提供 free（释放）方法。因此，选项 B 错误。

对于选项 C，当把对象的引用设置为 null 时，GC 会标记该对象为垃圾，会在后面一个不确定的时间内回收垃圾。因此，选项 C 错误。

对于选项 D，程序员无法明确强制垃圾回收器运行。因此，选项 D 正确。

所以，本题的答案为 D。

14．答案：A、B、C。

分析：本题考察的是 Spring 框架知识。

Spring 是一个 J2EE 的框架，这个框架提供了对轻量级的 IoC（Inverse of Control，控制反转，有时候也被叫作依赖注入）良好的支持，同时也提供了对 AOP（Aspect Oriented Programming，面向切面编程）技术非常好的封装。相比其他框架，Spring 框架的设计更加模块化，框架内的每个模块都能完成特定的工作，而且各个模块都可以独立地运行，不会相互牵制。因此，在使用 Spring 框架的时候，开发人员可以使用整个框架，也可以只使用框架内的一部分模块，例如可以只使用 Spring AOP 模块来实现日志管理功能，而不需要使用其他 模块。

具体而言，Spring 框架主要由 7 个模块组成，它们分别是 Spring AOP、Spring ORM、

Spring DAO、Spring Web、Spring Context、Spring Web MVC 和 Spring Core 等。具体如图 2 所示。

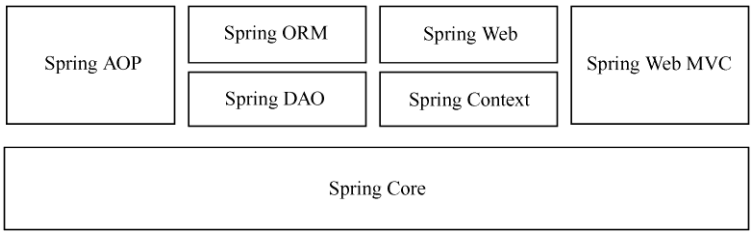


图 2　Spring 框架

各个模块的作用见表 5。

表 5　各个模块的作用

模　　块	描　　述
Spring AOP	采用了面向切面编程的思想，使 Spring 框架管理的对象支持 AOP，同时这个模块也提供了事务管理，可以不依赖具体的 EJB 组件，就可以将事务管理集成到应用程序中
Spring ORM	提供了对现有的 ORM 框架的支持，例如 Hibernate、JDO 等
Spring DAO	提供了对 DAO（Data Access Object，数据访问对象）模式和 JDBC 的支持。DAO 可以实现将业务逻辑与数据库访问的代码分离，从而降低代码的耦合度。通过对 JDBC 的抽象，简化了开发工作，同时简化了对异常的处理（可以很好地处理不同数据库厂商抛出的异常）
Spring Web	提供了 Servlet 监听器的 Context 和 Web 应用的上下文。同时还集成了一些现有的 Web 框架，例如 Struts
Spring Context	扩展核心容器，提供了 Spring 上下文环境，给开发人员提供了很多非常有用的服务，例如国际化、Email 和 JNDI 访问等
Spring Web MVC	提供了一个构建 Web 应用程序的 MVC 的实现
Spring Core	Spring 框架的核心容器，它提供了 Spring 框架的基本功能。这个模块中最主要的一个组件为 BeanFactory，它使用工厂模式来创建所需的对象。同时 BeanFactory 使用 IOC 思想，通过读取 XML 文件的方式来实例化对象，可以说 BeanFactory 提供了组件生命周期的管理，组件的创建、装配以及销毁等功能

通过以上分析可以看出，选项 A、选项 B 与选项 C 的描述都是正确的。对于选项 D，Spring 并没有提供日志系统，需要根据需求使用 AOP 的方式，借助 Spring 与日志系统 log4j 实现自己的日志功能。因此，选项 D 错误。

所以，本题的答案为 A、B、C。

15．答案：A、B。

分析：本题考察的是堆的知识。

堆是一种特殊的树形结构，有大顶堆和小顶堆两种。大顶堆（小顶堆）的特点是根结点的值最大（最小），且根结点的子树也为一个大顶堆（小顶堆）。

对于选项 A，完全二叉树是指除最后一层外，每一层上的结点数均达到最大值；在使用的时候堆是采用数组来存储的，因此，它满足完全二叉树的特点。所以，选项 A 正确。

对于选项 B，平衡二叉树（Balanced Binary Tree）又被称为 AVL 树（有别于 AVL 算法），具有以下性质：它是一棵空树或它的左右两个子树的高度差的绝对值不超过 1，并且左右两棵子树都是一棵平衡二叉树。由于完全二叉树一定满足平衡二叉树的性质。所以，选项 B

正确。

对于选项 C，排序二叉树有如下性质：

1）若左子树不为空，则左子树上所有结点的值均小于它的根结点的值。

2）若右子树不为空，则右子树上所有结点的值均大于或等于它的根结点的值。

3）左、右子树也分别为二叉排序树。

4）没有键值相等的结点。

显然，堆不满足这个性质。所以，选项 C 是错误的。

对于选项 D，满二叉树是指树中除最后一层无任何子结点外，每一层上的所有结点都有两个子结点的二叉树。满二叉树中结点的个数为 1、3、7 等特殊的数字，而堆中的结点可以是任意的，因此，不能保证堆是个满二叉树。所以，选项 D 不正确。

所以，本题的答案为 A、B。

16．答案：A、C、D。

分析：本题考察的是依赖注入的知识。

IoC（Inverse of Control，控制反转）有时候也被叫作依赖注入，是一种降低对象之间耦合关系的设计思想。一般而言，在分层体系结构中，都是上层调用下层的接口，上层依赖于下层的执行，即调用者依赖于被调用者。而通过 IoC 方式，使得上层不再依赖于下层的接口，即通过采用一定的机制来选择不同的下层实现，完成控制反转，使得由调用者来决定被调用者。IoC 通过注入一个实例化的对象来达到解耦和的目的。使用这种方法后，对象不会被显式地调用，而是根据需求通过 IoC 容器（例如 Spring）来提供。

采用 IoC 机制能够提高系统的可扩展性，如果对象之间通过显式的调用进行交互，那么会导致调用者与被调用者存在着非常紧密的联系，其中一方的改动将会导致程序出现很大的改动，例如，要为一家卖茶的商店提供一套管理系统，在这家商店刚开业的时候只卖绿茶（Green Tea），随着规模的扩大或者根据具体销售量，未来可能会随时改变茶的类型，例如红茶（Black Tea）等，传统的实现方法会针对茶抽象化一个基类，绿茶类只需要继承自该基类即可。如图 3 所示。

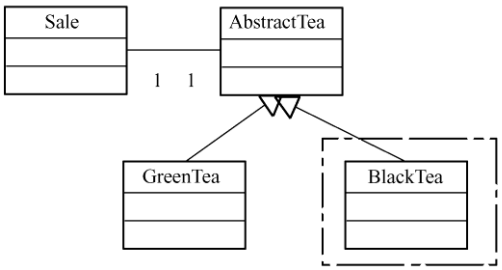


图 3　茶的继承

采用该实现方法后，在需要使用 Green Tea 的时候，只需要执行以下代码即可：Abstract Tea t = new Green Tea()，当然，这种方法是可以满足当前设计要求的。但是该方法的可扩展性不好，存在着不恰当的地方，例如，当商家发现绿茶的销售并不好，决定开始销售红茶（Black Tea）时，那么只需要实现一个 Black Tea 类，并且让这个类继承自 Abstract Tea 即可。但是，在系统中所有用到 Abstract Tea t = new Green Tea()的地方，都需要被改为 Abstract Tea t = new Black Tea()，而这种创建对象实例的方法往往会导致程序的改动量非常大。

那么怎样才能增强系统的可扩展性呢？此时可以使用设计模式中的工厂模式将创建对象的行为包装起来，实现方法如图 4 所示。

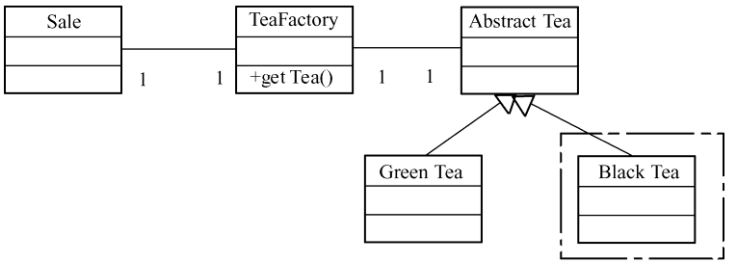


图 4　工厂模式

通过以上方法，可以把创建对象的过程委托给 TeaFatory 来完成，在需要使用 Tea 对象的时候，只需要调用 Factory 类的 get Tea 方法即可，具体创建对象的逻辑在 TeaFactory 中来实现，那么当商家需要把绿茶替换为红茶的时候，系统中只需要改动 TeaFactory 中创建对象的逻辑即可。当采用了工厂模式后，只需要在一个地方做改动就可以满足要求，从而增强了系统的可扩展性。

虽然说采用工厂设计模式后增强了系统的可扩展性，但是从本质上来讲，工厂模式只不过是把程序中会变动的逻辑移动到工厂类里面了，当系统中的类较多的时候，在系统扩展时需要经常改动工厂类中的代码。而采用 IoC 设计思想后，程序将会有更好的可扩展性，下面主要介绍 Spring 框架在采用 IoC 后的实现方法，如图 5 所示。

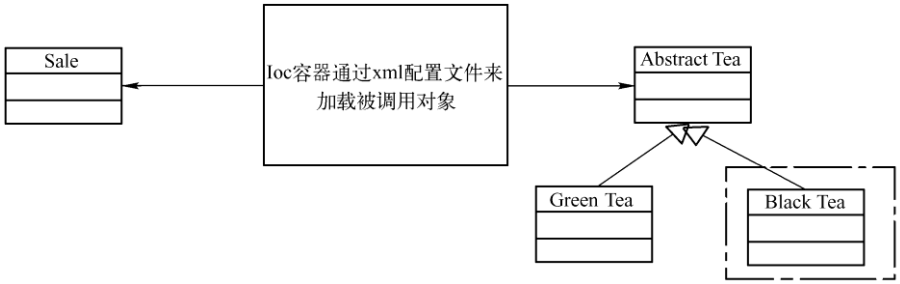


图 5　IoC 设计思想

Spring 容器将会根据配置文件来创建调用者对象（Sale），同时将被调用的对象（AbstractTea 的子类）的实例化对象通过构造方法或 set 方法的形式注入到调用者对象中。

首先，创建名为 SpringConfig.xml 的文件。

```
<beans>
<bean    id=”sale” class=”Sale” singleton=”false”>
<constrctor-arg>
<ref bean=”tea”/>
</constrctor-arg>
</bean>
<bean    id=”tea” class=”BlueTea” singleton=”false”>
</beans>
```

在实现 Sale 类的时候，需要按照如下方式实现：

```
class Sale{
private AbstractTea t;
public Sale(AbstractTea t){
      this.t=t;
}
//其他方法就可以使用 t 了
}
```

当 Spring 容器在创建 Sale 对象的时候，根据配置文件 SpringConfig.xml 就会创建一个 Blue Tea 的对象，作为 Sale 构造方法的参数。当需要把 Blue Tea 改为 Black Tea 时，只需要修改上述配置文件即可，而不需要修改代码。

在需要 Sale 的时候，可以通过如下方式来创建 Sale 对象：

ApplicationContext xtx=new FileSystemXmlApplicationContext(“SpringConfig.xml”);

Sale s=(Sale)ctx.getBean(“sale”);

上例中，Spring 采用 IoC 的方式来实现将实例化的对象注入到开发人员自定义的对象中，具有较强的可扩展性。

具体而言，IoC 主要有以下几个方面的优点：

1）通过 IoC 容器，开发人员不需要关注对象是如何被创建的，同时，增加新类也非常方便，只需要修改配置文件即可实现对象的热插拔。

2）IoC 容器可以通过配置文件来确定需要注入的实例化对象，因此，非常便于进行单元测试。

尽管如此，IoC 也有自身的缺点，具体表现为如下两点：

1）对象是通过反射机制实例化出来的，因此，会对系统的性能有一定的影响。

2）创建对象的流程变得比较复杂。

通过以上分析可知，只有选项 B 描述错误，注入降低了组件之间的耦合性，而不是使组件之间相互依赖。

所以，本题的答案为 A、C、D。

17．答案：A、C、D。

分析：本题考察的是 Hashtable 和 HashMap 知识。

HashMap 和 Hashtable 通过对象来进行索引，用来索引的对象叫作 key，其对应的对象叫作 value。两者具有许多相似之处，但也有很多不同之处。以下重点介绍两者的不同之处，具体而言，体现在以下几个方面：

1）它们都实现了 Map 接口，HashMap 允许空（null）键值（key）（但需要注意的是，最多只允许一条记录的键为 null，不允许多条记录的值为 null），而 Hashtable 不允许。

2）HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue 和 containsKey。因为 contains 方法容易引起误解。Hashtable 继承自 Dictionary 类，而 HashMap 继承自 AbstractMap 类。

3）Hashtable 的方法是线程安全的，而 HashMap 不是线程安全的。当多个线程访问 Hashtable 时，不需要开发人员对它进行同步，而对于 HashMap，开发人员必须提供额外的同步机制。所以，效率上 HashMap 可能高于 Hashtable。

4）“快速失败”也就是 fail-fast，它是 Java 集合的一种错误检测机制。当多个线程对集合进行结构上的改变的操作时，就有可能会产生 fail-fast 事件。例如，假设存在两个线程，它们分别是线程 1 与线程 2，当线程 1 通过 Iterator（迭代器）在遍历集合 A 中的元素的时候，如果线程 2 修改了集合 A 的结构（删除或增加新的元素），那么，这个时候程序就会抛出 ConcurrentModificationException 异常，从而产生 fail-fast 事件。

由于 Hashtable 是线程安全的，因此，没有采用快速失败机制，HashMap 是非线程安全的，因此，迭代 HashMap 采用了快速失败机制。

从以上分析可知，选项 A、选项 C 与选项 D 的描述都是正确的，只有选项 B 的描述不正确，因为 Hashtable 不允许键值为 null。

所以，本题的答案为 A、C、D。

18．答案：A。

分析：本题考察的是 List 迭代器 Iterator 的知识。

Iterator 支持从源集合中安全地删除对象，删除的方法为在 Iterator 上调用 remove()方法 。这样做的好处是可以避免 ConcurrentModifiedException 异常发生，当打开 Iterator 迭代集合时，同时又在对集合进行修改。有些集合不允许在迭代时删除或添加元素，但是调用 Iterator 的 remove() 方法是个安全的做法。

remove()方法的作用为从迭代器指向的集合中移除迭代器返回的最后一个元素（可选操作），每次调用 next()方法只能调用一次此方法。如果在进行迭代时，用调用此方法之外的其他方式修改了该迭代器所指向的集合，那么迭代器的行为是不明确的。因此，选项 A 正确。

所以，本题的答案为 A。

19．答案：A、B、C。

分析：本题考察的是算法基础知识。

算法指的是解题方案的准确而完整的描述，是一系列解决问题的清晰指令，算法代表着用系统的方法描述解决问题的策略机制。其主要功能是对输入结果特定的运算产生期望的输出，所以，输入数据、处理数据及输出结果都属于算法结构，显然，存储数据的功能的并不包括在内。所以，选项 A、选项 B 和选项 C 正确。

所以，本题的答案为 A、B、C。

20．答案：D。

分析：本题考察的是对二叉树各种遍历知识的理解。

要想找出本题的正确答案，首先要弄明白二叉树的几种遍历方式的原理。

后序遍历：首先遍历左子树，然后遍历右子树，最后访问根结点，在遍历左、右子树时，仍然先遍历左子树，然后遍历右子树，最后遍历根结点。

中序遍历：首先遍历左子树，然后访问根结点，最后遍历右子树。在遍历左、右子树时，仍然先遍历左子树，再访问根结点，最后遍历右子树。

前序遍历：首先访问根结点然后遍历左子树，最后遍历右子树。在遍历左、右子树时，仍然先访问根结点，然后遍历左子树，最后遍历右子树。

本题中，由后序遍历序列为 dabec 可知，根结点为 c，再通过中序遍历序列可知，右子树为空。接着由 dabe 可知，其根结点为 e，所以，在中序遍历序列 deba 中，左子树为 d，右子树为 ba。后序遍历序列为 ab，中序遍历序列为 ba，可以推断出 b 为根结点（相对于 a 而言），a 为右子树。所以，可以得到前序遍历序列为 cedba。所以，选项 B 正确。

所以，本题的答案为 B。

21．答案：D。

分析：本题考察的是算法的空间复杂度知识。

空间复杂度是对一个算法在运行过程中临时占用存储空间大小的量度。算法在运行时占用的临时空间与算法的长度、程序的指令条数以及程序所占用的存储空间都没有直接关系。显然，与此符合的描述只有选项 D。所以，选项 D 正确。

所以，本题的答案为 D。

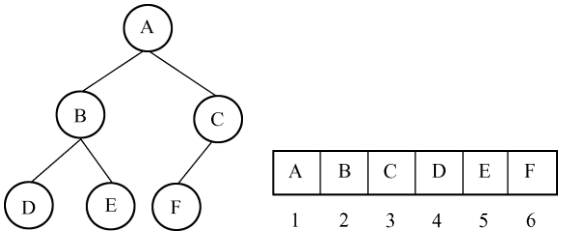
22．答案：D。

分析：本题考察的是二叉树的知识。

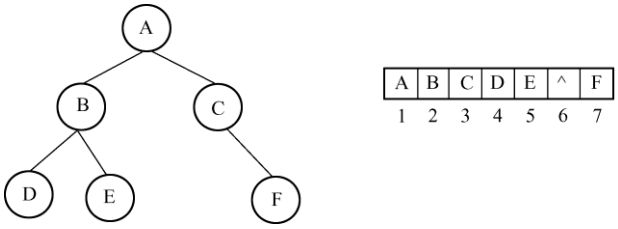
二叉树是非线性数据结构，即每个数据结点至多只有一个前驱，但可以有多个后继，可以使用顺序存储和链式存储两种结构来存储。以下将分别对这两种存储结构进行介绍。

（1）顺序存储结构

二叉树的顺序存储指的是用元素在数组中的下标表示一个结点与其孩子和父结点的关系。这种结构特别适用于近似满二叉树。这种方法的缺点是可能会有大量空间的浪费，在最坏的情况下，一个深度为 k 且只有 k 个结点的右单支树需要 2^k-1 个结点存储空间。如图 6 所示分别给出了完全二叉树和非完全二叉树的存储示意图。



a)



b)

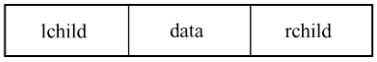
图 6　二叉树的顺序存储方式

a) 完全二叉树的存储方式　b) 非完全二叉树的存储方式

（2）链式存储结构

二叉树的链式存储结构是指用链表来表示一棵二叉树。

每个结点有一个数据域，两个指针域分别指向左孩子和右孩子。其结点结构为



如图 7 所示给出了一个二叉树的链表存储方式。

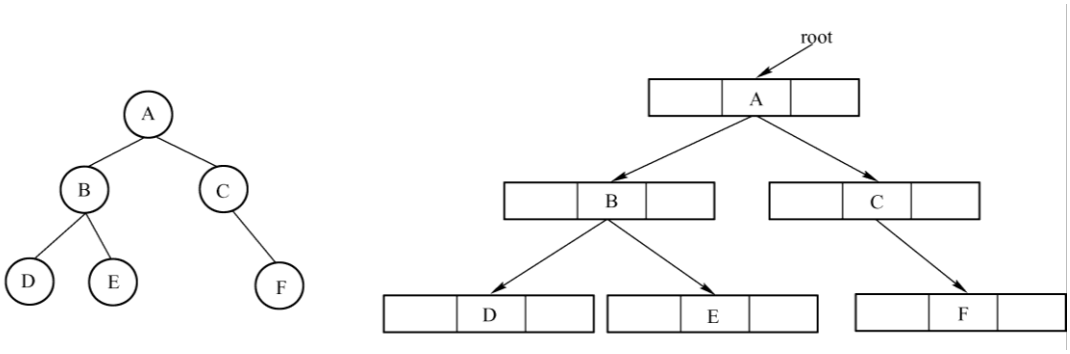


图 7　二叉树的链表存储方式

通过上面的分析可知，选项 D 正确。

所以，本题的答案为 D。

23．答案：A。

分析：本题考察的是二叉树的知识。

二叉树具有以下性质：一棵非空二叉树的第 i 层上最多有 2^(i-1)个结点（i≥1）。所以，本题中，第 4 层的结点数最多为 2^3=8。所以，选项 A 正确。

所以，本题的答案为 A。

24．答案：C。

分析：本题考察的是对快速排序算法的理解。

快速排序是一种非常高效的排序算法，它采用“分而治之”的思想，把大的拆分为小的，小的再拆分为更小的。其原理如下：对于一组给定的记录，通过一趟排序后，将原序列分为

两部分，其中前一部分的所有记录均比后一部分的所有记录小，然后再依次对前后两部分的记录进行快速排序，递归该过程，直到序列中的所有记录均有序为止。

一趟快速排序的算法步骤如下：

1）设置两个变量 i、j，排序开始的时候：i=0，j=N-1。

2）以第一个数组元素作为关键数据，赋值给 key，即 key=A[0]。

3）从 j 开始向前搜索，即逆向遍历数组(j--)，找到第一个小于 key 的值 A[j]，将 A[j]和 A[i]互换。

4）从 i 开始向后搜索，即正向遍历数组(i++)，找到第一个大于 key 的 A[i]，将 A[i]和 A[j]互换。

5）重复第 3)、4）步，直到 i=j。

对于本题而言，把 5 作为基准，执行过程如下：首先，从数组最右边开始遍历，由于 8>5，因此，不需要移动 8 的位置，接着向后遍历数组元素 3，由于 3<5，因此，交换这两个数组元素的值，此时数组中的元素为（3，2，6，5，8），接着正向遍历数组，下一个遍历的值为 2，由于 2<5，因此，不需要移动 2 的位置，接着遍历 6，由于 6>5，因此，需要交换 5 和 6 的位置，此时数组的值为（3，2，5，6，8），此时已经遍历了数组中所有的值，一趟快速排序结束。所以，选项 C 正确。

所以，本题的答案为 C。

25．答案：C。

分析：本题考察的是数据库的相关知识。

对于选项 A，Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲地使用对象编程思维来操纵数据库。所以，选项 A 不正确。

对于选项 B，Java 应用程序可以通过 JDBC 或 Hibernate 对数据库系统进行访问。虽然 JDBC 和 Hibernate 都提供了事务控制的接口，但这些接口只是把事务控制相关的命令发送给数据库系统，由数据库系统来控制事务的隔离级别。所以，选项 B 不正确。

对于选项 C，数据库系统是为适应数据处理的需要而发展起来的一种较为理想的数据处理系统，也是一个为实际可运行的存储、维护和应用系统提供数据的软件系统，是存储介质、处理对象和管理系统的集合体。在数据库操作中，为了保证在并发情况下数据读写的正确性，提出了事务隔离级别。在标准 SQL 规范中，定义了 4 个事务隔离级别，分别为未授权读取，也称为读未提交（read uncommitted）；授权读取，也称为读提交（read committed）；可重复读取（repeatable read）；序列化（serializable）。因此，事务隔离级别是由数据库系统实现的。所以，选项 C 正确。

对于选项 D，JDBC 驱动程序是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问。所以，选项 D 不正确。

所以，本题的答案为 C。

26．答案：B。

分析：本题考察的是对双向链表知识的理解。

对于选项 A，最后一个操作 p->right->left=s，此时，p->right 指向 s，p->right->left=s 等价于 s->left=s，显然是错误的。因此，选项 A 错误。

对于选项 B，描述正确。因此，选项 B 正确。

对于选项 C，如果先执行语句 p->right=s，由于没有记录结点 p 的后继结点，因此，后面的操作将无法找到结点 p 的后继结点。因此，选项 C 错误。

对于选项 D，p->right->left=s 也等价于 s->left=s，显然是错误的。因此，选项 D 错误。

所以，本题的答案为 B。

27．答案：C。

分析：本题考察对常见排序算法原理的理解。

对于选项 A，归并排序是利用递归与分治技术将数据序列划分成为越来越小的子序列，再对子序列排序，最后再用递归方法将排好序的子序列合并成为越来越大的有序序列。

对于选项 B，希尔排序也称为“缩小增量排序”，其基本原理如下：首先将待排序的数组元素分成多个子序列，使得每个子序列的元素个数相对较少，然后对各个子序列分别进行直接插入排序，待整个待排序序列“基本有序后”，最后再对所有元素进行一次直接插入排序。

对于选项 C，插入排序是指对于给定的一组记录，初始时假设第一个记录自成一个有序序列，其余的记录为无序序列。接着从第二个记录开始，按照记录的大小依次将当前处理的记录插入到其之前的有序序列中，直至最后一个记录插入到有序序列中为止。

对于选项 D，选择排序是一种简单直观的排序算法，它的基本原理如下：对于给定的一组记录，经过第一轮比较后得到最小的记录，然后将该记录与第一个记录的位置进行交换；接着对不包括第一个记录以外的其他记录进行第二轮比较，得到最小的记录并与第二个记录进行位置交换；重复该过程，直到进行比较的记录只有一个时为止。

从以上分析可以看出，题目所描述的排序方法为插入排序。所以，选项 C 正确。

所以，本题的答案为 C。

28．答案：A、B。

分析：本题考察的是操作系统的知识。

操作系统（Operating System，OS）是一个庞大的管理控制程序，管理计算机硬件与软件资源，同时，它也是计算机系统的内核与基石，大致包括 5 个方面的管理功能：进程与处理机管理、作业管理、存储管理、设备管理及文件管理。

通过上面的分析可知，选项 A 与选项 B 正确。

所以，本题的答案为 A、B。

29．答案：A、B、D。

分析：本题考察的是中断的基础知识。

中断源一般可分为两类：强迫性中断和自愿性中断。强迫性中断由随机事件引起而非程序员事先安排，它包括输入/输出中断、硬件故障中断、时钟中断、控制台中断和程序性中断。设备出错、执行 print 语句属于其中的输入/输出中断；断电属于硬件故障中断。时间片到时属于自愿性中断。

所以，选项 A、选项 B 和选项 D 正确。

所以，本题的答案为 A、B、D。

30．答案：A。

分析：本题考察的是操作系统基础知识。

进程的基本调度状态有运行、就绪和阻塞。进程调度程序从处于就绪状态的进程中选择一个投入运行。运行进程因等待某一事件而进入阻塞状态，因时间片到达而回到就绪状态。处于阻塞状态的进程当所等待的事件发生时，便进入就绪状态。

本题中，就绪队列是等待 CPU 时间的队列，其中存放着等待执行的任务。进程调度是从就绪队列中选择一个进程投入运行。所以，选项 A 正确。

所以，本题的答案为 A。

31．答案：D。

分析：本题考察的是对死锁基础知识的理解。

所谓死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或

系统产生了死锁，这些永远在互相等待的进程称为死锁进程。举一个简单的例子：在吃饭的时候，只有拿到一双筷子后才能开始吃饭，如果有两个人，每个人都只拿了一根筷子，而等待另一根筷子可用的时候，就拿过来开始吃饭，此时，两个人都已经占用了部分资源（一根筷子），而等待另一个资源（另一根筷子），此时两个人永远都在等待对方释放资源，因此，发生了死锁。可以通过外力作用把一个人的筷子强制释放掉而给另外一个人而解决死锁。很显然，选项 D 的描述符合死锁的定义。

所以，本题的答案为 D。

32．答案：B。

分析：本题考察的是死锁知识。

本题中，不发生死锁的条件是至少能保证 1 个进程获得 3 台打印机资源。最坏的情况是 1 个进程获取了 3 台打印机资源，另外 N-1 个进程获取到 2 台打印机，等待获取第 3 台。

所以，本题可以构建如下等式关系：3+(N-1)*2=11，解算结果为 N=5。所以，选项 B 正确。

所以，本题的答案为 B。

33．答案：A。

分析：本题考察的是计算机网络与通信知识。

IP（Internet Protocol，网络之间互联的协议，简称为“网协”）是为计算机网络相互连接进行通信而设计的协议。在因特网中，它是能使连接到网上的所有计算机网络实现相互通信的一套规则，规定了计算机在因特网上进行通信时应当遵守的规则。

IP 协议定义在 OSI-RM（Open System Interconnect/Reference Model，开放式系统互联参考模型）的第三层即网络层，在 IP 协议中，规定了在 Internet 上进行通信时应遵守的规则，例如 IP 数据包的组成、路由器如何将 IP 数据包送到目的主机等。所以，选项 A 正确。

对于选项 B，处于应用层中的协议有 HTTP、FTP、SMTP 和 POP3 等。所以，选项 B 不正确。

对于选项 C，处于数据链路层中的协议有 PPP、ARP 及 RARP 等。所以，选项 C 不正确。

对于选项 D，处于传输层中的协议有 TCP、UDP 等。所以，选项 D 不正确。

所以，本题的答案为 A。

34．答案：D。

分析：本题考察的是计算机网络与通信知识。

对于选项 A，IP 协议主要关心的是如何将数据从一个设备经过一个互联网络发送到另一个设备。所以，选项 A 不正确。

对于选项 B，ICMP 是互联网控制报文协议，IP 通过 ICMP 来交换错误信息或者其他重要信息，它工作在网络层。所以，选项 B 不正确。

对于选项 C，ARP（Address Resolution Protocol，地址解析协议）是根据 IP 地址获取物理地址的一个 TCP/IP 协议。主机发送信息时将包含目标 IP 地址的 ARP 请求广播到网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该 IP 地址和物理地址存入本机 ARP 缓存中并保留一定时间，下次请求时直接查询 ARP 缓存以节约资源。所以，选项 C 不正确。

对于选项 D，RARP（Reverse Address Resolution Protocol，反向地址解析协议）与 ARP 相反。RARP 发出要反向解析的物理地址，并希望返回其对应的 IP 地址，应答包括由能够提供所需信息的 RARP 服务器发出的 IP 地址。虽然发送方发出的是广播信息，RARP 规定只有 RARP 服务器能产生应答。许多网络指定了多个 RARP 服务器，这样做既是为了平衡负载，也是为了作为出现问题时的备份。所以，选项 D 正确。

所以，本题的答案为 D。

35．答案：B。

分析：本题考察的是计算机网络与通信的基础知识。

本题中，130.63.160.2 是 B 类 IP 地址，而 B 类 IP 地址的前 16 位（两个字节）为网络号，后 16 位是主机号，划分子网就是将主机号中的一部分拿出来当作子网号，本题中，子网掩码为 255.255.255.0，也就是把前三个字节当成了网络号。

与 B 类 IP 地址默认的前两个字节作为网络号相比，第三个字节就是子网号，即 160，所以，这个 IP 的网络号是 130.63，子网号为 160，主机号是 2。所以，选项 B 正确。

所以，本题的答案为 B。

36．答案：C。

分析：本题考察的是计算机网络与通信的知识。

IP 地址根据网络 ID 的不同分为 5 种类型：A 类地址、B 类地址、C 类地址、D 类地址和 E 类地址。

（1）A 类 IP 地址

一个 A 类 IP 地址由 1 字节的网络地址和 3 字节主机地址组成，网络地址的最高位必须是“0”，地址范围为 1.0.0.0～126.0.0.0。可用的 A 类网络有 126 个，每个网络能容纳 1 亿多个　　主机。

（2）B 类 IP 地址

一个 B 类 IP 地址由 2 个字节的网络地址和 2 个字节的主机地址组成，网络地址的最高位必须是“10”，地址范围为 128.0.0.0～191.255.255.255。可用的 B 类网络有 16382 个，每个网络能容纳 6 万多个主机。

（3）C 类 IP 地址

一个 C 类 IP 地址由 3 字节的网络地址和 1 字节的主机地址组成，网络地址的最高位必须是“110”。地址范围为 192.0.0.0～223.255.255.255。C 类网络可达 209 万余个，每个网络能容纳 254 个主机。

（4）D 类 IP 地址

D 类 IP 地址的第一个字节以“1110”开始，它是一个专门保留的地址。它并不指向特定的网络，目前这一类地址被用在多点广播（Multicast）中。多点广播地址用来一次寻址一组计算机，它标识共享同一协议的一组计算机。

（5）E 类 IP 地址

E 类 IP 地址的第一个字节以“11110”开始，为将来使用保留。

通过上面分析可知，200.5.6.4 属于 192.0.0.0～223.255.255.255 范围内，属于 C 类地址范畴。所以，选项 C 正确。

所以，本题的答案为 C。

37．答案：B。

分析：本题考察的是计算机网络与通信的基础知识。

网关是局域网连接广域网的出口，可以工作在 OSI 模型网络层以上的不同层次。所以，选项 B 正确。

所以，本题的答案为 B。

三、问答题

1．答案：<? extends T>表示类型的上界，也就是说，参数化的类型可能是 T 或者 T 的子类。例如，下面的写法都是合法的赋值语句：

```
List<?extends Number> list = new ArrayList<Number>();
List<? Extends Number> list = new ArrayList<Integer>(); // Integer 是 Number 的子类
```

List<? Extends Number> list = new ArrayList<Float>(); // Float 也是 Number 的子类

<? extends T>被设计为用来读数据的泛型（只能读取类型为 T 的元素），原因如下：

（1）在上面赋值的示例中，对读数据进行分析

1）不管给 list 如何赋值，可以保证 list 里面存放的一定是 Number 类型或其子类，因此，可以从 list 列表里面读取 Number 类型的值。

2）不能从 list 中读取 Integer，因为 list 里面可能存放的是 Float 值，同理，也不可以从 list 里面读取 Float。

（2）对写数据进行分析

1）不能向 list 中写 Number，因为 list 中有可能存放的是 Float。

2）不能向 list 中写 Integer，因为 list 中有可能存放的是 Float。

3）不能向 list 中写 Float，因为 list 中有可能存放的是 Integer。

从上面的分析可以发现，只能从 List<? extends T> 读取 T，因为无法确定它实际指向列表的类型，从而无法确定列表里面存放的实际的类型，所以，无法向列表里面添加元素。

<? super T>表示类型下界，也就是说，参数化的类型是此类型的超类型（父类型）。

List<?super Float>list = new ArrayList<Float>()

List<?super Float>list = new ArrayList<Number>(); // Number 是 Float 的父类

List<?super Float>list = new ArrayList<Object>(); // Object 是 Number 的父类

<? super T>被设计为用来写数据的泛型（只能写入 T 或 T 的子类类型），不能用来读，分析如下：

（1）读数据

无法保证 list 里面一定存放的是 Float 类型或 Number 类型，因为有可能存放的是 Object 类型，唯一能确定的是 list 里面存放的是 Object 或其子类，但是无法确定具体子类的类型。正是由于无法确定 list 里面存放数据的类型，因此，无法从 list 里面读取数据。

（2）写数据

1）可以向 list 里面写入 Float 类型的数据（不管 list 里面实际存放的是 Float、Number 或者 Object，写入 Float 都是允许的）；同理，也可以向 list 里面添加 Float 子类类型的元素。

2）不可以向 list 里面添加 Number 或 Object 类型的数据，因为 list 中可能存放的是 Float 类型的数据。

下面给出两个泛型使用的场景：

```
public class Collections
{
        public static <T> void copy(List<? super T> dest, List<? extends T> src)

        {
            for (int i=0; i<src.size(); i++)
            dest.set(i,src.get(i));
        }
}
```

2．答案：两种单例模式的实现代码如下所示：

写法一：

```
public class Test
{
    private static Test test = new Test();
    public Test(){ }
```

```
        public static Test getInstance()
        {
            return test;
        }
    }
```

写法二：

```
    public class Test
    {
        private static Test test = null;
        private Test(){ }
        public static Test getInstance()
        {
            if (test == null)
            {
                test = new Test();
            }
            return test;
        }
    }
```

对于第一种写法，当类被加载的时候，已经创建好了一个静态的对象，因此，是线程安全的，但缺点是在这个对象还没有被使用的时候就已经被创建出来了。

对于第二种写法，缺点如下：这种写法不是线程安全的，例如当第一个线程执行判断语句 if(test==null)时，第二个线程执行判断语句 if(test==null)，接着第一个线程执行语句 test = new Test()，第二个线程也执行语句 test = new Test()，在这种多线程环境下，可能会创建出两个对象。当然，这种写法的优点是按需创建对象，只有对象被使用的时候才会被创建。

3．答案：接口是一种特殊形式的抽象类，使用接口完全有可能实现与抽象类相同的操作，但一般而言，抽象类多用于在同类事物中有无法具体描述的方法的场景，所以，当子类和父类之间存在有逻辑上的层次结构时，推荐使用抽象类，而接口多用于不同类之间，定义不同类之间的通信规则，所以，当希望支持差别较大的两个或者更多对象之间的特定交互行为时，应该使用接口。使用接口能大大降低软件系统的耦合度。

4．答案：int 和 Integer 的区别如下：

1）int 是 Java 语言提供的 8 种基本的原始数据类型之一，当作为对象的属性的时候，它的默认值为 0。而 Integer 是 Java 为 int 提供的封装类，默认值为 null。由此可见，int 无法区分未赋值与赋值为 0 的情况，而 Integer 却可以区分这两种情况。

2）int 是基本类型，在使用的时候是值传递；而 Integer 是引用。

3）int 只能用来运算，而 Integer 可以做更多的事情，因为 Integer 提供了很多有用的方法。

4）当需要往容器（例如 List）里存放整数时，无法直接存放 int，因为 List 里面放的都是对象，所以在这种情况下只能使用 Integer。

5．答案：本题可以采用递归法与非递归法两种方法实现。以下将分别对这两种方法进行分析。

方法一：递归法

具体步骤如下所示：

1）比较链表 1（head1）和链表 2（head2）的第一个结点数据，如果 head1.data<head2.data，

则把结果链表头结点指向链表 head1 中的第一个结点。

2）对剩余的链表 head1.next 和链表 2（head2）再调用同样的方法，比较得到结果链表的第二个结点，添加到合并后列表的后面。

3）一直递归调用步骤 2），直到两个链表的结点都被加到结果链表中。

实现代码如下所示：

```
class Node
{
    Node next = null;
    int data;
    public Node(int data)
    {
        this.data = data;
    }
}
public class Test
{
    public static Node mergeList(Node head1, Node head2)
    {
        if (head1 == null)
            return head2;
        if (head2 == null)
            return head1;
        Node head = null;
        if (head1.data < head2.data)
        {
            head = head1;
            head.next = mergeList(head1.next, head2);
        }
        else
        {
            head = head2;
            head.next = mergeList(head1, head2.next);
        }
        return head;
    }

    public static void main(String[] args)
    {
        Node head1=new Node(1);
        Node node3=new Node(3);
        head1.next=node3;
        Node node5=new Node(5);
        node3.next=node5;
        node5.next=null;
```

```
        Node head2=new Node(2);
        Node node4=new Node(4);
        head2.next=node4;
        Node node6=new Node(6);
        node4.next=node6;
        node6.next=null;

        Node mergeHead=mergeList(head1,head2);
        while(mergeHead!=null)
        {
            System.out.print(mergeHead.data+" ");
            mergeHead=mergeHead.next;
        }
    }
}
```

程序的运行结果为

```
1 2 3 4 5 6
```

方法二：非递归法

在遍历两个链表的过程中，改变当前链表指针的指向来把两个链表串联成一个有序的列表，实现代码如下：

```
public static Node mergeList(Node head1, Node head2)
{
    if (head1 == null)
        return head2;
    if (head2 == null)
        return head1;
    Node p1, p2, head;
    // 确定合并后的头结点
    if (head1.data < head2.data)
    {
        head = head1;
        p1 = head1.next;
        p2 = head2;
    }
    else
    {
        head = head2;
        p1 = head1;
        p2 = head2.next;
    }
    Node pcur = head;
    while (p1 != null && p2 != null)
    {
```

```
                // 把链表 head1 当前遍历的结点添加到合并后链表的尾部
                if (p1.data <= p2.data)
                {
                    pcur.next = p1;
                    pcur = p1;
                    p1 = p1.next;
                }
                // 把链表 head2 当前遍历的结点添加到合并后链表的尾部
                else
                {
                    pcur.next = p2;
                    pcur = p2;
                    p2 = p2.next;
                }
                // head2 链表已经遍历结束，把 head1 遍历剩余的结点添加到合并后链表的尾部
                if (p1 != null)
                {
                    pcur.next = p1;
                }
                if (p2 != null)
                {
                    pcur.next = p2;
                }
            }
            return head;
    }
```

当使用方法一中的 main 方法进行测试时，可以得到同样的运行结果。

6. 答案：如果没有内存的限制，可以首先将文件 a 中的 url 全部读入内存，放到 HashSet 中，接着从文件 b 中读取 url，每读取一个 url，就判断这个 url 在 HashSet 中是否存在，如果存在，那么这个 url 就是这两个文件共同的 url，否则不是。

由于题目要求内存大小只有 4GB，而每个文件的大小为 50 亿*64B=5*64GB=320GB，远远超出了内存限制，因此，无法一次将所有 url 读取到内存中，此时可以采取分批读取的方法。下面介绍两种常用的方法：

方法一：Hash 法

通过对 url 求 Hash 值，把 Hash 值相同的 url 放到一个单独的文件里，这样就可以把 50 亿个 url 分解成数量较小的 url，然后一次读入内存进行处理，具体实现思路如下：

首先遍历文件 a，对每个 url 求 Hash 值并散列到 1000 个文件中，求解方法为 h=hash(url)%1000，然后根据 Hash 的结果把这些 url 存放到文件 fa 中，通过散列，所有的 url 将会分布在（fa0，fa2，fa3，…，fa999）这 1000 个文件中。每个文件的大小大约为 300MB。同理，将文件 b 中的 url 也以同样的计算方式散列到文件 fb 中，所有的 url 将会分布在（fb0，fb1，fb2，…，fb999）这 1000 个文件中。显然，与 fa0 中相同的 url 只可能存在于 fb0 中，因此，只需要分别找出文件 fai 与 fbi（0≤i≤999）中相同的 url 即可。

此外，如果经过 Hash 法处理后，还有小文件占的内存大小超过 4GB，此时可以采用相

同的方法把文件分割为更小的文件进行处理。

方法二：Bloom filter 法

日常生活中很多地方都会遇到类似这样的问题，例如，在设计计算机软件系统时，在程序中经常需要判断一个元素是否在一个集合中；在字处理软件中，需要检查一个英语单词是否拼写正确；在 FBI，一个嫌疑人的名字是否已经在嫌疑名单上；在网络爬虫里，一个网址是否被访问过等。

针对这些问题，最直接的解决方法就是将集合中全部的元素都存储在计算机中，每当遇到一个新元素时，就将它和集合中的元素直接进行比较即可。这种做法虽然能够准确无误地完成任务，但存在一个问题，就是比较次数太多，效率比较低，当数据量不大时，这种效率低的问题并不显著，但是当数据量巨大时，例如在海量数据信息处理中，存储效率低的问题就显现出来了。例如邮箱总是需要过滤垃圾邮件，一种办法就是记录下那些发垃圾邮件的 Email 地址，可是由于那些发送者还会不停地再注册新的地址，如果使用哈希表，里面存储一亿个 Email 地址，一般而言，每个 Email 地址需要占用 16B，所以，一共需要 1 亿*16B，大约 1.6GB 的内存，除非是超级计算机，一般服务器是无法存储如此海量信息的。

Bloom filter 正是解决这一问题的有效方法，它是一种空间效率和时间效率很高的随机数据结构，用来检测一个元素是否属于一个集合。但它同样带来一个问题：牺牲了正确率，Bloom filter 以牺牲正确率为前提，来换取空间效率与时间效率的提高。当它判断某元素不属于这个集合时，该元素一定不属于这个集合；当它判断某元素属于这个集合时，该元素不一定属于这个集合。具体而言，查询结果有两种可能，即“不属于这个集合（绝对正确）”和“属于这个集合（可能错误）”。所以，Bloom filter 适合应用在对于低错误率可以容忍的场合。

它的基本原理是位数组与 Hash 函数的联合使用。具体而言，首先，Bloom filter 是一个包含了 m 位的位数组，数组的每一位都初始化为 0，然后定义 k 个不同的 Hash 函数，每个函数都可以将集合中的元素映射到位数组的某一位。当向集合中插入一个元素时，根据 k 个 Hash 函数可以得到位数组中的 k 个位，将这些位设置为 1。如果查询某个元素是否属于集合，那么根据 k 个 Hash 函数可以得到位数组中的 k 个位，查看这 k 个位中的值，如果有的位不为 1，那么该元素肯定不在此集合中；如果这 k 个位全部为 1，那么该元素可能在此集合中（在插入其他元素时，可能会将这些位置为 1，这样就产生了错误）。

下面通过一个实例具体了解 Bloom filter，如图 8 所示。

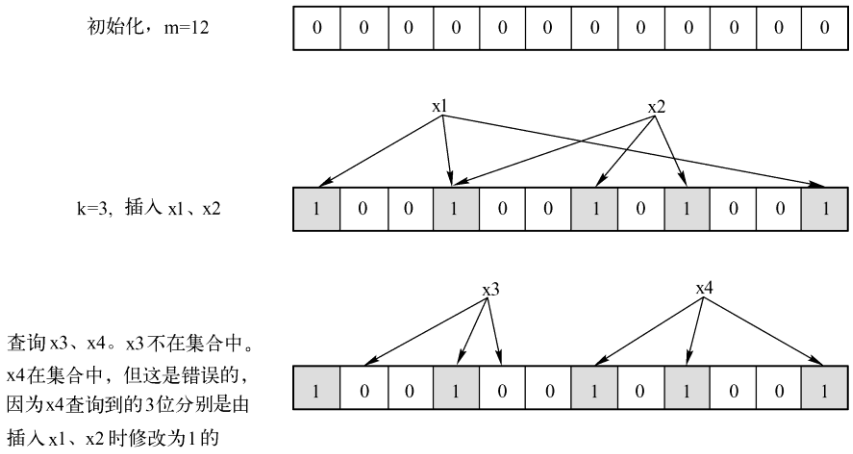


图 8　Bloom filter 实例

所以，使用 Bloom filter 的难点是如何根据输入元素个数 n，来确定位数组 m 的大小以及 Hash 函数。当 Hash 函数个数 k=(ln2)*(m/n)时错误率最小，在错误率不大于 E 的情况下，

m 至少要等于 n*lg(1/E)才能表示任意 n 个元素的集合。但 m 还应该更大些，因为还要保证位数组里至少一半为 0，则 m 应该大于等于 n*lg(1/E)*lge 大约为 n*lg(1/E)的 1.44 倍（lg 表示以 2 为底的对数）。

例如假设 E 为 0.01，即错误率为 0.01，则此时 m 应该大约为 n 的 13 倍，这样 k 大约是 8 个（注意：m 与 n 的单位不同，m 的单位是 bit，而 n 则是以元素个数为单位）。通常单个元素的长度都是有很多 bit 的，所以，使用 Bloom filter 内存通常都是节省的。

Bloom filter 的优点是具有很好的空间效率和时间效率。它的插入和查询时间都是常数，另外它不保存元素本身，具有良好的安全性。然而，这些优点都是以牺牲正确率为代价的。当插入的元素越多，错判“元素属于这个集合”的概率就越大。另外，Bloom filter 只能插入元素，却不能删除元素，因为多个元素的 Hash 结果可能共用了 Bloom filter 结构中的同一个位，如果删除元素，就可能会影响多个元素的检测。所以，Bloom filter 可以用来实现数据字典、进行数据的判重或者集合求交集。

对于本题而言：4GB 内存可以表示 340 亿 bit，把文件 a 中的 url 采用 Bloom filter 方法映射到这 340 亿 bit 上，然后遍历文件 b，判断是否存在。但是采用这种方法会有一定的错误率，只有当允许有一定的错误率的时候才可以使用这种方法。