队列

2018年5月3日 13:29

1.在循环队列中，存在两种方式来判断队列有没有满：（1）判断队列是否已满：(rear+1)%n==front；判断队列是否为空：front==rear；（2）设立标志位，当队列满时，标志位为1，空时标志位为0

(单选题) 设循环队列的存储空间为Q(1:35),初始状态为front = rear = 35。现经过一系列入队与退队运算后,front = 15,rear= 15,则循环队列中的元素个数为

A 15

B 16

C 20

D 0或35

(单选题) 循环队列的存储空间为Q(1:40)，初始状态为front=rear=40。经过一系列正常的入队与退队操作后，front=rear=15，此后又退出一个元素，则循环队列中的元素个数为（）

A 39，或0且产生下溢错误

B 14

C 40

D 15

正确答案：A 你的答案：B (错误)

顺序存储结构的循环队列

假设循环队列的队尾指针是rear，队头是front，其中QueueSize为循环队列的最大长度。

(1) 入队时队尾指针前进1：(rear+1)%QueueSize

(2) 出队时队头指针前进1：(front+1)%QueueSize

例1， 例2

(3) 队列长度：(rear-front+QueueSize)%QueueSize

例3：

现有一循环队列，其队头指针为front，队尾指针为rear；循环队列长度为N。其队内有效长度为？（假设队头不存放数据）

答案：(rear-front+N)%N

(4) 队空和队满的条件

为了区分队空还是堆满的情况，有多种处理方式：

方式1： 牺牲一个单元来区分队空和队满，入队时少用一个队列单元，即约定以"队头指针在队尾指针的下一位置作为队满的标志"。

> 队满条件为：(rear+1)%QueueSize==front
> 队空条件为：front==rear
> 队列长度为：(rear-front++QueueSize)%QueueSize

方式2： 增设表示队列元素个数的数据成员size，此时，队空和队满时都有front==rear。

> 队满条件为：size==QueueSize
> 队空条件为：size==0

方式3： 增设tag数据成员以区分队满还是队空

> tag表示0的情况下，若因删除导致front==rear，则队空；
> tag等于1的情况，若因插入导致front==rear则队满

例1：

循环队列的存储空间为Q(1:50)，初始状态为front=rear=50。

经过一系列正常的入队与退队操作后，front=rear=25。此后又插入一个元素，则循环队列中的元素个数为多少？

答案：1，或50且产生上溢错误

2.

8 循环队列存储在数组A[0..m]中,则入队时的操作为()

正确答案: D 你的答案: C (错误)

```
rear=rear+1
```

```
rear=(rear+1)mod(m-1)
```

```
rear=(rear+1)mod m
```

```
rear=(rear+1)mod(m+1)
```

rear=(rear+1)%(m+1) //m+1代表有m+1个空间。。它是0到m的数组。。看清楚,选D

栈

2018年5月3日　13:39

1.栈经常用于函数的调用。

(单选题)

一个栈的入栈序列为1,2,3,...,n，其出栈序列是 $p_1, p_2, p_3, \cdots p_n$。若 $p_2 = 3$，则 p_3 可能取值的个数是（）

A　n-3

B　n-2

C　n-1

D　无法确定

2.

汉诺塔：汉诺塔（又称河内塔）问题是源于印度一个古老传说的益智玩具。大梵天创造世界的时候做了三根金刚石柱子，在一根柱子上从下往上按照大小顺序摞着64片黄金圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。

设移动n个盘子的汉诺塔问题需要g(n)次移动操作来完成。由展示移动过程算法可知g(n)应是三部分之和。

(1) 将n个盘上面的n-1个盘子借助C桩从A桩移到B桩上，需g(n-1)次移动；

(2) 然后将A桩上第n个盘子移到C桩上（1次）；

(3) 最后，将B桩上的n-1个盘子借助A桩移到C桩上，需g(n-1)次。

因而有递归关系：

g(n)=2*g(n-1)+1

初始条件(递归出口)：

g(1)=1

即 1、3、7、15、31。。。即g(n) = 2^n -1

总结起来：规则就是一次移动一个盘，无论何时，小盘在上，大盘在下，在游戏中，总共有n个金属盘片的塔叫做n阶汉诺塔，完成n阶汉诺塔，则最少要移动2的n次方-1次

(单选题) 4个圆盘的Hanoi塔,总的移动次数为()

A 7

B 8

C 15

D 16

数组

2018年5月4日 22:08

1.数组通常具有的两种基本操作**查找和修改**

2.

10 查看解析 下面的说法那个正确

```
#define NUMA 10000000
#define NUMB 1000
int a[NUMA], b[NUMB];

void pa()
{
    int i, j;
    for(i = 0; i < NUMB; ++i)
        for(j = 0; j < NUMA; ++j)
            ++a[j];
}
void pb()
{
    int i, j;
    for(i = 0; i < NUMA; ++i)
        for(j = 0; j < NUMB; ++j)
            ++b[j];
}
```

99

欧阳

测试时pb比pa快，数组a比数组b大很多，可能跨更多的页，缺页率高或者缓存命中更低，所以pb快

推荐 编辑于 2015-08-22 19:12:10

总结

2018年5月4日 22:14

数据结构	查找	插入	删除	遍历
数组	O(N)	O(1)	O(N)	—
有序数组	O(logN)	O(N)	O(N)	O(N)
链表	O(N)	O(1)	O(N)	—
有序链表	O(N)	O(N)	O(N)	O(N)
二叉树（一般情况）	O(logN)	O(logN)	O(logN)	O(N)
二叉树（最坏情况）	O(N)	O(N)	O(N)	O(N)
平衡树（一般情况和最坏情况）	O(logN)	O(logN)	O(logN)	O(N)
哈希表	O(1)	O(1)	O(1)	—

链表

2018年5月4日　　22:29

1.

在下述几种树中,()可以表示静态查找表

A. 次优查找树

B. 二叉排序树

C. B-树

D. 平衡二叉树

静态查找表：只做查找操作的查找表。①查询某个“特定”数据元素是否在查找表中；②检索某个“特定”数据元素和各种属性。
动态查找表：在查找过程同时插入查找表中不存在的数据元素，或者从查找表中删除已经存在的某个数据元素。

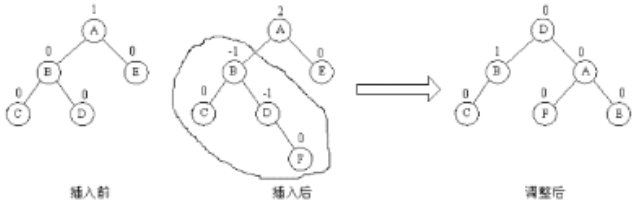
树

2018年5月4日　　22:50

插入前　　插入后　　调整后

（ 3 ）LR 型平衡旋转法

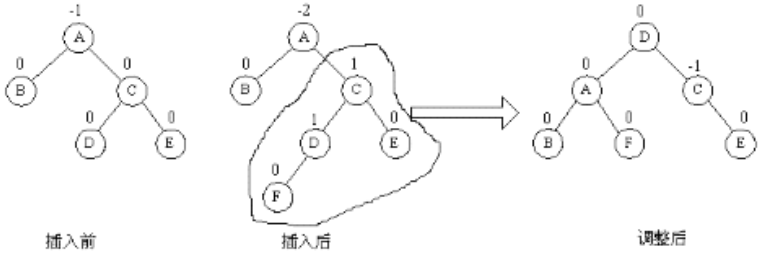
由于在 A 的左孩子 B 的右子数上插入结点 F ，使 A 的平衡因子由 1 增至 2 而失去平衡。故需进行两次旋转操作（先逆时针，后顺时针）。即先将 A 结点的左孩子 B 的右子树的根结点 D 向左上旋转提升到 B 结点的位置，然后再把该 D 结点向右上旋转提升到 A 结点的位置。即先使之成为 LL 型，再按 LL 型处理 。



如图中所示，即先将圆圈部分先调整为平衡树，然后将其以根结点接到 A 的左子树上，此时成为 LL 型，再按 LL 型处理成平衡型。

（ 4 ）RL 型平衡旋转法

由于在 A 的右孩子 C 的左子树上插入结点 F ，使 A 的平衡因子由 -1 减至 -2 而失去平衡。故需进行两次旋转操作（先顺时针，后逆时针），即先将 A 结点的右孩子 C 的左子树的根结点 D 向右上旋转提升到 C 结点的位置，然后再把该 D 结点向左上旋转提升到 A 结点的位置。即先使之成为 RR 型，再按 RR 型处理。



如图中所示，即先将圆圈部分先调整为平衡树，然后将其以根结点接到 A 的左子树上，此时成为 RR 型，再按 RR 型处理成平衡型。

平衡化靠的是旋转。 参与旋转的是 3 个节点（其中一个可能是外部节点 NULL ），旋转就是把这 3 个节点转个位置。注意的是，左旋的时候 p->right 一定不为空，右旋的时候 p->left 一定不为空，这是显而易见的。

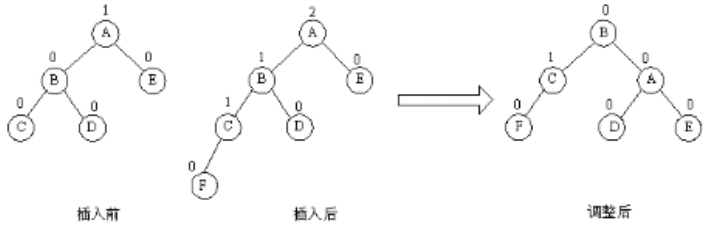
如果从空树开始建立，并时刻保持平衡，那么不平衡只会发生在插入删除操作上，而不平衡的标志就是出现 bf == 2 或者 bf == -2 的节点。

若向平衡二叉树中插入一个新结点后破坏了平衡二叉树的平衡性。首先要找出插入新结点后失去平衡的最小子树根结点的指针。然后再调整这个子树中有关结点之间的链接关系，使之成为新的平衡子树。当失去平衡的最小子树被调整为平衡子树后，原有其他所有不平衡子树无需调整，整个二叉排序树就又成为一棵平衡二叉树。

失去平衡的最小子树是指以离插入结点最近，且平衡因子绝对值大于 1 的结点作为根的子树。假设用 A 表示失去平衡的最小子树的根结点，则调整该子树的操作可归纳为下列四种情况。

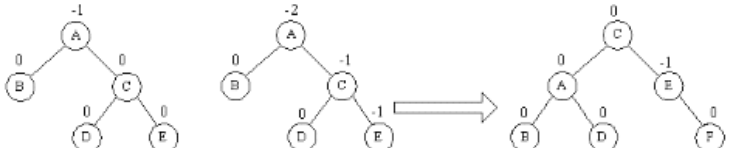
（ 1 ）LL 型平衡旋转法

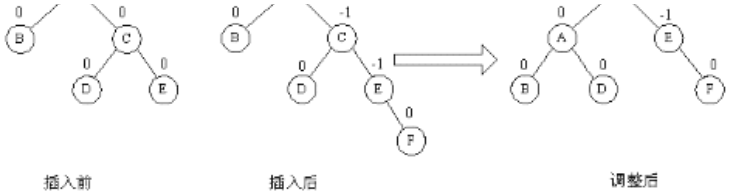
由于在 A 的左孩子 B 的左子树上插入结点 F ，使 A 的平衡因子由 1 增至 2 而失去平衡。故需进行一次顺时针旋转操作。 即将 A 的左孩子 B 向右上旋转代替 A 作为根结点， A 向右下旋转成为 B 的右子树的根结点。而原来 B 的右子树则变成 A 的左子树。



（ 2 ）RR 型平衡旋转法

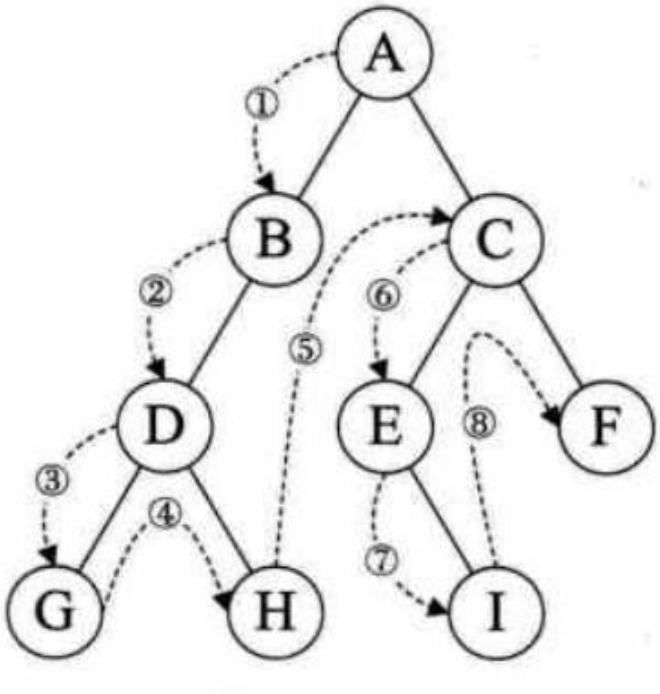
由于在 A 的右孩子 C 的右子树上插入结点 F ，使 A 的平衡因子由 -1 减至 -2 而失去平衡。故需进行一次逆时针旋转操作。即将 A 的右孩子 C 向左上旋转代替 A 作为根结点， A 向左下旋转成为 C 的左子树的根结点。而原来 C 的左子树则变成 A 的右子树。



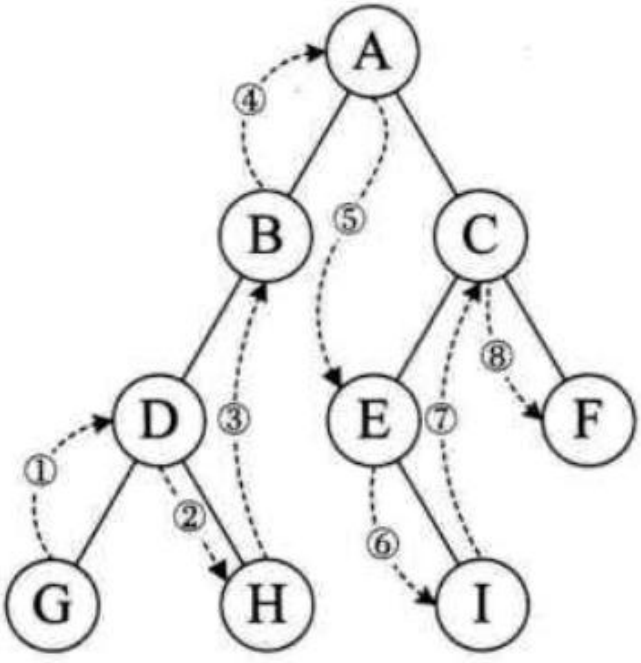


树的遍历

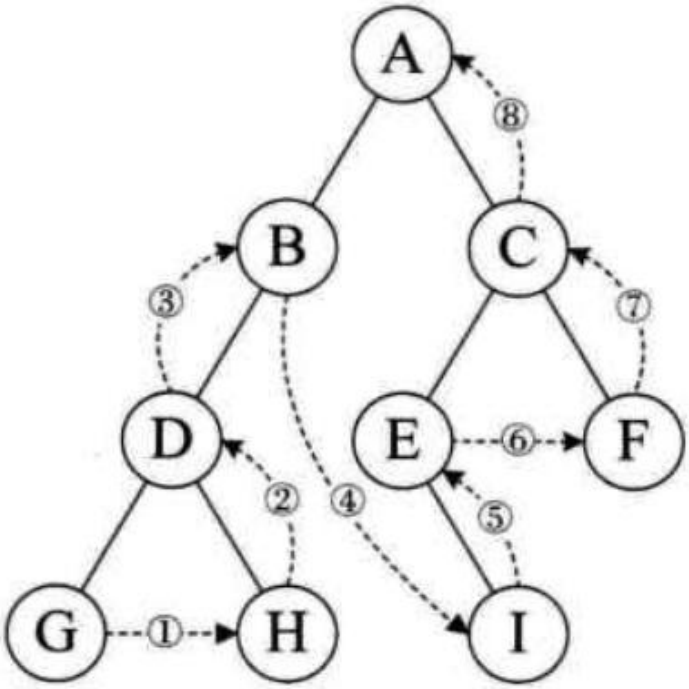
（1）树的前序遍历



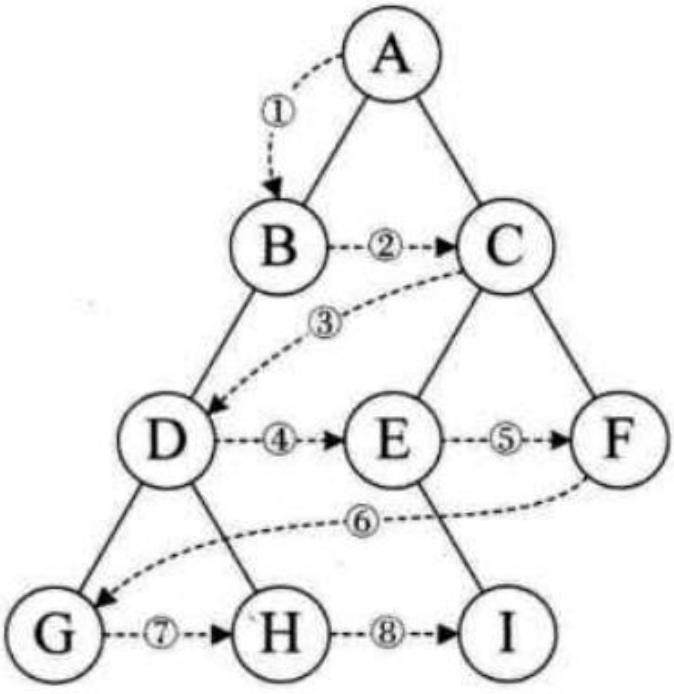
（2）树的中序遍历



（3）树的后序遍历



（4）树的层序遍历



二叉树遍历的性质①已知前序遍历序列和中序遍历序列，可以唯一确定一棵二叉树；②已知后序遍历序列和中序遍历序列，可以唯一确定一棵二叉树。

8.树、二叉树、森林之间的转换和关系

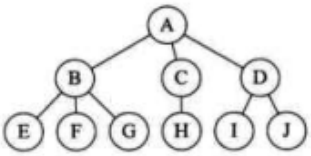
（1）树转换为二叉树

①加线：在所有兄弟结点之间加一条线

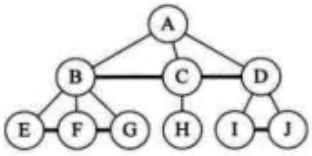
②去线。对树中的每一个结点，只保留它与第一个孩子结点的连线，删除它与其他孩子结点之间的连线

③层次调整。以树的根节点为轴心，将整棵树顺时针旋转一定的角度，使之结构层次分明。注意，第一个孩子一定是二叉树结点的左孩子，兄弟转换过来的孩子是结点的

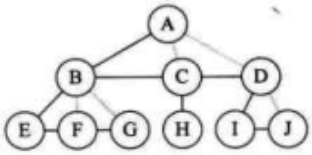
右孩子。



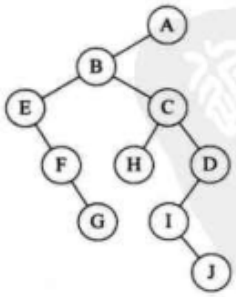
树



步骤1：给兄弟加线



步骤2：给除长子外的孩子去线



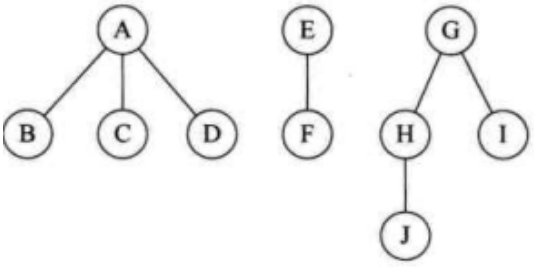
步骤3：层次调整

（2）森林转换为二叉树

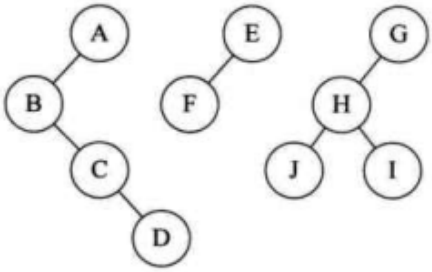
森林是由若干棵树组成的，所以完全可以理解为，森林中的每一棵树都是兄弟，可以按照兄弟的处理办法来操作。步骤如下：

1．把每个树转换为二叉树。

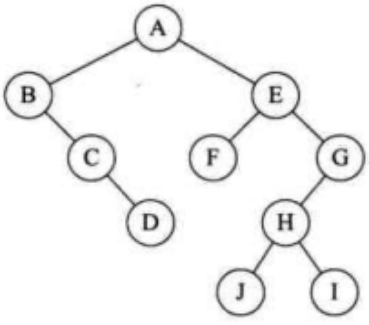
2．第一棵二叉树不动，从第二棵二叉树开始，依次把后一棵二叉树的根结点作为前一棵二叉树的根结点的右孩子，用线连接起来。当所有的二叉树连接起来后就得到了由森林转换来的二叉树。



拥有三棵树的森林



步骤1:森林中每棵树转换为二叉树



步骤2：将所有二叉树转换为一棵二叉树

（3）二叉树转换成树

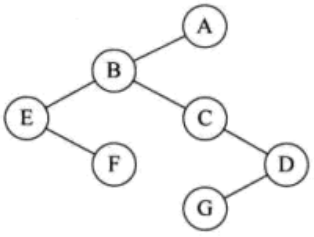
二叉树转换为树是树转换为二叉树的逆过程，也就是反过来做而已。如图 6-11-4 所示。步骤如下：

1．加线。若某结点的左孩子结点存在，则将这个左孩子的右孩子结点、右孩子的右孩子结点、右孩子的右孩子的右孩子结点……哈，反正就是左孩子的 n 个

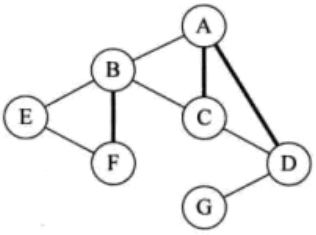
右孩子结点都作为此结点的孩子。将该结点与这些右孩子结点用线连接起来。

2．去线。删除原二叉树中所有结点与其右孩子结点的连线。

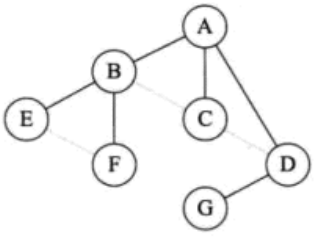
3．层次调整。使之结构层次分明。



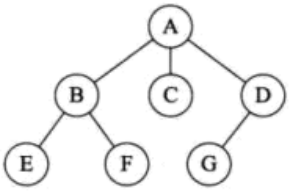
二叉树



步骤1：加线



步骤2：去线



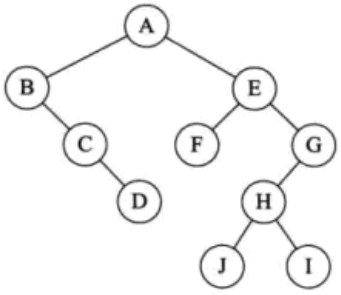
步骤3：层次调整

（4）二叉树转换成森林

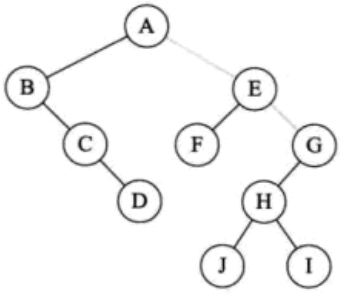
判断一棵二叉树能不能转换成一棵树或者森林，标准很简单，就是看这课二叉树有没有右孩子。有就是森林，没有就是树

1．从根结点开始，若右孩子存在，则把与右孩子结点的连线删除，再查看分离后的二叉树，若右孩子存在，则连线删除……，直到所有右孩子连线都删除为止，得到分离的二叉树。

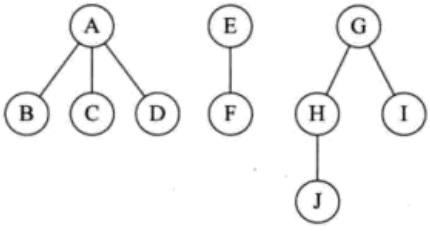
2．再将每棵分离后的二叉树转换为树即可。



二叉树



步骤1：寻找右孩子去线



步骤2：将分离的二叉树转换成树

（5）树和森林的遍历

①

树的遍历分为两种方式。

1．一种是先根遍历树，即先访问树的根结点，然后依次先根遍历根的每棵子树。

2．另一种是后根遍历，即先依次后根遍历每棵子树，然后再访问根结点。比如图 6-11-4 中最右侧的树，它的先根遍历序列为 ABEFCDG，后根遍历序列为 EFBCGDA。

②

森林的遍历也分为两种方式：

1．**前序遍历**：先访问森林中第一棵树的根结点，然后再依次先根遍历根的每棵子树，再依次用同样方式遍历除去第一棵树的剩余树构成的森林。比如图 6-11-5 右侧三棵树的森林，前序遍历序列的结果就是 ABCDEFGHJI。

2．**后序遍历**：是先访问森林中第一棵树，后根遍历的方式遍历每棵子树，然后再访问根结点，再依次同样方式遍历除去第一棵树的剩余树构成的森林。比如图

总结:森林的前序遍历和二叉树的前序遍历结果相同；森林的后序遍历和二叉树的中序遍历结果相同

10.赫夫曼算法

通过刚才的步骤，我们可以得出构造赫夫曼树的赫夫曼算法描述。

1．根据给定的 n 个权值{ $w_1,w_2,\cdots,w_n$ }构成 n 棵二叉树的集合 F={ $T_1,T_2,\cdots,T_n$ }，其中每棵二叉树 T_i 中只有一个带权为 w_i 根结点，其左右子树均为空。

2．在 F 中选取两棵根结点的权值最小的树作为左右子树构造一棵新的二叉树，且置新的二叉树的根结点的权值为其左右子树上根结点的权值之和。

3．在 F 中删除这两棵树，同时将新得到的二叉树加入 F 中。

4．重复 2 和 3 步骤，直到 F 只含一棵树为止。这棵树便是赫夫曼树。

2018.05.22

2018年5月22日 9:00

（一）数据结构部分

1.栈和队列

1 查看解析 设栈S和队列Q的初始状态为空，元素e1，e2，e3，e4，e5，e6依次压入栈S,一个元素出栈后即进入队列Q，若出队列的顺序为e2,e4,e3,e6,e5,e1则栈S的容量要求最小值为

答案：3

每操作一次 栈的状态是 e1

e1 e2

e1

e1 e3

e1 e3 e4

e1 e3

e1

e1 e5

e1 e5 e6

e1 e5

e1

空

所以要求最小值应该为 3

B

2.字符串

3 串是一种特殊的线性表，其特殊性体现在（ ）。

串是一种特殊的线性表，它的每一个结点是一个字符，所以串也被称为字符串。

3.叙述顺序表

顺序表是在计算机内存中以数组的形式保存的线性表，线性表的顺序存储使用一组连续的存储单元依次存储线性表的各个元素，使得线性表在逻辑结构上相邻的数据元素存储在相邻的物理存储单元中，即通过数据元素物理存储的相邻关系来反应数据元素之间逻辑上的相邻关系。采用顺序存储结构的线性表通常称为顺序表。顺序表是将表中的结点一次存放在计算机内存中的一组连续的存储单元中。

5 查看解析 关于顺序表叙述，正确的是（ ）

4.数组

7 若二维数组 a 有 m 列，则在数组元素 a[i][j] 前的元素个数为（ ）

5.二叉平衡查找树

①左子树中所有的节点的值小于根的值，右子树中所有节点的值大于根的值

②左右子树的高度差绝对值为0或者1

6.偏数学类的题

13 设输入序列是1,3,5…m,经过栈的作用后输出序列的第一个元素是m,则输出序列中第i个输出元素是()

7.森林的遍历

14 查看解析 将一棵树t转换为孩子兄弟链表表示的二叉树h,则t的后序遍历是h的()

正确答案: B 你的答案: 空 (错误)

前序遍历

中序遍历

后序遍历

森林是树的集合，有两种遍历方式

①先序遍历森林：若森林不为空，则可按照以下次序进行遍历（1）访问森林第一棵树的根节点（2）先序遍历第一棵树中的子树森林（3）先序遍历除去第一棵树之后剩余的树构成的森林。

②中序遍历森林：若森林不为空，则可按照以下次序进行遍历（1）中序遍历第一棵树的子树森林（2）访问森林中第一棵树的根节点（3）中序遍历除去第一棵树之后剩余的树构成的森林。

总结：

①树的前序遍历，对应二叉树的前序遍历

②树的后序遍历，对应二叉树的中序遍历

③树的层次遍历，对应二叉树的后序遍历

④森林的前序遍历，对应二叉树的前序遍历

⑤森林的中序遍历，对应二叉树的中序遍历

8.二叉排序树

分别以下列序列构造二叉排序树，与用其它三个序列所构造的结果不同的是（ ）

A. (100,80,60,90,120,130,110)

B. (100,120,110,130,80,60,90)

C. (100,80,90,60,120,110,130)

D. (100,60,80,90,120,110,130)

答案：D

9.关键路径

17 下面关于求关键路径的说法不正确的是()

正确答案: A 你的答案: D (错误)

一个事件的最迟开始时间为以该事件为尾的弧的活动最迟开始时间与该活动的持续时间的差

求关键路径是以拓扑排序为基础的

一个事件的最早开始时间同以该事件为尾的弧的活动最早开始时间相同

关键活动一定位于关键路径上

10.二叉树的基本知识点

19 查看解析 关于二叉树，下面说法正确的是？

正确答案: B C 你的答案: B C D (错误)

对于N个节点的二叉树，其高度为nlog2n

一个具有1025个节点的二叉树，其高度范围在11~1025之间

二叉树的先序遍历是EFHIGJK，中序遍历为HFIEJKG，该二叉树的右子树的根为G

二叉树中至少有一个节点的度为2

bc

a 高度应该是 $[\log_2 N]+1$

b 对 最低考虑完全二叉树,最高就成链表了. 刚好是长度.

c 根据定义来.

d 二叉树只有一个节点.时候 不成立

编辑于 2015-12-01 15:26:04

（二）Java部分

（1）异常

2 关于异常的编程，以下描述错误的是：（ ）

正确答案: C 你的答案: A (错误)

在有除法存在的代码处，抛不抛出异常均可

```
int i=Integer.parseInt("123a");将产生NumberFormatException
```

```
int a[]=null; a[0]=1; 将产生ArrayIndexOutOfBoundsException
```

输入输出流编程中，读和写时都必须抛出IOException

NumberFarmatExceptio数字格式异常

ArrayIndexOutOfBoundException数组下标越界异常

A.除数为0 等ArithmeticException，是RuntimException的子类。而运行时异常将由运行时系统自动抛出，不需要使用throw语句。Java编译器 允许忽略运行时异常，一个方法可以既不捕捉，也不声明抛出运行时异常。

C.产生NullPointerException。

2.

在一个基于分布式的游戏服务器系统中，不同的服务器之间，哪种通信方式是不可行的（ ）？

A. 管道

B. 消息队列

C. 高速缓存数据库

D. 套接字

解释：管道是一种半双工的通信方式，数据只能单向流动，而且只能在具有亲缘关系的进程间使用。进程的亲缘关系通常是指父子进程关系

3.套接字

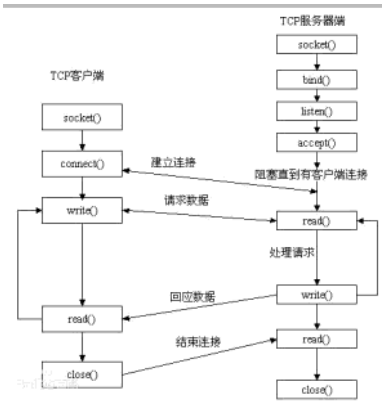
socket编程中，以下哪个socket的操作是不属于服务端操作的（ ）？

A. accept

B. listen

C. connect

D. close



答案：C

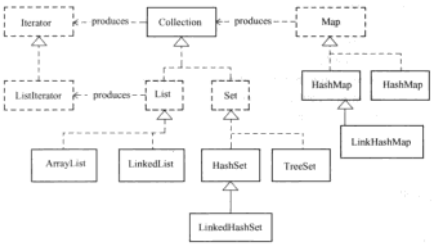
4.Set、List、Map三个接口

①Set是数学意义上集合的概念，最主要的特点是集合的元素不能重复，因此存入Set的每个元素都必须定义equals（ ）方法来确保对象的唯一性。该接口有两个实现类：HashSet和TreeSet。其中TreeSet实现了SortedSet接口，因此TreeSet容器中的元素是有序的

②List又称为有序的Collection。它是按照对象进入的顺序保存对象，能对列表中没有元素的插入和删除位置进行精确的控制。同时，它可以保存重复的对象。LinkList、List和Vector接口都实现了List接口。

③Map提供了一个从键映射到值的数据结构。用于保存键值对，其中值可以重复，但键是唯一的，不能重复。Java类库中有多个实现该接口的类：HashMap、TreeMap、LinkedHashMap、WeakHashMap和IdentityHashMap。它们的执行效率完全不同。HashMap是基于散列表实现的，采用对象的HashCode进行快速查询；LinkedHashMap采用列表来维护内部的顺序。TreeMap基于红黑树的数据结构来实现的，内部顺序是按需排列的。

Collection 的框架类图如图 4-12 所示。



5.ArrayList、LinkedList、HashMap三者的异同点

（1）首先三类都是在java.util包中，均为可伸缩数组，即可以动态改变长度的数组

（2）ArrayList和Vector都是基于存储元素的Object[ ] array来实现的，它们会在内存中开辟一块连续的空间来存储，由于数据存储是连续的，因此，它们支持用下标来访问元素，同时索引数据的速度比较快。缺点：在插入元素时需要移动容器中的元素，所以对数据插入的执行比较慢。ArrayList和Vector都有一个初始化的容量大小，当里面的存储元素超过这个大小时就需要动态的扩充它们的存储空间，为了提高程序的效率，每次扩充容量不是简单的扩充一个存储单元，而是一次性增加多个存储单元。Vector默认扩充为原来的2倍（每次扩充空间大小是可以设置的），ArrayList默认扩充为原来的1.5倍（没有提供方法来设置空间扩充的方法）

（3）两者最大的区别是synchronization（同步）的使用，没有一个ArrayList是同步的，而Vector的绝大多数方法都是同步的（例如add、insert、remove、set、equals、hashcode）等都是直接或者间接同步的，所以Vector是线程安全的机制，ArrayList不是线程安全的，正是由于Vector提供了线程安全的机制，其性能也要略逊于ArrayList。

（4）LinkedList是采用双向列表来实现的，对数据的索引需要从列表头开始遍历，因此用于随机访问效率较低，但是插入元素时不需要对数据进行移动，因此插入效率比较高。LinkedList是非线程安全容器。

总结：在实际中这几种容器应该如何选用呢？①当数据主要操作是索引或者只在集合的末端增加、删除元素时，使用ArrayList或Vector效率比较高②当对数据的操作主要为指定位置的插入或删除操作时，使用LinkedList效率比较高③当在多线程中使用荣放弃是（即多个线程同时访问该容器），选用Vector比较安

6.初始化顺序

以下程序执行的结果是：

```
class X{
    Y y=new Y();
    public X(){
        System.out.print("X");
    }
}
class Y{
    public Y(){
        System.out.print("Y");
    }
}
public class Z extends X{
    Y y=new Y();
    public Z(){
        System.out.print("Z");
    }
    public static void main(String[] args) {
        new Z();
    }
}
```

A. ZYXX

B. ZYXY

C. YXYZ

D. XYZX

答案C

初始化过程：

1. 初始化父类中的静态成员变量和静态代码块 ；

2. 初始化子类中的静态成员变量和静态代码块 ；

3.初始化父类的普通成员变量和代码块，再执行父类的构造方法；

4.初始化子类的普通成员变量和代码块，再执行子类的构造方法；

（1）初始化父类的普通成员变量和代码块，执行 Y y=new Y(); 输出Y

（2）再执行父类的构造方法；输出X

（3）初始化子类的普通成员变量和代码块，执行 Y y=new Y(); 输出Y

（4）再执行子类的构造方法；输出Z

所以输出YXYZ

7.算法结构

下列不属于算法结构的是（ ）

A. 输入数据

B. 处理数据

C. 存储数据

D. 输出结果

算法包括0个或多个输入，1个或多个输出，中间有无穷个处理过程

答案：C

8.读程序题

（1）

当你编译和运行下面的代码时，会出现下面选项中的哪种情况？

```
public class Pvf{
            static boolean Paddy;
        public static void main(String args[]){
            System.out.println(Paddy);
        }
}
```

A. 编译时错误

B. 编译通过并输出结果false

C. 编译通过并输出结果true

D. 编译通过并输出结果null

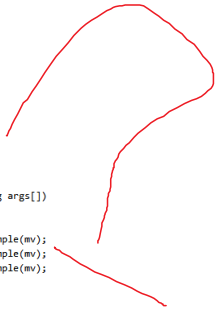
类中声明的变量有默认的初始值；方法中声明的变量没有默认的初始值，必须在定义时初始化，否则在访问该变量时会出错。

数据类型	初始值
byte	0
short	0
int	0
long	0L
char	'\u0000'
float	0.0f
double	0
boolean	false
所有引用类型	null

（2）

执行如下程序，输出结果是（ ）

```
class Test
{
    private int data;
    int result = 0;
    public void m()
    {
        result += 2;
        data += 2;
        System.out.print(result + "  " + data);
    }
}
class ThreadExample extends Thread
{
    private Test mv;
    public ThreadExample(Test mv)
    {
        this.mv = mv;
    }
    public void run()
    {
        synchronized(mv)
        {
            mv.m();
        }
    }
}
class ThreadTest
{
    public static void main(String args[])
    {
        Test mv = new Test();
        Thread t1 = new ThreadExample(mv);
        Thread t2 = new ThreadExample(mv);
        Thread t3 = new ThreadExample(mv);
        t1.start();
        t2.start();
        t3.start();
    }
}
```



A. 0 22 44 6

B. 2 42 42 4

C. 2 24 46 6

D. 4 44 46 6

Test mv =newTest()声明并初始化对data赋默认值

使用synchronized关键字加同步锁线程依次操作m()

t1.start();使得result=2,data=2,输出即为2 2

t2.start();使得result=4,data=4,输出即为4 4

t3.start();使得result=6,data=6,输出即为6 6

System.out.print(result +" "+ data);,是print()方法不会换行,输出结果为2 24 46 6

（3）

如下代码，执行test()函数后，屏幕打印结果为（ ）

```
public class Test2
{
    public void add(Byte b)
    {
        b = b++;
    }
    public void test()
    {
        Byte a = 127;
        Byte b = 127;
        add(++a);
        System.out.print(a + " ");
        add(b);
        System.out.print(b + "");
    }
}
```

A. 127 127

B. 128 127

C. 129 128

D. 以上都不对

答案 “D

（4）

以下程序的输出结果为

```
class Base{
    public Base(String s){
        System.out.print("B");
    }
}
public class Derived extends Base{
    public Derived (String s) {
        System.out.print("D");
    }
    public static void main(String[] args){
        new Derived("C");
    }
}
```

A. BD

B. DB

C. C

D. 编译错误

答案：D。子类构造方法在调用时必须先调用弗雷德，由于父类没有无参构造，必须在子类中显示调用

```
public Derived(String s){
        super("s");
        System.out.print("D");
    }
```

9不知道是什么知识点

以下是java concurrent包下的4个类，选出差别最大的一个

A. Semaphore

B. ReentrantLock

C. Future

D. CountDownLatch

Semaphore:类，控制某个资源可同时被访问的个数
ReentrantLock：类，具有与使用synchronized方法和语句所访问的隐式监视器锁相同的一些基本行为和语义，但功能更强大
Future：接口，表示异步计算的结果
CountDownLatch：类，可以用来在一个线程中等待多个线程完成任务的类

10.Spring框架

关于Spring MVC的核心控制器DispatcherServlet的作用，以下说法错误的是（ ）？

A. 它负责处理HTTP请求

B. 加载配置文件

C. 实现业务操作

D. 初始化上下应用对象ApplicationContext

SpringMVC的原理：SpringMVC是Spring中的模块，它实现了mvc设计模式的web框架，首先用户发送请求，请求到达SpringMVC的前端控制器（DispatcherServlet），前端控制器根据用户的url请求处理器映射查找匹配该url的handler并返回一个执行链，前端控制器再请求处理器适配器调用相应的hander进行处理并返回给前端控制器一个modelAndView，前端控制器再请求视图解析器对返回的视图逻辑进行解析，最后前端控制器将返回的视图进行渲染并把数据装入request域返回给用户。
DispatcherServlet作为sprMVC的前端控制器，负责接收用户的请求并根据用户的请求返回相应的视图给用户。实现业务操作在Service层，所以C错误。

11.HttpServlet

下面哪个不属于HttpServletResponse接口完成的功能？

A. 设置HTTP头标

B. 设置cookie

C. 读取路径信息

D. 输出返回数据

A：设置HTTP头标

```
response.setHeader("Refresh","3"); //三秒刷新页面一次
```

B：设置cookie

```
Cookie c1 = new Cookie("username","only");
response.addCookie(c1);
```

C（错误）：读取路径信息,request读取路径信息

```
从request获取各种路径总结
request.getRealPath("url"); // 虚拟目录映射为实际目录
request.getRealPath("./");    // 网页所在的目录
request.getRealPath("../"); // 网页所在目录的上一层目录
request.getContextPath();    // 应用的web目录的名称
```

D：输出返回数据

```
HttpServleteResponse.getOutputStream().write();
```

12.对象序列化

以下关于对象序列化描述正确的是

A. 使用FileOutputStream可以将对象进行传输

B. 使用PrintWriter可以将对象进行传输

C. 使用transient修饰的变量不会被序列化

D. 对象序列化的所属类需要实现Serializable接口

在Java中，对象的序列化可以通过两个接口来实现。若实现的是Serializable接口，则所有序列化将会自动执行；若实现的是Externalizable接口，则没有任何东西可以自动序列化，（Externailzable接口是Serizable接口的子类），需要在writeExternal方法中进行手工指定所要序列化的变量，这与是否被transient修饰无关。

使用ObjectOutputStream和ObjectInputStream可以将对象进行传输.

声明为static和transient类型的成员数据不能被串行化。因为static代表类的状态， transient代表对象的临时数据。

13.关于字符串数组的定义

Which statement declares a variable a which is suitable for referring to an array of 50 string objects?
（Java）

A. char a[][];

B. String a[];

C. String[] a;

D. Object a[50];

E. String a[50];

F. Object a[];

答案：BCF

在Java中，声明一个数组不能直接限定数组的长度，只有在创建实例化对象时，才能给定数组的长度。

A：char[][] 定义了二位字符数组。在Java中，**使用字符串对char数组赋值，必须使用toCharArray()方法进行转换**。所以A错误。

B、C：在Java中定义String数组，有**两种定义方式：String a[]和String[] a。所以B、C正确。**

D、E：数组是**一个引用类型变量**，因此**使用它定义一个变量时，仅仅定义了一个变量，这个引用变量还未指向任何有效的内存，因此定义数组不能指定数组的长度。所以D、E错误。**

F：**Object类是所有类的父类。子类其实是一种特殊的父类，因此子类对象可以直接赋值给父类引用变量，无须强制转换，这也被称为向上转型。这体现了多态的思想。所以F正确。**

最后选B、C、F

14.依赖式注入

下面关于依赖注入(DI)的说法不正确的是()

A. 只有通过Spring才能实现依赖注入(DI)

B. 依赖注入的主要目的是解耦合

C. 常见的依赖注入方式有Setter和构造方法

查看正确选项

答案：A

Spring依赖式注入（DI）的三种方式：（1）接口注入（2）Setter方法注入（3）构造器注入

依赖注入是一种思想，或者说是一种设计模式，在java中是通过反射机制实现的，与具体框架无关。

15.java中的字符流和字节流

BufferedReader的父类是以下哪个？

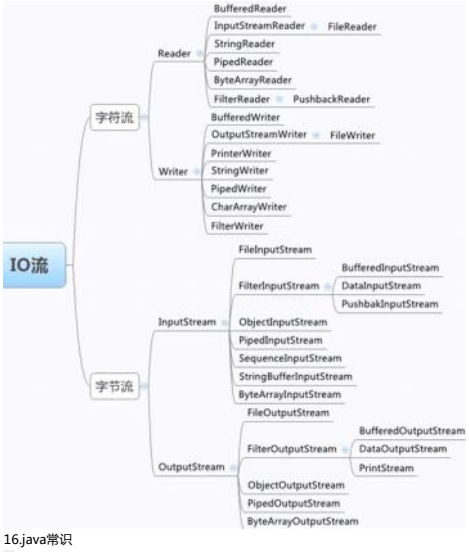
A. FilterReader

B. InputStreamReader

C. PipedReader

D. Reader

答案：D



16.java常识

在程序代码中写的注释太多，会使编译后的程序尺寸变大。

A. 对

B. 错

注释是写给人看的，不是写给及其看的

答案：错

17.final关键字

以下关于final关键字说法错误的是

A. final是java中的修饰符，可以修饰类、接口、抽象类、方法和属性

B. final修饰的类肯定不能被继承

C. final修饰的方法不能被重载

D. final修饰的变量不允许被再次赋值

（1）final用于声明属性、方法和类分别表示不可变属性、方法不可覆盖和类不可被继承
（2）final属性，有两重含义①引用不可变②对象不可变。被final修饰的变量必须被初始化。
（3）final方法，该方法不允许任何子类重写这个方法
（4）final类：当一个类被声明是final时，此类不能被继承，所有方法不能被重写。
答案：AC
（三）数据结构新题
1.

3 有向图中顶点V的度等于其邻接矩阵中第V行中的1的个数。（ ）

正确答案: B 你的答案: A (错误)

正确

错误

答案：错误。
有向图的度=入度+出度，邻接矩阵各行之后是出度，各列之和是入度。
2.知识点：有向图

8 n个顶点的有向图中含有向边的数目最多为（ ）

正确答案: D 你的答案: C (错误)

n-1

n

n(n-1)/2

n(n-1)

n个结点的有向图含有向边的数目最多为n（n-1），无向图含边最多为n（n-1）/2
2.图的深度优先遍历

9 设无向图G 中的边的集合 E={(a ，b) ，(a ，e) ，(a ，c) ，(b ，e) ，(e ，d) ，(d ，f) ，(f ，c)}，则从顶点a出发进行深度优先遍历可以得到的一种顶点序列为（ ）。

正确答案: A 你的答案: 空 (错误)

aedfcb

acfebd

aebcfd

aedfbc

3.哈希表的。。定制方法

11 设哈希表长为14，哈希函数为h(key)=key%11。表中现有数据15、38、61和84，其余位置为空，如果用二次探测再散列处理冲突，则49的位置是

正确答案: D 你的答案: 空 (错误)

8

3

5

9

4.

如下C++程序

```
int i=0x22222222;
char szTest[]="aaaa";  //a的ascii码为0x61
func(I, szTest);     //函数原型为void func(int a,char *sz);
```

请问刚进入func函数时，参数在栈中的形式可能为（左侧为地址，右侧为数据—）

A. 0x0013FCF0 0x61616161
0x0013FCF4 0x22222222
0x0013FCF8 0x00000000

B. 0x0013FCF0 0x22222222
0x0013FCF4 0x0013FCF8
0x0013FCF8 0x61616161

C. 0x0013FCF0 0x22222222
0x0013FCF4 0x61616161
0x0013FCF8 0x00000000

D. 0x0013FCF0 0x0013FCF8
0x0013FCF4 0x22222222
0x0013FCF8 0x61616161

答案：C

①对于x86，栈的增长方向是从大地址到小地址②对于函数调用，参数的入栈顺序是从右到左

③函数调用入栈顺序是右边参数》左边参数》函数返回地址

5.

设树T的度为4,其中度为1,2,3和4的结点个数分别为4,2,1,1,则T中的叶子数为()

A. 5

B. 6

C. 7

D. 8

一棵含有n个结点的树，有n-1个分支，即 $n = 1*4 + 2*2 + 3*1 + 4*1 + 1 = 16$;

又由于 $n = n_0 + n_1 + n_2 + n_3 + n_4 = n_0 + 8$;

$n_0 + 8 = 16$，所有叶子结点个数为8

2018年5月22日 20:28

1.子串和主串

串中任意个数的连续字符组成的子序列称为该串的子串，相应的包含该子串的串称为主串

2.串的比较

穿的比较是通过组成串的字符之间的编码来进行的

3.串是一种特殊的线性表。线性表更关注的是单个元素的操作，比如查找一个元素，插入或者删除一个元素。但串中更多的是查找子串的位置、得到指定位置的子串、替换子串等操作。

4.串的存储结构

和线性表相同，串的存储结构分为两种（1）顺序存储。用一组地址连续的存储单元来存储串中的字符序列，一般用定长的数组来实现，为每个定义串的变量分配一个固定长度的存储区

（2）链式存储

可以在连接串与串的操作时有一定的方便外，总的来说不如顺序存储灵活，性能不如顺序存储结构好。

5.串的模式匹配

子串的定位操作通常称为串的模式匹配，应该算是串中的重要操作。但是朴素的模式匹配效率太低

6.KMP模式匹配

可以大大避免重复遍历的情况

图

2018年5月22日 21:45

1.图中的术语

7.2.4 图的定义与术语总结

术语终于介绍得差不多了，可能有不少同学有些头晕，我们再来整理一下。

图按照有无方向分为**无向图**和**有向图**。无向图由**顶点**和**边**构成，有向图由顶点和**弧**构成。弧有**弧尾**和**弧头**之分。

图按照边或弧的多少分**稀疏图**和**稠密图**。如果任意两个顶点之间都存在边叫**完全图**，有向的叫**有向完全图**。若无重复的边或顶点到自身的边则叫**简单图**。

图中顶点之间有**邻接点**、**依附**的概念。无向图顶点的边数叫做**度**，有向图顶点分为**入度**和**出度**。

图上的边或弧上带**权**则称为**网**。

图中顶点间存在**路径**，两顶点存在路径则说明是**连通**的，如果路径最终回到起始点则称为**环**，当中不重复叫**简单路径**。若任意两顶点都是连通的，则图就是**连通图**，有向则称**强连通图**。图中有子图，若子图极大连通则就是**连通分量**，有向的则称**强连通分量**。

无向图中连通且 n 个顶点 n－1 条边叫**生成树**。有向图中一顶点入度为 0 其余顶点入度为 1 的叫**有向树**。一个有向图由若干棵有向树构成**生成森林**。

2.图的存储结构

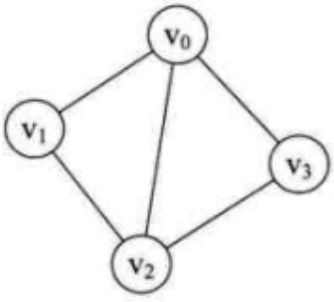
（1）有向图和无向图的邻接矩阵

图的邻接矩阵（**Adjacency Matrix**）存储方式是用两个数组来表示图。一个一维数组存储图中顶点信息，一个二维数组（称为邻接矩阵）存储图中的边或弧的信息。

设图 G 有 n 个顶点，则邻接矩阵是一个 n×n 的方阵，定义为：

$$arc[i][j] = \begin{cases} 1, 若(v_i,v_j) \in E 或 < v_i,v_j > \in E \\ 0, 反之 \end{cases}$$

我们来看一个实例，图 7-4-2 的左图就是一个无向图。



顶点数组：

v_0	v_1	v_2	v_3

边数组：

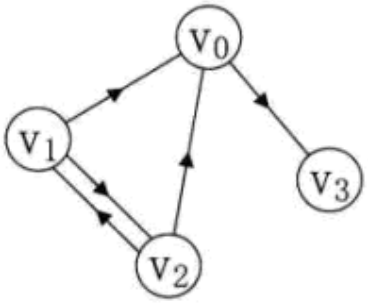
	v_0	v_1	v_2	v_3
v_0	0	1	1	1
v_1	1	0	1	0
v_2	1	1	0	1
v_3	1	0	1	0

v_1的度为2

主对角线

总结：无向图的邻接矩阵是主对角线为零的对称矩阵

我们再来看一个有向图样例，如图 7-4-3 所示的左图。



顶点数组：

v_0	v_1	v_2	v_3

边数组：

	v_0	v_1	v_2	v_3
v_0	0	0	0	1
v_1	1	0	1	0
v_2	1	1	0	0
v_3	0	0	0	0

v_1的出度为2

v_1的入度为1

3.图的遍历

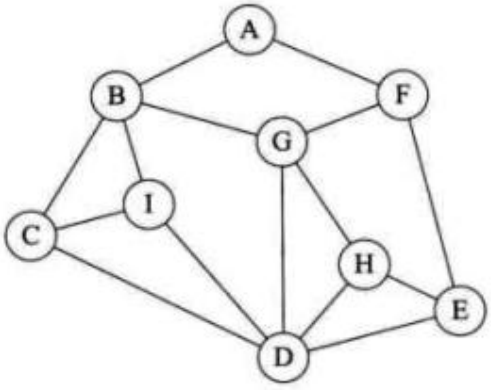
从图中某一顶点出发遍访图中的其余顶点，且使每个顶点仅被访问一次，这个过程叫做图的遍历。

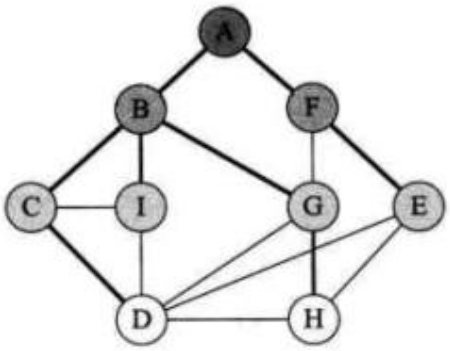
分为两种：深度优先遍历和广度优先遍历

①深度优先遍历

②广度优先遍历

图的广度优先遍历类似于树的层序遍历，也是借助队列来实现的。首先将图A做相应的变化，好像在视觉上发生了变化，但其实顶点和边的关系还是相同的。





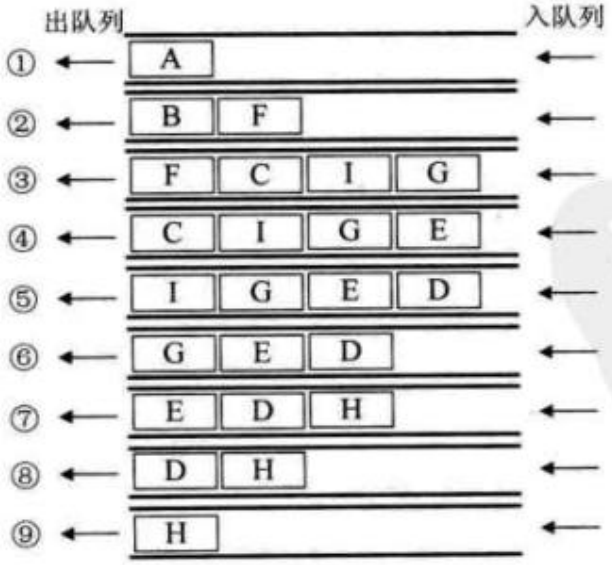


图 7-5-3

深度优先遍历和广度优先遍历的对比

相同点：时间复杂度都是一样的

不同点：对顶点的访问顺序不同。

深度优先更适合目标比较明确，以找到目标为主要目的的情况，而广度优先更适合不断扩大遍历范围是找到相对最优解的情况。

查找

2018年5月22日 22:19

1.二叉排序树

二叉排序树是以链接的方式存储，保持了链接存储结构在执行插入或删除操作时不用移动元素的特点，只要找到合适的插入和删除位置后，仅需要移动链接指针即可，插入的时间性能好。

②对于二叉树的查找，走的是从根结点到要查找的节点的路径，其比较的次数是二叉树的层数，再差也不会超过树的深度。

也就是说，我们希望二叉排序树是比较平衡的，即其深度与完全二叉树相同，均为$\lfloor \log_2 n \rfloor+1$，那么查找的时间复杂也就为 $O(\log n)$，近似于折半查找，事实上，图 8-6-18 的左图也不够平衡，明显的左重右轻。

不平衡的最坏情况就是像图 8-6-18 右图的斜树，查找时间复杂度为 $O(n)$，这等同于顺序查找。

2.散列表查找（哈希表）

散列技术是在记录的存储位置和它的关键字之间建立一个确定的对应关系f，使得每一个关键字key对应一个存储位置f（key）。查找是根据这个确定的对应关系找到给定值key的映射f（key），若查找集合中存在这个记录，则必定在f（key）的位置上。f称为散列函数，又称为哈希函数，采用散列技术将记录存储在一块连续的存储空间中，这块连续的存储空间称为散列表或哈希表。关键字对应的记录存储位置称为散列地址。

2.散列技术最适合求解的问题是查找与定值相等的记录。

3.冲突：

①两个关键字key1不等于key2，但是存在f（key1）=f（key2），这种现象称为冲突，其中key1和key2称为这个散列函数的同义词。出现冲突非常糟糕，这将会造成数据的查找错误。

②处理散列表冲突的方法

（1）开放定址法

就是说一旦发生了冲突，就去寻找下一个空的散列地址，只要散列表足够大，空的散列地址总能找到，并将记录存入

这里有三个公式：

①线性探测法

$$f_i(key) = (f(key) + d_i) \ MOD \ m \quad (d_i=1,2,3,\cdots\cdots,m-1)$$

②二次探测法

$f_i(key)=(f(key)+d_i)\ MOD\ m\ (d_i=1^2,-1^2,2^2,-2^2,\cdots,q^2,-q^2,q\leqslant m/2)$

③随机探测法

$f_i(key)=(f(key)+d_i)\ MOD\ m\ (d_i是一个随机数列)$

（2）再散列函数法

以买房子来举例，如果你看房时的选择标准总是以市中心、交通便利、价格适中为指标那这些房子基本等你看上时，都已经被人买了去了，不妨换种思维，可以去买市郊的房子。对于散列表而言，就是事先准备多个散列函数

$f_i(key)=RH_i(key)\quad(i=1,2,\cdots,k)$

RH就是不同的散列函数，每当发生散列地址冲突时，就换一个散列函数计算，相信总有一个会把冲突解决掉

（3）链地址法：不要换表，就在原地操作

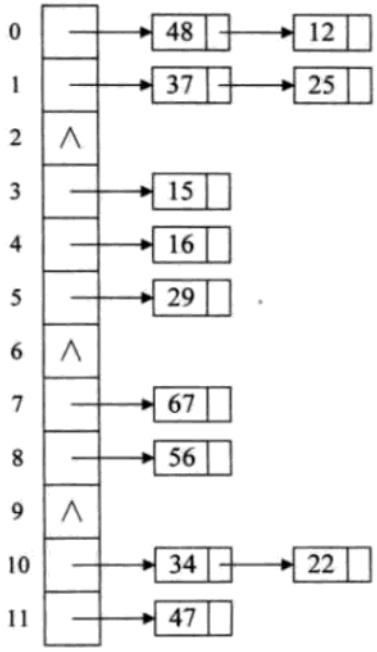


图 8-11-1

链地址法对于可能会造成很多冲突的散列函数来说，提供了绝不会出现找不到地址的保障。当然，这也就带来了查找时需要遍历单链表的性能损耗。

（4）公共溢出区法

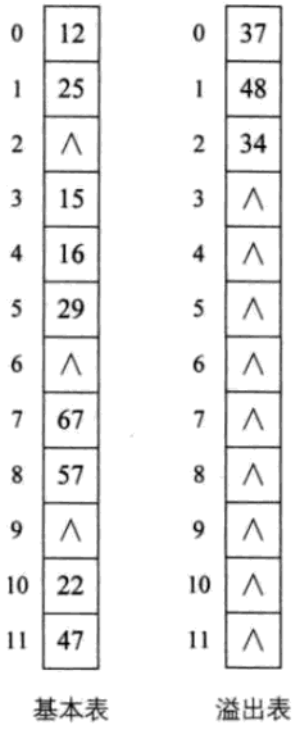
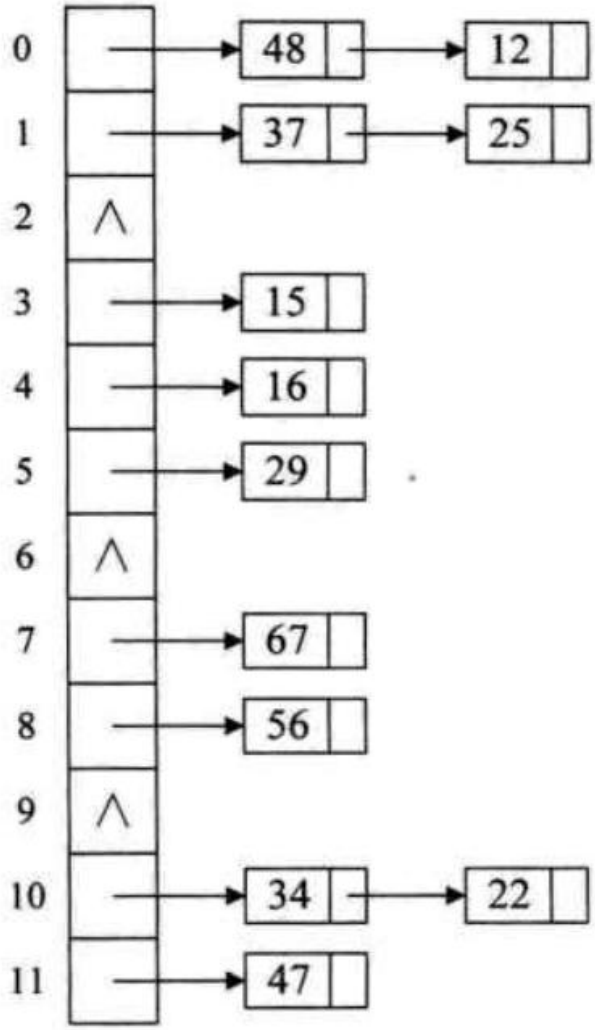


图 8-11-2

在查找时，对给定值通过散列函数计算出散列地址后，先与基本表的相应位置进行比对，如果相等，则查找成功；如果不相等，则到溢出表去进行顺序查找。如果相对于基本表而言，有冲突的数据很少的情况下，公共溢出区的结构对查找性能来说还是非常高的。



③散列表查找的性能分析

散列表查找是所有介绍的查找中效率最高的了，它的时间复杂度是O（1）

2018.05.24

2018年5月24日 15:04

（一）Java部分

1.几种常用但是不会的方法

设有下面两个赋值语句：

```
a = Integer.parseInt("1024");
b = Integer.valueOf("1024").intValue();
```

下述说法正确的是（ ）

A. a是整数类型变量，b是整数类对象。

B. a是整数类对象，b是整数类型变量。

C. a和b都是整数类对象并且它们的值相等。

D. a和b都是整数类型变量并且它们的值相等。

intValue（ ）是把Integer对象类型转化成int的基础数据类型

parseInt（ ）是把String变成int的基础数据类型

ValueOf（ ）是把String类型转化成Integer对象类型（现在的JDK版本支持自动拆箱装箱了）

本题中parseInt得到的是基础数据类型，valueof得到的是装箱数据类型Integer，然后通过valueInt转化成int

答案：D

2.匿名内部类

关于匿名内部类叙述正确的是？（）

A. 匿名内部类可以继承一个基类，不可以实现一个接口

B. 匿名内部类不可以定义构造器

C. 匿名内部类不能用于形参

D. 以上说法都不正确

匿名内部类的创建格式为：new 父类构造器（参数列表）|实现接口（ ）{

//匿名内部类的类体实现

}

①使用匿名内部类，必须继承一个类或实现一个接口

②匿名内部类由于没有名字，所以不能定义构造函数

③匿名内部类不能含有静态成员变量和静态方法

答案：D

3.依赖注入

关于依赖注入，下列选项中说法错误的是（ ）

A. 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

B. 依赖注入使组件之间相互依赖，相互制约

C. 依赖注入提供使用接口编程

D. 依赖注入指对象在使用时动态注入

依赖注入和控制反转是对同一件事情的不同描述，只是描述的角度不同。

①依赖注入是从应用程序的角度描述：应用程序依赖容器创建并注入它所需要的外部资源；是一个重要的面向对象编程的法则来削减计算机程序的耦合问题，降低了组件之间的耦合性。可以使应用程序的配置和依赖性规范与实际的应用代码分开，其中一个特点就是通过文本配置文件进行应用程序组件件相互关系的配置，而不用重新修改并编译具体的代码。

②而控制反转是从容器的角度在描述：容器控制应用程序，由容器反向的向应用程序中注入应用程序所需要的外部资源。

4.HashTable和HashMap

HashMap和HashTable的描述，错误的是？

A. 他们都实现了Map接口。

B.

HashMap非线程安全，在多个线程访问Hashtable时，不需要自己为它的方法实现同步，而HashMap就必须为之提供额外同步。

C. HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。

D. 通过contains方法可以判断一个对象是否存在于HashMap或者Hashtable中。

hashmap	线程不安全	允许有null的键和值	效率高一点、	方法不是Synchronize的要提供外同步	有containsvalue和containsKey方法	HashMap是Java1.2引进的Map interface的一个实现	HashMap是Hashtable的轻量级实现
hashtable	线程安全	不允许有null的键和值	效率稍低、	方法是是Synchronize的	有contains方法方法	、Hashtable继承于Dictionary类	Hashtable比HashMap要旧

1. **关于HashMap的一些说法：**
   a) **HashMap实际上是一个“链表散列”的数据结构，即数组和链表的结合体。HashMap的底层结构是一个数组，数组中的每一项是一条链表。**
   b) **HashMap的实例有俩个参数影响其性能：“初始容量” 和 装填因子。**
   c) **HashMap实现不同步，线程不安全。 HashTable线程安全**
   d) **HashMap中的key-value都是存储在Entry中的。**
   e) **HashMap可以存null键和null值，不保证元素的顺序恒久不变，它的底层使用的是数组和链表，通过hashCode()方法和equals方法保证键的唯一性**
   f) **解决冲突主要有三种方法：定址法，拉链法，再散列法。HashMap是采用拉链法解决哈希冲突的。**

注： 链表法是将相同hash值的对象组成一个链表放在hash值对应的槽位；
用开放定址法解决冲突的做法是：当冲突发生时，使用某种探查(亦称探测)技术在散列表中形成一个探查(测)序列。 沿此序列逐个单元地查找，直到找到给定 的关键字，或者碰到一个开放的地址(即该地址单元为空)为止（若要插入，在探查到开放的地址，则可将待插入的新结点存人该地址单元）。
拉链法解决冲突的做法是： 将所有关键字为同义词的结点链接在同一个单链表中 。若选定的散列表长度为m，则可将散列表定义为一个由m个头指针组成的指针数 组T[0..m-1]。凡是散列地址为i的结点，均插入到以T[i]为头指针的单链表中。T中各分量的初值均应为空指针。在拉链法中，装填因子α可以大于1，但一般均取α≤1。拉链法适合未规定元素的大小。

2. **Hashtable和HashMap的区别：**
   a) **继承不同。**
   public class Hashtable extends Dictionary implements Map
   public class HashMap extends AbstractMap implements Map
   b) **Hashtable中的方法是同步的，而HashMap中的方法在缺省情况下是非同步的。在多线程并发的环境下，可以直接使用Hashtable，但是要使用HashMap的话就要自己增加同步处理了。**
   c) **Hashtable 中， key 和 value 都不允许出现 null 值。 在 HashMap 中， null 可以作为键，这样的键只有一个；可以有一个或多个键所对应的值为 null 。当 get() 方法返回 null 值时，即可以表示 HashMap 中没有该键，也可以表示该键所对应的值为 null 。因此，在 HashMap 中不能由 get() 方法来判断 HashMap 中是否存在某个键， 而应该用 containsKey() 方法来判断。**
   d) **两个遍历方式的内部实现上不同。Hashtable、HashMap都使用了Iterator。而由于历史原因，Hashtable还使用了Enumeration的方式 。**
   e) **哈希值的使用不同，HashTable直接使用对象的hashCode。而HashMap重新计算hash值。**
   f) **Hashtable和HashMap它们两个内部实现方式的数组的初始大小和扩容的方式。HashTable中hash数组默认大小是11，增加的方式是old*2+1。HashMap中hash数组的默认大小是16，而且一定是2的指数。**

注： HashSet子类依靠hashCode()和equal()方法来区分重复元素。
HashSet内部使用Map保存数据，即将HashSet的数据作为Map的key值保存，这也是HashSet中元素不能重复的原因。而Map中保存key值的,会去判断当前Map中是否含有该Key对象，内部是先通过key的hashCode,确定有相同的hashCode之后，再通过equals方法判断是否相同。

5.JVM配置

对于JVM内存配置参数：

> -Xmx10240m -Xms10240m -Xmn5120m -XXSurvivorRatio=3

,其最小内存值和Survivor区总大小分别是（ ）

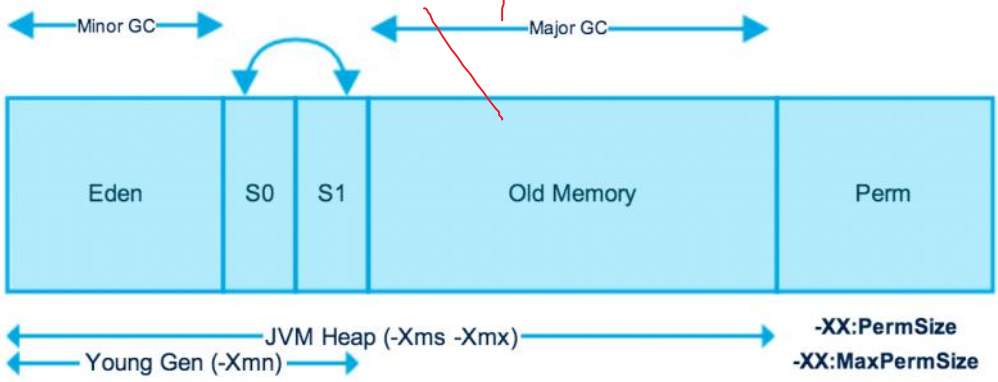
A. 5120m，1024m

B. 5120m，2048m

C. 10240m，1024m

D. 10240m，2048m

-Xmx：最大堆大小
-Xms：初始堆大小
-Xmn：年轻代大小
-XXSurvivorRatio：年轻代中的Eden区与Survivor区的大小比值



6.

Consider the following code:
String s=null;
Which code fragments cause an object of type NullPointerException to be thrown?

A. `if((s!=null)&(s.length()>0))`

B. `if((s!=null)&&(s.length()>0))`

C. `if((s==null)|(s.length()==0))`

D. `if((s==null)||(s.length()==0))`

s为null，因此只要调用了s.length方法就会抛出空指针异常，因此这个题目就是考察if语句的后半段是否会执行。（A）单个与操作的符号用在整数上是按位与，无论前面是否为真，后面必须得执行，因此会抛出异常（C）也是一样
答案：AC
7.接口与抽象类

抽象类

特点:

1.抽象类中可以构造方法

2.抽象类中可以存在普通属性，方法，静态属性和方法。

3.抽象类中可以存在抽象方法。

4.如果一个类中有一个抽象方法，那么当前类一定是抽象类；抽象类中不一定有抽象方法。

5.抽象类中的抽象方法，需要有子类实现，如果子类不实现，则子类也需要定义为抽象的。

接口

1.在接口中只有方法的声明，没有方法体。

2.在接口中只有常量，因为定义的变量，在编译的时候都会默认加上

public static final

3.在接口中的方法，永远都被public来修饰。

4.接口中没有构造方法，也不能实例化接口的对象。

5.接口可以实现多继承

6.接口中定义的方法都需要有实现类来实现，如果实现类不能实现接口中的所有方法

7.则实现类定义为抽象类。

8.sleep（ ）和wait（ ）方法

在JAVA中，下列哪些是Object类的方法（ ）

A. synchronized()

B. wait()

C. notify()

D. notifyAll()

E. sleep()

（A）是java中的一个关键字，不是方法，当它用来修饰一个方法或者一个代码块 的时候能够保证在同一时刻最多只有一个线程执行该代码块

（B）（C）（D）都是Object类的方法

notify（ ）：表示唤醒一个正在等待该对象的线程；

notifyAll（ ）：表示唤醒所有等待该对象的线程

（E）是Thread类的方法

答案：BCD

9.JAVA中的保留字和关键字

true、false、null、sizeof、goto、synchronized 哪些是Java关键字？

A. true

B. false

C. null

D. sizeof

E. goto

F. synchronized

goto和const是保留字也是关键字。

1，Java **关键字**列表 (依字母排序 共50组)：

abstract, assert, boolean, break, byte, case, catch, char, class, **const**（保留关键字）, continue, default, do, double, else, enum, extends, final, finally, float, for, **goto**（保留关键字）, if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while

2，**保留字**列表 (依字母排序 共14组)，Java保留字是指现有Java版本尚未使用，但以后版本可能会作为关键字使用：

byValue, cast, **false**, future, generic, inner, operator, outer, rest, **true**, var, **goto**（保留关键字）, **const**（保留关键字）, **null**

（二）数据结构部分

1.

1 线索二叉树中某结点 R 没有左孩子的充要条件是（ ）。

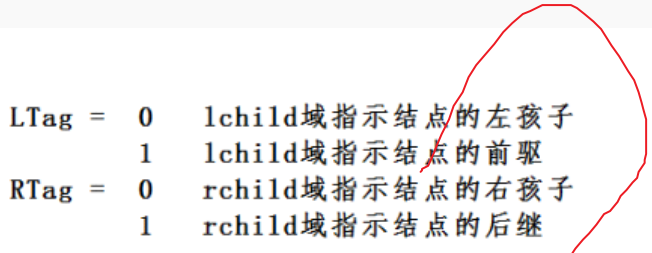
正确答案: C 你的答案: A (错误)

R.lchild=NULL

R.ltag=0

R.ltag=1

R.rchild=NULL



2.二叉树的遍历

6 若一棵二叉树的前序遍历序列和后序遍历序列分别为 1,2,3,4 和 4,3,2,1，则该二叉树的中序遍历序列不会是（ ）

正确答案: C 你的答案: A (错误)

1,2,3,4

2,3,4,1

3,2,4,1

```
3,2,4,1
```

```
4,3,2,1
```

根据题目中的条件就可以知道1在第一层，2在第二层，3在的三层，4在第四层

答案选C

3.线性表

线性表是具有 n 个（ ）的有限序列(n>0)

A. 表元素

B. 字符

C. 数据元素

D. 数据项

就是考定义啊，**线性表是由n个数据元素组成的有限序列**

4.二叉树的总结点数

①

设某种二叉树有如下特点：每个结点要么是叶子结点，要么有2棵子树。假如一棵这样的二叉树中有m（m>0）个叶子结点，那么该二叉树上的结点总数为（ ）。

A. 2m+1

B. 2m-1

C. 2(m-1)

D. 2m

规律：出度为零的结点为m，出度为2的结点数为出度为0的节点数减一，题中以说，此树只有度为0和度为2的节点，所以选B

②

1 一个具有767个节点的完全二叉树，其叶子节点个数为（）

A. 383

B. 384

C. 385

D. 386

完全二叉树：总节点数=度为零+度为2=2*度为零-1=767

叶子结点数为384，答案选B

5.栈（出栈和入栈）

一个栈的输入序列为123、、、、、n，若输出序列的第一个元素是n，输出第i（1<=i<=n）个元素是（ ）

A. 不确定

B. n-i+1

C. i

D. n-i

验证一下就好，2出栈是第（n-1）个，那么第i个元素出栈就是（n-i+1）

答案：B

6.堆（怎么判断是堆）

15 下列关键字序列中，（ ）是堆。

正确答案: B 你的答案: 空 (错误)

16,72,31,23,94,53

16,23,53,31,94,72

94,23,31,72,16,53

16,53,23,94,31,72

把这个序列看成是数组型二叉树，若根节点时i，左子树是2*i，右子树是2*i+1，堆分为最大堆和最小堆
①最大堆中所有父结点都比左子树、右子树大②最小堆中所有父结点都比左子树、右子树小
7.树和森林

设森林F中有三棵树,第一、第二、第三棵树的结点个数分别为M1、M2和M3,与森林F对应的二叉树根结点的右子树上的结点个数是()

A. M1

B. M1+M2

C. M3

D. M2+M3

树或森林与二叉树之间有一个自然的一一对应关系。任何一个森林或者一棵树可唯一对应到一棵二叉树。反之，任何一棵二叉树也能唯一地对应到一个森林或一棵树。
森林转换成二叉树的一般步骤为：
①将森林中的每棵子树转换成相应的二叉树，形成有若干二叉树的森林
②按森林图形中树的先后次序，依次将后边的一棵二叉树作为前边一棵二叉树根节点的右子树，这样整个森林就形成了一棵二叉树，实际上第一棵树的根节点便是生成后的二叉树的根节点。
8.三维数组
①

假设以行优先顺序存储三维数组A[5][6][7],其中元素A[0][0][0]的地址为1100，且每个元素占2个存储单元，则A[4][3][2]的地址是()

A. 1150

B. 1291

C. 1380

D. 1482

参考沐浴星光的潇洒少年

把三维坐标想象成立方体。分配的空间A[5][6][7]表示层高为5、行数为6、列数为7

因为数数的基本单位其实是列，二维坐标是行号列号，虽然平时可能习惯行号列号层号，但是按照二维的规律，那么三维坐标应该是层号行号列号

那么A[4][3][2]中4、3、2分别对应这个点的层数、行号、列号

位置为4*(6*7)+3*7+2=191

每个元素两个存储单元，最终结果为191*2+1100=1482

②

20 查看解析 一个稀疏矩阵A_{m*n}采用三元组形式表示,若把三元组中有关行下标与列下标的值互换,并把m和n的值互换,则就完成了A_{m*n}的转置运算()

正确答案: B 你的答案: B (正确)

对

错

三元组转置：（1）将数组的行列值相互交换（2）将每个三元组的i和j相互交换（3）重排三元组的之间的次序便可实现矩阵的转置

9.二叉树的深度

如果有N个节点用二叉树结构来存储，那么二叉树的最小深度是多少？

A. 以2为底N+1的对数，向下取整

B. 以2为底N的对数，向上取整

C. 以2为底2N的对数，向下取整

D. 以2为底2N+1的对数，向上取整

A成立的条件是[$\log_2$（N+1）]（向上取整，比如2.3取整数3，未说明的情况下一般默认向下取整，如2.3取整数2）。

C的表达式为$\log_2$（2N）=$\log_2$N + $\log_2$2=$\log_2$N + 1。故C为正确答案。

答案：C

10.哈希索引和树的索引

（1）哈希索引仅满足“=”、“IN"和”<=>"查询，不能使用范围查询。因为哈希索引比较的是经常哈希运算之后的哈希值，因此，只能进行等值的过滤，不能基于范围的查找，因为经过哈希算法处理后的哈希值的大小关系，并不能保证与处理前的哈希大小关系对应。

（2）哈希索引无法用来进行数据的排序操作

（3）对于组合索引，哈希索引在计算哈希值的时候是组合索引键合并后再一起计算哈希值，而不是单独计算哈希值，所以通过组合索引的前面一个或几个索引键进行查询的时候，哈希索引无法被使用

（4）哈希索引遇到大量哈希值相等的情况后性能并不一定就会比B-Tree高

2018.05.26

2018年5月26日 15:12

看的是网上总结的java面试基础：

1.说一下多态的表现形式

多态是面向对象程序设计中代码重用的重要机制，有两种表现形式：

（1）重载（overload），是指同一个类中有多个同名的方法，这些方法有不同的参数，因此在编译的时候就可以确定调用哪一种方法，它是一种编译时的多态。

（2）覆盖（override）或重写，子类和父类中某个方法的名字和参数完全相同。①通过子类创建的实例对象调用这个方法时，将调用子类定义的方法，这相当于把父类定义的那个完全相同的方法给覆盖了。②子类只能比父类抛出更少 的异常，或者是抛出父类抛出的异常的子异常。③**子类方法的访问权限只能比父类的更小。**

2.数据的加密模式？加密模式的顺序？

（1）加密模式分为对称式加密和非对称式加密，顺序：传输加密、数据存储加密、数据完整型的鉴别、密钥管理

3.ArrayList和LinkList的区别和联系。

（1）ArrayList实现了基于动态数组的数据结构，由于数组的存储空间是连续的，所以会在内存中开辟一块连续的空间来存储，因此A..支持下标来访问元素，同时索引速度比较快；LinkList是基于链表的数据结构，对数据的索引需要从头开始遍历，因此随机方法的效率较低，但是在插入和删除不需要移动数据，所以插入效率较高。

（2）对于随机访问元素及索引ArrayList的速度比LinkList快

（3）对于插入和删除元素，ArrayList的速度比LinkList慢

（4）ArrayList和LinkList都是线程不安全的

4.基本类型和引用类型的区别

5.接口和抽象类的区别

6.hashMap和HashTable的区别？哪个不安全？为什么?怎样遍历HashMap？

Java.util.Map一共有三个实现类：HashMap、HashTable和TreeMap

比较HashTable和HashMap

（1）HashMap是HashTable的轻量级实现（非线程安全的实现），它们都实现了Map接口，主要区别在于，HashMap允许空（null）键值（key），但要注意，最多只允许一条记录的键值为null，不允许多条记录键值为null，而HashTable一条都不允许。

（2）HashMap将HashTble的contains方法去掉了，改成containsvalue和containsKey，因为contains方法容易引起误解。

（3）HashMap继承自Dictionary类（陈旧），而HashTable是Map interface的一个实现

（4）HashTable是线程安全的，而HashMap不支持线程同步，因此是非线程安全的。在多个线程访问HashTable的时候，不需要开发人员对它进行同步；但是HashMap需要提供额外的同步机制

（5）HashTable使用Eunameration，HashMap使用Iterator

（6）两者性能不会有很大的差异，因为都是使用hash/reharh算法

（7）在HashTable中，hash数组的默认大小是11，增加方式是2*old+1；在HashMap中，hash数组默认的大小是16，而且一定是2的指数。

```
(5)遍历hashmap:两种方式,Map map=new HashMap();
Iterator iter=map.entrySet().iterator();s
Iterator iter=map.keySet().iterator();
```

7.list、Set、和Map接口什么时候用？彼此之间的区别和联系？

（1）List被称为是有序的Collection。它是按照对象进入的顺序保存对象的，所以它能对列表中每一个元素的插入和删除位置进行精确的控制，同时也可以保存重复的对象。LinkList、ArrayList和Vector都实现了List接口。

（2）Set是集合。其最主要的特点就是集合中的元素不能重复，因此存入集合中的元素必须定义equals（）方法确保对象的唯一性。有两个实现类：HashSet和TreeSet（实现了Sorted接口，因此容器中的元素是有序的）

（3）Map。提供了一个从键映射到值的数据结构，用于保存键值对，其中键不能重复，值可以重复。Java类库中提供了多个实现该接口的类：HashMap、TreeMap、LinkedHashMap、WeakHashMap和IndentityHashMap。

8.线程有几种状态？谈谈多线程的问题？线程和同步？

（1）线程通常有5中状态：创建、就绪、运行、阻塞和死亡。

①创建，在生成线程对象，并没有调用该对象的start方法。

②就绪状态：当调用了线程的start方法之后，线程进入就绪状态；当线程运行之后，从等待或者睡眠中回来之后，也会处于就绪状态。

③运行状态：线程调度程序将处于就绪状态的线程设置成为当前的线程，此时线程进入了就绪状态，开始运行run函数中的代码

④阻塞状态：线程正在运行的时候，被暂停，通常是为了等待某个事件的发生之后再继续运行。sleep、suspend、wait方法都可能导致线程的阻塞。

⑤死亡状态。如果一个线程的run方法执行结束或者调用stop方法后，该线程就会死亡。对于已经死亡的线程，无法在使用start方法令其进入就绪。

（2）多线程问题

引用多线程的好处：①易于调度

②提高并发性。通过线程可方便有效的实现并发性。进程可创建多个线程来执行同一程序的不同部分。

③开销小。创建线程比创建进程要快，所需开销很少。

④利于充分发挥多处理器的功能。通过创建多线程，即一个进程中可具有两个或者更多个线程，每个线程在处理器上进行，从而提高应用程序的并发性，使得每个处理器得到充分的运行。

9.进程、线程

（1）进程是程序实体的运行过程，是系统进行资源分配和调度的一个基本单位

（2）线程是进程的一个实体，是CPU调度和分派的基本单位，它是比进程更小的能独立运行的基本单位。

（3）线程自己基本上不拥有系统资源，只拥有在运行过程中必不可少的一点资源（如程序的寄生器，栈），但是线程可以与同属一个进程的其他线程共享进程的所有资源。

有几点需要注意：（1）一个线程只能属于一个进程，但是一个进程可以包含多个线程

（2）资源分配给进程，同一进程的所有线程共享该进程的所有资源

（3）处理机分配给线程，即真正在处理机上运行的是线程

（4）线程在执行过程中，需要协作同步。不同的进程间要利用消息通信的办法实现同步。线程是指在进程内的一个执行单元，也是进程内的可调度实体。

10.sleep（）和wait（）方法的区别和联系

③sleep（）是Thread的静态方法，是线程用来控制自身流程的，它会使线程自动暂停一段时间，而把其他机会让给其他线程，等到计时时间一到，自动苏醒

④wait（）是Object类的方法，用于线程间的通信，这个方法会使拥有该对象锁的进程等待，知道其他线程调用notify（）或者notifyAll（）时才醒来

⑤对锁处理的机制不同。sleep（）不涉及线程间的通信，因此，调用sleep（）方法并不会释放锁；调用wait（）方法后，线程会释放掉它所占用的锁，从而使线程所在的对象中的其他synchronized数据可被其它线程使用。

⑥wait（）方法必须放在同步控制方法和同步语句块中，而sleep（）方法可以放在任何地方使用。

⑦sleep方法必须捕获异常，因为在sleep的过程中，有可能被其他对象调用它的interrupt，产生InterruptedException异常。而wait（）、notify（）、notifyAll（）不需要捕获异常

因为sleep不会释放“锁标志”，容易导致死锁问题的发生，所以，一般不推荐使用sleep（）方法，而推荐使用wait（）方法。

另外还有一种的说法是很好的：

（1）sleep（）方法是线程类（Thread）的静态方法，让调用线程进入睡眠状态，让出执行机会给其它线程，等到休眠时间结束后，线程进入就绪状态和其他线程一起竞争cpu的执行时间。因为sleep（）是static静态的方法，因此不能改变对象锁，当一个同步代码块调用了sleep（）方法，线程虽然进入休眠，但是对象的锁没有释放，其它线程依然无法访问这个对象。

（2）wait（）是Object类的方法，当一个线程执行到wait（）方法时，它就进入到一个和对象相关的等待池，同时释放对象的锁，使得其他线程能够访问，可以用过notify、notifyAll等方法唤醒等待的线程。

11.Thread类的几个重要的方法：

（1）start（）调用该方法开始执行该线程

（2）stop（）调用该方法强制结束该线程的执行

（3）join（），调用该方法等待该线程结束

（4）sleep（）使该线程等待

（5）run（）直接执行run（）是同步的

12.LOCK接口的源码

```
public interface Lock {

    /**
     * Acquires the lock.
     */
    void lock();

    /**
     * Acquires the lock unless the current thread is
     * {@linkplain Thread#interrupt interrupted}.
     */
    void lockInterruptibly() throws InterruptedException;

    /**
     * Acquires the lock only if it is free at the time of invocation.
     */
    boolean tryLock();

    /**
     * Acquires the lock if it is free within the given waiting time and the
     * current thread has not been {@linkplain Thread#interrupt interrupted}.
     */
    boolean tryLock(long time, TimeUnit unit) throws InterruptedException;

    /**
     * Releases the lock.
     */
    void unlock();

}
```

- lock()：获取锁，如果锁被暂用则一直等待
- unlock():释放锁
- tryLock(): 注意返回类型是boolean，如果获取锁的时候锁被占用就返回false，否则返回true
- tryLock(long time, TimeUnit unit)：比起tryLock()就是给了一个时间期限，保证等待参数时间
- lockInterruptibly()：用该锁的获得方式，如果线程在获取锁的阶段进入了等待，那么可以中断此线程，先去做别的事

（2）synchronize与Lock的异同

java中提供了两种锁机制来实现某个共享资源的同步：synchronize和Lock

①synchronize使用Objiect对象本身的wait、notify、notifyAll调度机制；Lock使用Condition实现线程之间的调度，完成synchronized实现的所有功能

二者的区别主要有：

②用法不一样，前者只需要在需要同步的

2018.05.27

2018年5月27日 9:09

1.Java数据库的操作

Java数据库连接（JDBC）在java程序中实现数据库操作的操作，提供了执行sql语句，访问数据库的各种方法，并为各种不同的数据库提供了统一的操作接口，java.sql中包含了JDBC操作数据库的所有的类，通过JDBC访问数据库一般有如下步骤：

（1）加载JDBC驱动器

（2）加载JDBC驱动

（3）建立数据库连接

（4）建立Statement对象或者PreStatement对象

（5）执行SQL语句

（6）访问结果集ResultSet对象

（7）依次将ResultSet、Statement、PrepaerdStatement、Connection对象关闭，释放掉所占用的资源

2.Class.forName的作用是什么

作用是将类加载到JVM中，它会返回一个与带有给定字符串名的类或接口相关联的class对象，并且JVM会加载这个，同时JVM会执行该类的静态代码段。

Java基础知识部分

1.Java的作用域

Java的作用域是由变量来决定的，在Java中，主要有三种变量：

（1）成员变量 ：成员变量的作用范围和类的作用范围是一样的，只要类已经创建，成员变量就开始作用

（2）静态变量：又称为static变量或者（全局变量）可以由类名调用，或者方法名调用

（3）局部变量：作用范围是花括号内部

2.构造函数

构造函数是一类特殊的函数，用来在对象实例化的时候初始化对象的成员变量

（1）构造函数必须与类名相同，且不能有返回值

（2）每个类都可以有构造函数

（3）构造函数不能被继承，因此不能被覆盖，但是可以被重载；构造函数可以有0个、1个或1个以上的参数，当开发人员没有写构造函数的时候，系统会自动生成一个无参的构造函数

（4）构造函数的主要作用是完成对象的初始化

（5）子类可以用Super关键字显示调用父类的构造函数。当父类没有提供无参构造函数的时候，子类必须显示调用父类的构造函数并且必须位于第一行。当父类提供了无

参构造函数的时候，子类可以不显示调用父类的构造函数，此时，系统默认调用父类无参的构造函数。当有父类的时候，在实例化对象的时候必须先实例化父类的构造函数，再实例化子类的构造函数

2018.05.28

2018年5月28日 20:00

1.抽象类和接口的异同？

①相同点

（1）都不能被实例化

（2）实现接口的类或抽象类的子类都只有实现了接口和抽象类中方法后才能被实例化

②不同点

（1）接口中只有定义，其方法不能在接口中实现，只有实现接口的类才能是想接口中定义的方法；而抽象类可以有定义和实现，其方法可以在抽象类中实现。

（2）接口需要实现（implements），抽象类需要实现（extends），一个类可以实现多个接口，但是一个类只能继承一个抽象类，因此使用接口可以间接达到多继承的目的。

2018.05.31

2018年5月31日　20:29

Java部分

1.

java功能性注释：

（1）功能性注释嵌在源程序中，用于说明程序段或语句功能以及数据的状态

（2）可使用空行或缩进，以便很容易区分注释和程序

（3）修改程序也容易修改注释

2.时间复杂度的问题

这个程序时二分查找，所以时间复杂度是O（nlogn）

O(1): 表示算法的运行时间为常量
O(n): 表示该算法是线性算法
O(log2n): 二分查找算法
O(n2): 对数组进行排序的各种简单算法，例如直接插入排序的算法。
O(n3): 做两个n阶矩阵的乘法运算
O(2n): 求具有n个元素集合的所有子集的算法
O(n!): 求具有N个元素的全排列的算法
优<----------<劣
O(1)<O(log2n)<O(n)<O(n2)<O(2n)
时间复杂度按数量级递增排列依次为：常数阶O(1)、对数阶O(log2n)、线性阶O(n)、线性对数阶O(nlog2n)、平方阶O(n2)、立方阶O(n3)、……k次方阶O(nk)、指数阶O(2n)。

3.关于main函数

如下的Java程序

```
public class Test {
    public static void main(String[] args) {
    System.out.println(args[0]);
    }
}
```

若采用命令行"java Test one two three"调用，则程序输出的结果为：

A. Test

B. one

C. two

D. java

答案：B

main函数是程序的入口方法，第一个执行的就是main（ ）。通常来说，要执行一个类的方法，先必须实例化一个类的对象，然后通过对象来调用这个方法。但是由于main是程序的入口方法，此时还没有实例化对象，因此在编写main（ ）方法时就要去不需要实例化对象就可以调用这个方法，所以，main（ ）方法需要被定义被public和static

```
public class Test{
    public static void main(String[ ] args){
        for(int i=0;i<args.length;i++){
            System.out.println(args[i]);
        }
    }
}
```

在控制台下，使用 javac Test.java 指令编译上述程序，使用 java Test arg1 arg2 arg3 指令运行程序，程序运行结果为：

```
arg1
arg2
arg3
```

4.变量的定义

以下的变量定义语句中，**合法**的是（ ）

A. byte=128

B. boolean=null

C. long a=123L

D. double=0.9239d

答案：C

5.scoket通信编程

关于 Socket 通信编程，以下描述错误的是：（ ）

A. 服务器端通过new ServerSocket()创建TCP连接对象

B. 服务器端通过TCP连接对象调用accept()方法创建通信的Socket对象

C. 客户端通过new Socket()方法创建通信的Socket对象

D. 客户端通过new ServerSocket()创建TCP连接对象

答案：D

Scoket套接字，是网络上两个程序通过一个双向的通信连接实现数据的交换，根据连接启动的方式以及本地套接字的要连接的目标，套接字直接的连接过程可以分为三个步骤：服务器监听、客户端请求，连接确认。

Socket套接字

就是源ip地址，目标IP地址，源端口号和目标端口号的组合

服务器端：ServerSocket提供的实例

ServerSocket server= new ServerSocket(端口号)

客户端：Socket提供的实例

Socket soc=new Socket(ip地址，端口号)

6.Java中包的作用

下列说法正确的是

A. java中包的主要作用是实现跨平台功能

B. package语句只能放在import语句后面

C. 包（package）由一组类（class）和接口（interface）组成

D. 可以用#include关键词来标明来自其它包中的类

答案：C

这是一道常见笔试题。

A：java中包的引入的主要原因是java本身跨平台特性的需求，实现跨平台功能的是JVM

B：package语句是Java源文件的第一条语句（若缺省该语句，则指定是无名包）

D：java中无#include关键字，如果想在另一个类里面引用包里面的类，要把名字写全（相当用文件的绝对路径访问）或者用import导入

包的作用

- 1 把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。
- 2 如同文件夹一样，包也采用了树形目录的存储方式。同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。
- 3 包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

package必须放在import的前面

7.PreparedStatement与Statement的区别与联系

关于PreparedStatement与Statement描述错误的是（ ）

A. 一般而言，PreparedStatement比Statement执行效率更高

B. PreparedStatement会预编译SQL语句

C. Statement每次都会解析/编译SQL，确立并优化数据获取路径

D. Statement执行扫描的结果集比PreparedStatement大

1:创建时的区别：

```
Statement statement = conn.createStatement();
PreparedStatement preStatement = conn.prepareStatement(sql);
执行的时候:
ResultSet rSet = statement.executeQuery(sql);
ResultSet pSet = preStatement.executeQuery();
```

由上可以看出，PreparedStatement有预编译的过程，已经绑定sql，之后无论执行多少遍，都不会再去进行编译，
而 statement 不同，如果执行多变，则相应的就要编译多少遍sql，所以从这点看，
preStatement 的效率会比 Statement要高一些

答案：D

8.Java语言的特点

下列不属于Java语言性特点的是

A. Java致力于检查程序在编译和运行时的错误

B. Java能运行虚拟机实现跨平台

C. Java自己操纵内存减少了内存出错的可能性

D. Java还实现了真数组，避免了覆盖数据类型的可能

答案：D

真数组：数组在内存中是一个接着一个线性存放的，通过第一个元素就能访问随后的元素，避免了数据覆盖的可能性，和数据类型覆盖并没有关系。

9.

在运行时，由java解释器自动引入，而不用import语句引入的包是()。

A. java.lang

B. java.system

C. java.io

D. java.util

答案：A

Java.lang包是java语言的核心包，lang是language的缩写
Java.lang包定义了一些基本的类型，包括Integer,String之类的，是java程序必备的包，有解释其自动引入，无需手动引入。System是属于java.lang包下面的
10.Java的反射机制

考虑下面这个简单的例子，让我们看看reflection是如何工作的。

```
import java.lang.reflect.*;
public class DumpMethods{
    public static void main(String[] args) {
        try {
            Class c=Class.forName(args[0]);
            Method m[]=c.getDeclaredMethods();
            for (int i = 0; i < m.length; i++) {
                System.out.println(m[i].toString());
            }
        } catch (Throwable e) {
            System.err.println(e);
        }
    }
}
```

其中"c.getDeclaredMethods"的作用是:

A. 取得类的公有方法对象

B. 取得类的所有公有方法名称

C. 取得类的所有方法对象

D. 以上选项都不正确

答案：D
11.java调试机

下列哪个选项是Java调试器？如果编译器返回程序代码的错误，可以用它对程序进行调试。

A. `java.exe`

B. `javadoc.exe`

C. `jdb.exe`

D. `javaprof.exe`

Java.exe是java虚拟机；
Javadoc.exe是用来制作java文档的
Jdb.exe是java的调试器
Javaprof.exe是剖析工具
12.Servlet的生命周期

下列有关Servlet的生命周期，说法不正确的是？

A. 在创建自己的`Servlet`时候，应该在初始化方法`init()`方法中创建`Servlet`实例

B.
在`Servlet`生命周期的服务阶段，执行`service()`方法，根据用户请求的方法，执行相应的`doGet()`或是`doPost()`方法

C. 在销毁阶段，执行`destroy()`方法后会释放`Servlet` 占用的资源

D. `destroy()`方法仅执行一次，即在服务器停止且卸载`Servlet`时执行该方法

Servlet的生命周期可以分为加载、创建、初始化、处理客户端请求和卸载5个阶段
（1）加载。容器通过类加载器使用Servlet类对应的文件来加载Servlet

（2）创建。通过调用Servlet的构造函数创建一个Servlet实例
（3）初始化。通过Servlet的init（）方法完成初始化工作，这个方法是在Servlet已被创建但向客户端提供请求提供服务之前调用的。需要注意的是，init（）方法只会被调用一次。
（4）处理客户请求。Servlet一旦被创建之后，它就可以为客户端提供服务了，每当有新的客户请求到来时，容器都会创建一个新的线程来处理该请求。接着会调用Servlet的service（）方法来完成客户端的请求，当然service（）方法会根据请求的method属性的不同调用决定是调用doGet（）方法还是doPost（）方法来完成具体的响应。
（5）卸载容器在卸载Servlet之前需要调用destroy（）方法，让Servlet自己释放其占用的系统资源，一旦destroy方法被调用，容器不会向这个Servlet发送任何请求消息。如果容器需要这个Servlet，那么就必须重新创建并初始化一个实例，
destroy（）方法只会被调用一次

13.内部类

> 下列说法正确的是（）？
>
> A.
> 对于局部内部类，只有在方法的局部变量被标记为`final`或局部变量是`effctively final`的，内部类才能使用它们
>
> B. 成员内部类位于外部类内部，可以直接调用外部类的所有方法（静态方法和非静态方法）
>
> C. 由于匿名内部类只能用在方法内部，所以匿名内部类的用法与局部内部类是一致的
>
> D. 静态内部类可以访问外部类的成员变量

答案：AB

14.Java并发

> JDK提供的用于并发编程的同步器有哪些？
>
> A. `Semaphore`
>
> B. `CyclicBarrier`
>
> C. `CountDownLatch`
>
> D. `Counter`

A，Java 并发库 的Semaphore 可以很轻松完成信号量控制，Semaphore可以控制某个资源可被同时访问的个数，通过 acquire() 获取一个许可，如果没有就等待，而 release() 释放一个许可。
B，CyclicBarrier 主要的方法就是一个：await()。await() 方法没被调用一次，计数便会减少1，并阻塞住当前线程。当计数减至0时，阻塞解除，所有在此 CyclicBarrier 上面阻塞的线程开始运行。
C，直译过来就是倒计数(CountDown)门闩(Latch)。倒计数不用说，门闩的意思顾名思义就是阻止前进。在这里就是指 CountDownLatch.await() 方法在倒计数为0之前会阻塞当前线程。
D，Counter不是并发编程的同步器

答案：ABC

15.会话跟踪技术

有关会话跟踪技术描述正确的是（ ）

A. Cookie是Web服务器发送给客户端的一小段信息，客户端请求时，可以读取该信息发送到服务器端

B.
关闭浏览器意味着临时会话ID丢失，但所有与原会话关联的会话数据仍保留在服务器上，直至会话过期

C. 在禁用Cookie时可以使用URL重写技术跟踪会话

D. 隐藏表单域将字段添加到HTML表单并在客户端浏览器中显示

答案：ABC

2018.06.01

2018年6月1日 20:56

1.关于形式参数

> 下列语句正确的是（ ）
>
> A. 形式参数可被视为local variable
>
> B. 形式参数可被所有的字段修饰符修饰
>
> C. 形式参数为方法被调用时，是真正被传递的参数
>
> D. 形式参数不可以是对象

（A）形参是可被视为local variable.形参和局部变量一样不能离开方法。都只有在方法内才能发生作用，也只有在方法中才能发挥作用，不会在方法外可见。

（B）关于形式参数也只能使用final修饰符，其他任何修饰符都会引起编译器错误。但是用这个修饰符也有一定的限制，就是在方法中不能对参数做任何修改。不过一般情况下，一个方法的形参不用final修饰。只有一种特殊的情况：方法内部类。一个方法内的内部类如果使用了这个方法的参数或局部变量的话，这个参数或局部变量应该是final

（C）形参在调用时根据调用者更改，实参则用自身的值更改形参的值，也就是真正给被传递的是实参。

（D）方法的参数列表指定要传递给方法什么样的信息，采用的都是对象的模式。因此，在参数列表中必须指定每个所传递对象的类型及名字，像Java中任何传递对象的场合一样，这里传递的实际上也是引用，并且引用类型必须是正确的。

2.

> 下列代码的输出结果是_____

```
1 boolean b=true?false:true==true?false:true;
2 System.out.println(b);
```

> A. true
>
> B. false
>
> C. null
>
> D. 空字符串

==优先级高于三目运算符，先判断true==true，此时返回false.

布尔类型的表达书转换为：boolean b=true ?false:false

所以最后输出就是false

答案：B

3.关于抽象类

对于abstract声明的类，下面说法正确的是

A. 可以实例化

B. 不可以被继承

C. 子类为abstract

D. 只能被继承

E. 可以被抽象类继承

答案：E

（A）抽象类不能实例化

（B）抽象类可以被继承，子类可以实现抽象方法

（C）子类可以是抽象的，也可是非抽象的

（D）只能被继承说法太肯定了，不正确

（E）抽象类可以被抽象类继承，也可以被非抽象类继承

3.String、StringBuffer、StringBuilder三者之间的联系和区别

①String，大姐，出生于JDK 1.0时代，是不可变字符序列

②StringBuffer，二姐，出生于JDK 1.0时代，是线程安全的可变字符序列

③StringBuilder，小妹，出生于JDK 1.5时代，是非线程安全的可变字符序列。

Java中的String是一个类，而非基本数据类型。string是值传入，不是引用传入。StringBuffer和StringBuilder算是双胞胎，这两者方法没有很大的区别。但是在线程安全性方面，StringBuffer允许多线程进行字符操作，这是因为StringBuffer的很多方法都被synchronized修饰了，而StringBuilder没有。StringBuilder的效率比StringBuffer稍高，考虑到线程安全的问题，StringBuilder是首选。另外，JVM运行程序主要的时间耗费是在创建对象和回收对象上。

4.还是关于抽象方法

选项中哪一行代码可以替换 //add code here 而不产生编译错误

```
1  public abstract class MyClass {
2      public int constInt = 5;
3      //add code here
4      public void method() {
5      }
}
```

A. `public abstract void method(int a);`

B. `consInt=constInt+5;`

C. `public int method();`

D. `public abstract void anotherMethod(){}`

A是抽象方法，抽象类可以包含抽象方法，也可以不包含，实现重载。（√）

B 在类中不能constInt = constInt + 5（×）

C 返回值不能作为重载的依据（×）

D 有方法体的不能作为抽象函数（×）

5.

①

代码片段：

```
byte b1=1,b2=2,b3,b6;
final byte b4=4,b5=6;
b6=b4+b5;
b3=(b1+b2);
System.out.println(b3+b6);
```

关于上面代码片段叙述正确的是（ ）

A. 输出结果：13

B. 语句：b6=b4+b5编译出错

C. 语句：b3=b1+b2编译出错

D. 运行期抛出异常

答案：C

C.

被final修饰的变量是常量，这里的b6=b4+b5可以看成是b6=10；在编译时就已经变为b6=10了

而b1和b2是byte类型，java中进行计算时候将他们提升为int类型，再进行计算，b1+b2计算后已经是int类型，赋值给b3，b3是byte类型，类型不匹配，编译不会通过，需要进行强制转换。

Java中的byte，short，char进行计算时都会提升为int类型。

Java 语言在涉及 byte、short 和 char 类型的运算时，首先会把这些类型的变量值强制转换为 int 类型，然后对 int 类型的值进行计算，最后得到的值也是 int 类型。因此，如果把两个 short 类型的值相加，最后得到的结果是 int 类型；如果把两个 byte 类型的值相加，最后也会得到一个 int 类型的值。如果需要得到 short 类型的结果，就必须显式地把运算结果转换为 short 类型，例如对于语句 short s1 = 1; s1 = s1 + 1，由于在运行时会首先将 s1 转换成 int 类型，因此 s1 + 1 的结果为 int 类型，这样编译器会报错，所以，正确的写法应该 short s1 = 1;s1 = (short)(s1 + 1)。

有一种例外情况。“+=”为 Java 语言规定的运算法，Java 编译器会对其进行特殊处理，因此，语句 short s1 = 1; s1 += 1 能够编译通过。

②

```
byte b1=1,b2=2,b3,b6,b8;
final byte b4=4,b5=6,b7;
b3=(b1+b2);  /*语句1*/
b6=b4+b5;    /*语句2*/
b8=(b1+b4);  /*语句3*/
b7=(b2+b5);  /*语句4*/
System.out.println(b3+b6);
```

下列代码片段中，存在编译错误的语句是()

A. 语句2

B. 语句1

C. 语句3

D. 语句4

答案：BCD

Java表达式转型规则，由高到低转换：

（1）所有的byte、short、char的值将被提升为int型

（2）如果有一个操作数是long型，计算结果是long型

（3）如果一个操作数是float型，计算结果是float型

（4）如果一个操作数是double型，计算结果是double型

（5）被final修饰的变量不会自动改变类型，当2个final修饰相操作时，结果会根据左

边变量的类型而转化

6.Collection接口

实现或继承了Collection接口的是（ ）

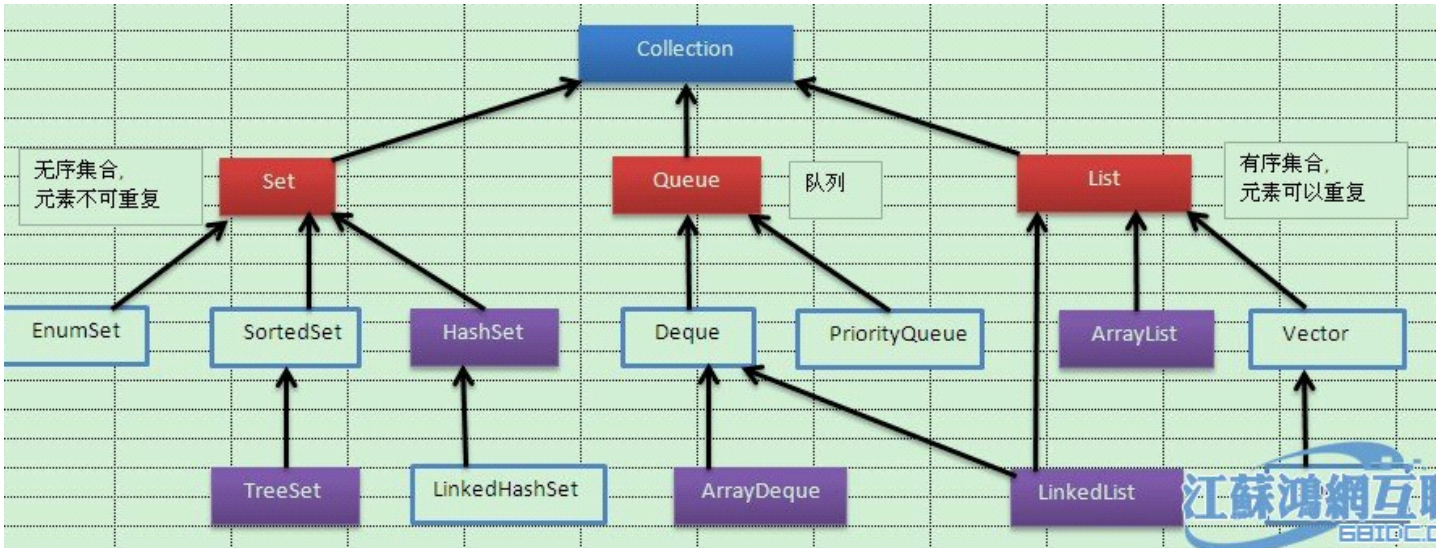
A. Map

B. List

C. Vector

D. Iterator

E. Set



在java.util包中提供了一些集合类，常用的有List、Set、Map类，其中List和Set类继承了Collection接口，这些集合类又称为容器，长度是可变的。注意数组是用来存放基本数据类型的数据，集合是用来存放类对象的引用。

List接口、Set接口、Map接口以及Collection接口的主要特征如下：

（1）Collection接口是List接口和Set接口的父接口，通常情况下不被直接使用

（2）Lsit接口继承了Collection接口，List接口允许存放重复的对象，排序的方式为按照对象的插入顺序

（3）Set接口继承了Collection接口，Set接口不允许存放重复的对象，排序方式为按照自身内部的排序规则

（4）Map接口以键值对（key-value）的形式存放对象，其中键（key）对象不可以重复，value（值）可以重复，排序方式按照自身内部规则

（5）Vector实现了List接口，即间接实现了Collection

（6）Iterator是Java迭代器最简单的实现，没有实现Collection接口

7

Java.Thread的方法resume()负责重新开始被以下哪个方法中断的线程的执行（ ）。

A. stop

B. sleep

C. wait

D. suspend

答案：D

resume与suspended一起使用

wait与notify(notifyAll)一起使用

sleep会让线程暂时不执行

8.PreparedStatement和Statement的区别和联系

以下可以正确获取结果集的有

A.

```
Statement sta=con.createStatement();<br>ResultSet rst=sta.executeQuery("select *
from book");
```

B.

```
Statement sta=con.createStatement("select * from book"); ResultSet
rst=sta.executeQuery();
```

C.
```
PreparedStatement pst=con.prepareStatement();
  ResultSet rst=pst.executeQuery("select * from book");
```

D.
```
PreparedStatement pst=con.prepareStatement("select * from book");
  ResultSet rst=pst.executeQuery();
```

①使用PreparedStatement的好处：

（1）提高可读性和可维护性

（2）最大程度的提高性能。Preparement的第一次执行消耗是很高的，它的性能体现在后面的重复（缓存的作用）

（3）防止SQL的注入

PreparedStatement的使用：

```
String sql = "select distinct loan_type from loan where bank=?";
PreparedStatement preStatement = conn.prepareStatement(sql);
preStatement.setString(1, "Citibank");
ResultSet result = preStatement.executeQuery();
```

Statement的使用：

String sql="select *from users where username=' "+username+" ' AND "+"password=' "+password+ " ' " ;

statement=connection.createStatement（ ）；

resultSet=statement.excuteQuery（sql）;

②PrementStatement和Statement的区别和联系

PrepareStatement继承自Statement

2018.06.04

2018年6月4日 19:38

1.

下列关于包（package）的描述，正确的是（ ）

A. 包（package）是Java中描述操作系统对多个源代码文件组织的一种方式。

B. import语句将所对应的Java源文件拷贝到此处执行。

C. 包（package）是Eclipse组织Java项目特有的一种方式。

D. 定义在同一个包（package）内的类可以不经过import而直接相互使用。

答案：D

（A）为了更好地组织类，Java中提供了包机制。包是类的容器，用于分割类名空间。如果没有指定报名，所有的示例都属于一个默认的无名包。Java中的包一般包含相关的类，java是跨平台的，所以java中包和操作系统没有任何关系，java中的包是用来组织文件的一种虚拟文件系统。

（B）import语句并没有将相应的java源文件拷贝到此处仅仅是引入，告诉编译器有使用外部文件，编译的时候要去读取这个外部文件

（C）Java提供的包机制与IDE没有关系

（D）定义在同一个包（package）内的类可以不经过import而直接相互作用

2.

在开发中使用泛型取代非泛型的数据类型（比如用ArrayList<String>取代ArrayList），程序的运行时性能会变得更好。（ ）

A. 正确

B. 错误

泛型只是在编译的时候保证数据类型的正确性，和运行性能无关

答案：B（错误）

3.

对于文件的描述正确的是（ ）

A. 文本文件是以".txt"为后缀名的文件，其他后缀名的文件是二进制文件。

B. File类是Java中对文件进行读写操作的基本类。

C. 无论文本文件还是二进制文件，读到文件末尾都会抛出EOFException异常。

D. Java中对于文本文件和二进制文件，都可以当作二进制文件进行操作。

答案：D

（A）文件分为文本文件和二进制文件，计算机只认识二进制，所以实际上都是二进制的不同解释方式。文本文件是以不同的编码格式显示的字符，例如，Ascii、Unicode等，window中文本文件有“.txt”,".log",各种编程语言的源码文件，只要能有文本打开的文件都可以算是文本文件，例如“.png”,".bmp"等，计算机中大部分的文件还是二进制文件。

（B）File类是对文件整体或文件属性进行操作的类，例如创建文件、删除文件、查看文件是否存在等功能，不能操作文件内容；文件的内容是用IO流操作的

（C）当输入过程中意外到达文件或流的末尾时，抛出EOFException异常，正常情况下读取到文件末尾时，返回一个特殊值表示文件读取完成，例如read（）返回-1表示文件读取完成。

4

在Java中，对于不再使用的内存资源，如调用完成的方法，“垃圾回收器”会自动将其释放。（ ）

A. 正确

B. 错误

答案：错误

JVM内存可以简单的分为三个区：

（1）堆区：用于存放所有的多想，是线程共享的（注：数组也是对象）

（2）栈区：用于存放基本数据类型的数据和对象的引用，是线程私有的（分为虚拟机栈和本地方法栈）

（3）方法区：用于存放类信息、常量、静态变量、编译后的字节码等，是线程共享的（也被称为非堆）

Java的垃圾回收器主要针对堆区

5.

Java数据库连接库JDBC用到哪种设计模式?

A. 生成器

B. 桥接模式

C. 抽象工厂

D. 单例模式

答案：A

桥接模式

定义：将抽象部分与它的实现部分分离，使它们都可以独立地变化

意图：将抽象与实现解耦

桥接模式所涉及的角色：

1. **Abstraction** ：定义抽象接口，拥有一个Implementor类型的对象引用
2. **RefinedAbstraction** ：扩展Abstraction中的接口定义
3. **Implementor** ：是具体实现的接口，Implementor和RefinedAbstraction接口并不一定完全一致，实际上这两个接口可以完全不一样Implementor提供具体操作方法，而Abstraction提供更高层次的调用
4. **ConcreteImplementor** ：实现Implementor接口，给出具体实现

Jdk中的桥接模式：JDBC

JDBC连接 数据库 的时候，在各个数据库之间进行切换，基本不需要动太多的代码，甚至丝毫不动，原因就是JDBC提供了统一接口，每个数据库提供各自的实现，用一个叫做数据库驱动的程序来桥接就行了

6.

> 下列哪个修饰符可以使在一个类中定义的成员变量只能被同一包中的类访问？
>
> A. `private`
>
> B. 无修饰符
>
> C. `public`
>
> D. `protected`

名称	说明	备注
public	可以被任何类访问	
protected	可以被同一包中的所有类访问 可以被所有子类访问	子类没有在同一包中也可以访问
private	只能够被 当前类的方法访问	
缺省 无访问修饰符	可以被同一包中的所有类访问	如果子类没有在同一个包中，也不能访问

7.

关于sleep()和wait()，以下描述错误的一项是（ ）

A. sleep是线程类（Thread）的方法，wait是Object类的方法；

B. sleep不释放对象锁，wait放弃对象锁

C. sleep暂停线程、但监控状态仍然保持，结束后会自动恢复

D. wait后进入等待锁定池，只有针对此对象发出notify方法后获得对象锁进入运行状态

答案：D

Java中的多线程是一种抢占式的机制，而不是分时式的机制。抢占式的机制是有多个线程处于可运行状态，但只有一个线程在运行。

共同点：

1. 他们都是在多线程的环境下，都可以在程序的调用处阻塞指定的毫秒数，并返回。

2. wait()和sleep()都可以通过interrupt()方法 打断线程的暂停状态 ，从而使线程立刻抛出InterruptedException。

如果线程A希望立即结束线程B，则可以对线程B对应的Thread实例调用interrupt方法。如果此刻线程B正在wait/sleep/join，则线程B会立刻抛出InterruptedException，在catch() {} 中直接return即可安全地结束线程。

需要注意的是，InterruptedException是线程自己从内部抛出的，并不是interrupt()方法抛出的。对某一线程调用 interrupt()时，如果该线程正在执行普通的代码，那么该线程根本就不会抛出InterruptedException。但是，一旦该线程进入到 wait()/sleep()/join()后，就会立刻抛出InterruptedException 。

不同点 ：

1.每个对象都有一个锁来控制同步访问。Synchronized关键字可以和对象的锁交互，来实现线程的同步。

sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可以使用同步控制块或者方法。

2.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而sleep可以在任何地方使用

3.sleep必须捕获异常，而wait，notify和notifyAll不需要捕获异常

4.sleep是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用sleep不会释放对象锁。

5.wait是Object类的方法，对此对象调用wait方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出notify方法（或notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

8.

下列哪个选项是正确计算42度（角度）的余弦值？

A. `double d=Math.cos（42）`

B. `double d=Math.cosine（42）`

C. `double d=Math.cos（Math.toRadians（42））`

D. `double d=Math.cos（Math.toDegrees（42））`

答案：C

Math.cos为计算弧度的余弦值，Math.toRadians函数讲角度转换为弧度

9.

What might cause the current thread to stop executing

A. `An InterruptedException is thrown.`

B. `The thread executes a wait() call.`

C. `The thread constructs a new Thread.`

D. `A thread of higher priority becomes ready.`

E. `The thread executes a waitforID()call on a MediaTracker.`

答案：ABE

2018.06.05

2018年6月5日 11:09

1.异常

请问所有的异常类皆直接继承于哪一个类？（ ）

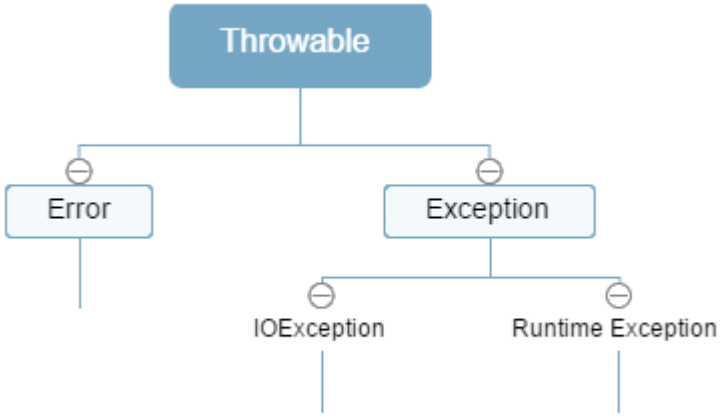
A. java.applet.Applet

B. java.lang.Throwable

C. java.lang.Exception

D. java.lang.Error

答案：C



看好题目，说的是直接继承

2.

若 a 的值为 3 时，下列程序段被执行后， c 的值是多少？（ ）

```
int c = 1;
    if ( a>0 )
            if ( a>3 )
                c = 2;
            else
                c = 3;
        else
            c = 4;
```

A. 1

B. 2

C. 3

D. 4

答案：C

```
public class Demo {
    public static void main(String[] args) {
        int c=1;
        int a=3;
        if (a>0) {
            if (a>3) {
                c=2;
            }else {
                c=3;
            }
        }else {
            c=4;
        }
        System.out.println(c);
    }
```

据说这道题出的和翔似的

3.

Java 语言用以下哪个类来把基本类型数据封装为对象（ ）

A. 包装类

B. Class

C. Math

D. Object

答案：A

Java的数据类型分为两大类：基本类型和引用类型

基本类型只能保存一些常量数据，，引用类型除了可以保存数据，还能提供操作这些数据的功能；为了操作基本类型的数据，java也对它们进行了封装，得到8个类，就是java中的基本类型的封装类，它们是：

八种基本类型： byte、 short、 int、 long、 float、 double、 char、 boolean

对应的包装类：Byte、 Short、 Integer、 Long、 Float、 Double、 Character Boolean

4.forward和redirect的区别和联系

看以下代码：

文件名称：forward.jsp

```
<html>
     <head><title> 跳转  </title> </head>
     <body>
         <jsp:forward page="index.htm"/>
     </body>
 </html>
```

如果运行以上jsp文件，地址栏的内容为

A. http://127.0.0.1:8080/myjsp/forward.jsp

B. http://127.0.0.1:8080/myjsp/index.jsp

C. http://127.0.0.1:8080/myjsp/index.htm

D. http://127.0.0.1:8080/myjsp/forward.htm

答案：A

Servlet中主要有两种跳转方式：forward方式和redirect方式。①forward是服务器内部重定向，服务器直接访问目标地址的URL，将那个URL的响应读取过来，而客户端并不知道，因此在客户端浏览器的地址栏中不会显示转向后的地址，还是原来的地址。由于整个定向的过程用的是同一个Request，因此forward会将Request的信息带到被定向的JSP或Servlet中使用。②Redirct则是客户端的重定向，是完全的跳转，即客户端浏览器会获取跳转后的地址，然后重新发送请求，因此浏览器中会显示跳转后的地址。同时这种方式比forward方式多了一次网络请求，因此其效率要低于forward方式。同时，客户端的重定向可以通过设置特定的HTTP头或写JavaScript脚本实现。

5.

下面有关java的instanceof、?、&、&&说法正确的有？

A. instanceof 可用来判断某个实例变量是否属于某种类的类型。

B. "?:" 三目运算符

C. &在逻辑运算中是非短路逻辑与，在位运算中是按位与

D. && 逻辑运算：逻辑与

答案：ABCD

String str = new String("abc")，"abc"在内存中是怎么分配的？

A. 堆

B. 栈

C. 字符串常量区

D. 寄存器

答案：AC

①字符串常量池

JVM为了减少字符串对象的重复创建，其维护了一个特殊的内存，这段内存称为字符串常量池。

工作原理：当代码中出现创建的字符串对象是，JVM首先会对这份字面量进行检查，如果字符串常量池中存在相同内容的字符串对象的引用则将这个引用返回；否则新的字符串对象被创建，然后将这个引用放入字符串常量池，并返回该引用。

字符串常量池的实现的前提条件是Java中String对象是不可变的，这样可以安全保证多个变量共享同一个对象。

②堆和栈

Java中所有类实例化的对象和数组都存放在堆内存中，无论是成员变量、局部变量还是类变量，它们指向的对象都存储在堆内存中。而栈内存是用来存储局部变量的方法调用的

③寄存器

Java中运行时数据区有一个程序寄存器（又称程序计数器），该寄存器是线程私有。Java中的程序计数器用来记录当前线程中正在执行的质量。如果当前正在执行的方法是本地方法，那么此刻程序计数器的值是undefied

6.

jvm中垃圾回收分为scanvenge gc和full GC，其中full GC触发的条件可能有哪些

A. 栈空间满

B. 年轻代空间满

C. 老年代满

D. 持久代满

E. System.gc()

答案：CDE

1.Scavenge GC[Minor GC]：当新对象生成，并且在Eden区申请失败的时候将会出发Scavenge GC。将会遍历整个Eden区，将活动的对象移到Survivor区，回收不活动的对象。并整理两个Survivor区。

2.Full GC：遍历整个堆，包括年老代和持久代。速度比Scanvenge GC慢很多，所以尽量减少Full GC.

造成Full GC的原因：

1.年老代被写满

2.持久代被写满

3.显示调用System.GC

4.上一个GC后个heap的各域分配策略动态变化。

2018.06.06

2018年6月6日 19:08

1.Swing容器

下列哪一项不属于Swing的顶层容器？（ ）

A. JApplet

B. JTree

C. JDialog

D. JFrame

Swing提供三种顶层容器类：JFrame、JDialog、Japplet

答案：B

2.

访问权限

protected访问权限要小于包访问权限。（ ）

A. 正确

B. 错误

答案：B

	private	default	protected	public
同一类	✓	✓	✓	✓
同一包中的类		✓	✓	✓
子类			✓	✓
其他包中的类				✓

3.

一个文件中的数据要在控制台上显示，首先需要（ ）。

A. 使用标准输出流System.out.println()。

B. 建立文件输出流。

C. 建立文件输入流。

D. 标准输入流System.in.read()。

答案：C

首先把这个文件读进来，然后再输出，所以首先要建立文件输入流，然后再建立标准输出流

4.

```
1 java中关于内存回收的正确说法是
```

A. 程序员必须创建一个线程来释放内存

B. 内存回收程序负责释放无用内存

C. 内存回收程序允许程序员直接释放内存

D. 内存回收程序可以在指定的时间释放内存对象

5.JSP四大作用域

```
1 Web程序中，当前用户上下文信息应该保存在下面哪个对象中（）
```

A. page

B. request

C. session

D. Application

答案：C

JSP四大作用域：page（作用范围最小）、request、session、application（作用范围最大）

①存储在application对象中的属性可以被同一个WEB程序中的所有Servlet和JSP页面访问（属性作用范围最大）

②存储在session对象中的属性可以被属于同一个会话（浏览器打卡直到关闭称为一次会话且在此期间会话不失效）的所有Servlet好热JSP页面访问

③存储在request对象中的属性可以被属于同一个请求的所有Servlet和Jsp页面访问（在有转发的情况先可以跨页面获取属性值），例如使用PageContext.forward和PageContext.include方法连接起来的多个Servlet和JSP页面

④存储在pageContext对象中的属性仅可以被当前JSP页面的当前响应过程中调用的各个组件访问

6.

如下代码的输出结果是什么？

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

A. 0

B. 1

C. 2

D. 编译失败

答案：D

静态变量只能在类的主体中定义，不能再方法中定义

7.什么是Web Service

下面有关webservice的描述，错误的是？

A. Webservice是跨平台，跨语言的远程调用技术

B. Webservice通信机制实质就是json数据交换

C. Webservice采用了soap协议（简单对象协议）进行通信

D. WSDL是用于描述 Web Services 以及如何对它们进行访问

答案：B

Web Service是一种基于网络的分布式模块化组件，它可以将可调用的功能发布到Web上以供应用程序访问。Web Service具有一定的技术规范，因此它能够与其他组件或系统有很好的兼容性。Web Service具有下面一些协议来实现的。

①可扩展可标记语言（XML）,它是实现Web Service的基础，非常适用于网络上传输数据使用

②Web服务描述语言（WSDL）,它是采用XML语言来描述Web Service属性的语言。它将Web Service描述为能够进行信息交换的服务访问点的集合，具体定义了Web Service可以做什么、在哪里以及怎样去调用。

③通用描述、发现与集成服务（UDDI）,它是一种由OASIS指定的规范，主要提供基于Web服务的注册和发现记住，为Web提供3个重要技术支持（1）标准、透明、专

门描述Web服务的机制（2）调用Web服务的机制（3）可以访问Web服务注册中心。它维护了一个Web Service的全球目录，其中信息描述的格式也是基于XML格式的。UDDI的核心组件是UDDI商业注册，它使用XML文档来描述企业及其提供的Web Service

④简单对象存取协议（SOAP），它是Web Service的通信协议。当用户通过UDDI找到对应的WSDL描述符后，可以通过SOAP调用Web服务中的操作。SOAP是基于XML描述的方法调用规范

解析：WebService是跨平台、跨语言的远程调用技术，它的通信机制实质就是xml数据交换；采用了soap（简单对象）协议进行通信

8.

```
1 | 以下说法错误的是（）
```

A. 虚拟机中没有泛型，只有普通类和普通方法

B. 所有泛型类的类型参数在编译时都会被擦除

C. 创建泛型对象时请指明类型，让编译器尽早的做参数检查

D. 泛型的类型擦除机制意味着不能在运行时动态获取List<T>中T的实际类型

答案：D

（1）创建泛型对象的时候，一定要指出类型变量T的具体类型。争取让编译器检查出错误，而不是留给JVM运行的时候抛出不匹配的异常（C对）

（2）JVM如何理解泛型的概念-------类型擦除

事实上，JVM并不知道泛型，所有的泛型在编译阶段就已经被处理成了普通的类和方法。处理方法很简单，叫做类型T的擦除。（A对，B对）

总结：

（1）虚拟机中没有泛型，只有普通类和方法

（2）在编译阶段，所有泛型的类型参数都会被Object或者它们的限定边界来替换（类型擦除）

（3）在继承泛型类型的时候，桥方法的合成是为了避免类型擦除所带来的多态灾难，无论我们如何定义一个泛型的类型，相应的都会有一个原始类型自动提供。原始类型的名字就是类型擦除类型参数的泛型类型的名字

9.

下面哪段程序能够正确的实现了GBK编码字节流到UTF-8编码字节流的转换：

```
byte[] src,dst;
```

A. dst=String.fromBytes(src, "GBK").getBytes("UTF-8")

B. dst=new String(src, "GBK").getBytes("UTF-8")

C. dst=new String("GBK", src).getBytes()

D. dst=String.encode(String.decode(src, "GBK")), "UTF-8")

答案：B

实际操作的步骤就是先解码再编码

用new String（src,"GBK"）解码得到字符串，用getBytes（"UTF-8"得到UTF-8）编码字符数组

10.Servlet层级结构和体系

下面有关servlet的层级结构和常用的类，说法正确的有?

A. GenericServlet类：抽象类，定义一个通用的、独立于底层协议的Servlet。

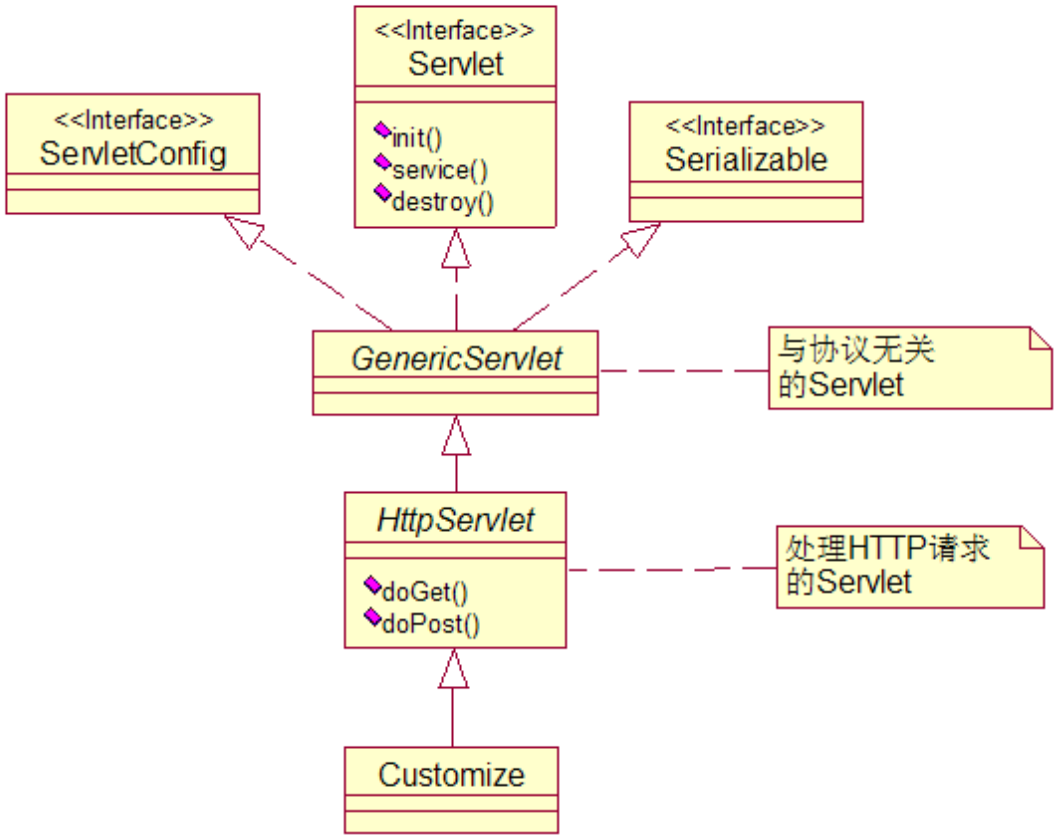
B. 大多数Servlet通过从GenericServlet或HttpServlet类进行扩展来实现

C. ServletConfig接口定义了在Servlet初始化的过程中由Servlet容器传递给Servlet得配置信息对象

D. HttpServletRequest接口扩展ServletRequest接口，为HTTP Servlet提供HTTP请求信息

答案：ABCD

Servlet的体系结构



Java中有关servlet的层级结构和常用的类的描述：

1. GenericServlet类：抽象类，定义一个通用的、独立于底层协议的Servlet。

2. 大多数Servlet通过从GenericServlet或HttpServlet类进行扩展来实现

3. ServletConfig接口定义了在Servlet初始化的过程中由Servlet容器传递给Servlet得配置信息对象

4. HttpServletRequest接口扩展ServletRequest接口，为HTTP Servlet提供HTTP请求信息

HttpServlet是GenericServlet的子类。

GenericServlet是个抽象类，必须给出子类才能实例化。它给 出了设计servlet的一些骨架，定义了servlet生命周期，还有一些得到名字、配置、初始化参数的方法，其设计的是和应用层协议无关的，也就是说 你有可能用非http协议实现它。

HttpServlet是子类，当然就具有GenericServlet的一切特性，还添加了doGet, doPost, doDelete, doPut, doTrace等方法对应处理http协议里的命令的请求响应过程。

一般没有特殊需要，自己写的Servlet都扩展HttpServlet 。

11.抽象类和接口

在Jdk1.7中，下述说法中抽象类与接口的区别正确的有哪些？

A. 抽象类中可以有普通成员变量，接口中没有普通成员变量。

B. 抽象类和接口中都可以包含静态成员常量。

C. 一个类可以实现多个接口，但只能继承一个抽象类

D. 抽象类中可以包含非抽象的普通方法，接口中的方法必须是抽象的，不能有非抽象的普通方法。

答案：ABCD

关于抽象类和接口的方法已经做过很多了为什么还是会错呢？

总结：（1总）如果一个类中包含抽象方法，那么这个类就是抽象类，可以通过把类或类中的某些方法声明为abstract来表示类是一个抽象类；接口就是一个方法的集合，接口中所有的方法都设有方法体，通过关键字interface来实现

（2分）二者的相同点如下：

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类接口或抽象类中的方法后才能被实例化

（3分）二者的不同点如下：

①接口中只有定义，其方法不能在接口中实现，只有实现接口的类才能实现接口中定义的方法；而抽象类的方法可以在抽象类中定义或实现。

②接口实现用户（implements），但抽象类只能被继承（用extends），一个类可以实现多个接口，但是只能继承一个抽象类

③接口中定义成员变量默认为public static final，而且必须为其赋初值；所有的成员方法都是public 、abstract的，而且只能被这两个关键词修饰。抽象类可以柚子街的数据成员变量，亦可以有非抽象的成员方法；抽象类中的成员变量默认是default（本包可见），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重新定义，也可以被重新赋值；抽象类中的抽象方法不能用private、static、synchronized、native等访问修饰符修饰，同时方法必须以分号结尾，而且不带花括号。

12.

java中 String str = "hello world"下列语句错误的是？

A. str+=' a'

B. int strlen = str.length

C. str=100

D. str=str+100

ABCD

2018.06.07

2018年6月7日　　12:54

1.

Java语言中，String s =new String("xyz");创建了几个string object ?

A. 1

B. 2

C. 3

D. 4

答案：B

创建了两个，一个在堆中，第二个在字符串常量池中。如果在Java字符串常量池中已经存在，就只会创建一个。

2.访问修饰符

根据以下代码段，下列说法中正确的是(　)。

```
public class Parent {
  private void m1(){}
    void m2(){}
  protected void m3(){}
public static void m4(){}
  }
```

A. 子类中一定能够继承和覆盖Parent类的m1方法

B. 子类中一定能够继承和覆盖Parent类的m2方法

C. 子类中一定能够继承和覆盖Parent类的m3方法

D. 子类中一定能够继承和覆盖Parent类的m4方法

答案：C

	类内部	本包	子类	外部包
public	√	√	√	√
protected	√	√	√	×
default	√	√	×	×
private	√	×	×	×

Public 和protected都可以作用于子类，但是在多态情况下，静态函数调用时编译和

运行看左边，所以子类父类存在同名静态函数时访问的是父类，子类并不能覆盖父类的方法。

3.

> 关于 Java 线程，下面说法错误的是（ ）。
>
> A. 创建线程的方法有两种：实现Runnable接口和继承Thread类
>
> B. java利用线程使整个系统成为异步
>
> C. 新线程一旦被创建，它将自动开始运行

答案：C

java中创建线程的方法有三种：（1）继承Thread类，重写run（ ）方法

（2）实现Runnable接口，并对对象实例作为参数传递给Thread类的构造方法

（3）实现callable接口，并实现call方法，并且线程执行完毕后会有返回值

（1）和（2）都是调用start（ ）方法启动线程的，然后再JVM虚拟机将此线程放到就绪队列中，有处理机可用时，则执行run方法，这两种方法都重写了run方法，但是没有返回值。

4.

> 在java的一个异常处理中，可以包含多个的语句块是（ ）。
>
> A. try
>
> B. finally
>
> C. throws
>
> D. catch

> 异常处理一般格式：
>
> 捕获异常：
>
> ```
> try{
>     //代码块
> }catch(异常类型，例如: Exception e){
>     //需要抛出的异常，例如: e.printStackTrace();
> }catch(异常类型){
>     //需要抛出的异常
> }finally{
>     //必定执行的代码块
> }
> ```
>
> 所以说在一个异常处理中catch语句块是可以多个的，也就是可以抛出多个异常！

5.

Panel 和 Applet 的默认布局管理器是（ ）

A. CardLayout

B. FlowLayout

C. BorderLayout

D. GridLayout

答案：B

6.

Given:

```
1      //point X
2  public class Foo {
3      public static void main(String[] args) throws Exception {
4
5          PrintWriter out = new PrintWriter(
6              new java.io.OutputStreamWriter(System.out), true);
7          out.printIn("Hello");
8      }
9  }
```

Which statement at PointX on line I allows this code to compile and run?

A. import java.io.PrintWriter;

B. include java.io.PrintWriter;

C. import java.io.OutputStreamWriter;

D. include java.io.OutputStreamWriter;

E. no statement is needed.

答案：A

7.

下面的程序 编译运行后，在屏幕上显示的结果是（ ）

```
public class test {
public static void main(String args[]) {
int x,y;
x=5>>2;
y=x>>>2;
System.out.println(y);
}
}
```

A. 0

B. 2

C. 5

D. 80

答案：A

5的二进制是0101。

x=5>>2（>>带符号右移）

将0101右移2位，为：0001。

y=x>>>2（>>>无符号右移，左边空缺补充为0）

将0001右移2位，补0。结果为：0000。

所以得出答案0

8.实例方法

以下叙述正确的是

A. 实例方法可直接调用超类的实例方法

B. 实例方法可直接调用超类的类方法、

C. 实例方法可直接调用子类的实例方法

D. 实例方法可直接调用本类的实例方法

答案：D

9.定义二维数组

以下二维数组声明合法的是（ ）

A. `char[2][3] ch = new char[][]`

B. `char[2][] ch = new char[][3]`

C. `char[][] ch = new char[2][3]`

D. `char[][] ch = new [2]char[3]`

答案：C

定义数组，等号左边不能出现数字，也就是不管数组大小是什么，不能出现在左边

```
// 数据类型[][] 数组名;
int [][] table = new int[2][2];
int [][] table = new int[2][];
int [] table [] = new int[2][2];
int [] table [] = new int[2][];
```

10.

下面那些情况可以终止当前线程的运行？

A. 当一个优先级高的线程进入就绪状态时

B. 抛出一个异常时

C. 当该线程调用sleep()方法时

D. 当创建一个新线程时

答案：B

线程结束的原因有三个：（1）run方法执行完成，线程正常结束

（2）线程抛出一个未捕获的Exception或者Error

（3）直接调用该线程的Stop方法结束线程（不建议使用，容易造成死锁）

11.

```
public class Tester{
public static void main(String[] args){
   Integer var1=new Integer(1);
   Integer var2=var1;
   doSomething(var2);
   System.out.print(var1.intValue());
   System.out.print(var1==var2);
}
public static void doSomething(Integer integer){
    integer=new Integer(2);
    }
}
```

A. 1true

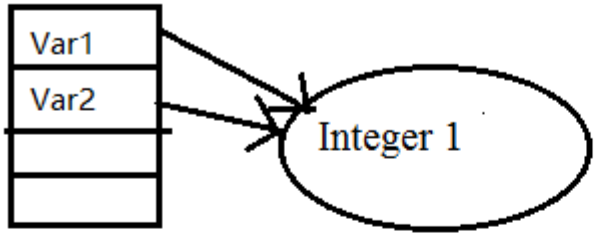
B. 2true

C. 1false

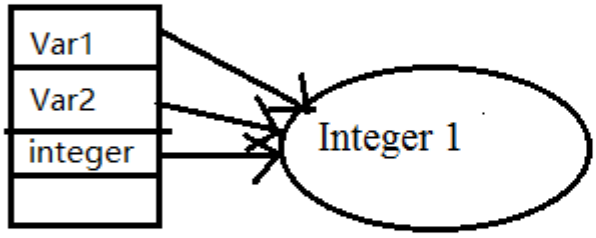
D. 2false

java中引用类型的实参向形参的传递，只是传递的引用，而不是传递的对象本身。

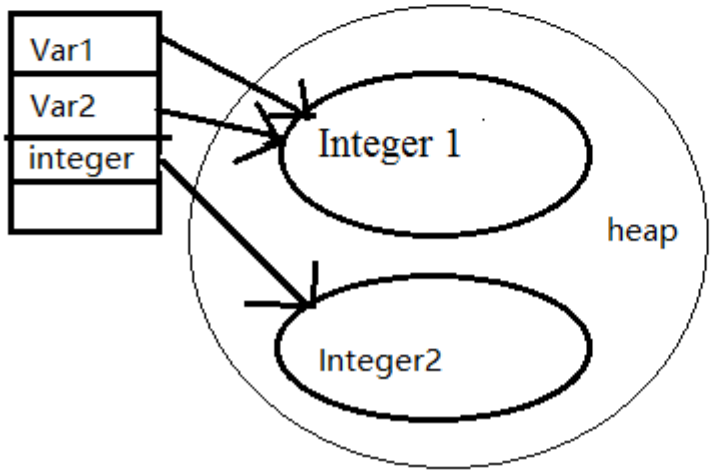
自己画的图，而且没有visio！直接画图画的！



然后调用了都something函数，形参为integer，将Var2的引用给了integer



然后dosomething里将integer引用付给一个新对象



同时说明下equal和==的比较

equal是值的比较，==是引用的比较

答案：D

12.

在Java中，HashMap中是用哪些方法来解决哈希冲突的？

A. 开放地址法

B. 二次哈希法

C. 链地址法

D. 建立一个公共溢出区

答案：C

解决哈希冲突常用的两种方法是：开放定址法和链地址法

开放定址法：当冲突发生时，使用某种探查(亦称探测)技术在散列表中形成一个探查(测)序列。沿此序列逐个单元地查找，直到找到给定 的关键字，或者碰到一个开放的地址(即该地址单元为空)为止（若要插入，在探查到开放的地址，则可将待插入的新结点存人该地址单元）。查找时探查到开放的 地址则表明表中无待查的关键字，即查找失败。

链地址法：将所有关键字为同义词的结点链接在同一个单链表中。若选定的散列表长度为m，则可将散列表定义为一个由m个头指针组成的指针数 组T[0..m-1]。凡是散列地址为i的结点，均插入到以T[i]为头指针的单链表中。T中各分量的初值均应为空指针。

13.

针对以下代码，哪些选项执行后是true的：（ ）

```
class CompareReference{
   public static void main(String [] args){
   float f=42.0f;
   float f1[]=new float[2];
   float f2[]=new float[2];
   float[] f3=f1;
   long x=42;
   f1[0]=42.0f;
  }
}
```

A. f1==f2

B. x==f1[0]

C. f1==f3

D. f2==f1[1]

Java中规定，两个数值进行二元操作时，会有如下转换操作：
①如果两个操作中有一个是double，另一个操作就会转换为double类型
②如果其中一个操作数是float类型，另一个就会转换为float类型
③如果其中一个操作数是long类型，另一个就会转换成long类型
④否则，两个操作数都会转换成int类型
答案：BC

2018.06.08

2018年6月8日 16:45

1.

编译java程序的命令文件是( )

A. java.exe

B. javac.exe

C. applet.exe

答案：B

javac把java文件编译成.class文件；java把".class"文件运行

2.

下面代码的执行结果是：

```
class Chinese{
    private static Chinese objref =new Chinese();
    private Chinese(){}
    public static Chinese getInstance() { return objref; }
}

public class TestChinese {
    public static void main(String [] args) {
    Chinese obj1 = Chinese.getInstance();
    Chinese obj2 = Chinese.getInstance();
    System.out.println(obj1 == obj2);
}
}
```

A. true

B. false

C. TRUE

D. FALSE

正确答案：A

这是饿汉单例模式，在调用Chinese.getInstance（ ）时，进行初始化，由于初始化只会执行一次，所以可以实现单例模式

3.

对于子类的构造函数说明，下列叙述中错误的是（　　）。

A. 子类可以继承父类的构造函数。

B. 子类中调用父类构造函数不可以直接书写父类构造函数，而应该用super();。

C.
用new创建子类的对象时，若子类没有带参构造函数，将先执行父类的无参构造函数，然后再执行自己的构造函数。

D. 子类的构造函数中可以调用其他函数。

答案：A

4.

java中将ISO8859-1字符串转成GB2312编码，语句为 ？

A. new String("ISO8959-1".getBytes("ISO8959-1"),"GB2312")

B. new String(String.getBytes("GB2312") , ISO8859-1)

C. new String(String.getBytes("ISO8859-1"))

D. new String(String.getBytes("GB2312"))

答案：A

5.

有这么一段程序：

```
public class Test{
    public String name="abc";
    public static void main(String[] args){
        Test test=new Test();
        Test testB=new Test();
        System.out.println(test.equals(testB)+","+test.name.equals(testB.name));
    }
}
```

请问以上程序执行的结果是（ ）

A. true,true

B. true,false

C. false,true

D. false,false

答案：C

```
public boolean equals(Object obj) {
        return (this == obj);
    }
```

Object中equals源码如上，没有重写equals时，是直接用==判断，而String中重写了equals方法。

6.

下列不属于java语言鲁棒性特点的是

A. java能检查程序在编译和运行时的错误

B. java能运行虚拟机实现跨平台

C. java自己操纵内存减少了内存出错的可能性

D. java还实现了真数组，避免了覆盖数据的可能

答案：B

程序设计语言中，数组元素在内存中是一个接着一个线性存放的，通过第一个元素就能访问随后的元素，这样的数组称之为“真数组”，实现了真数组为Java语言的健壮性的特点之一。

7.

下列哪项不属于jdk1.6垃圾收集器？

A. Serial收集器

B. parNew收集器

C. CMS收集器

D. G1收集器

答案：D

1.Serial收集器

单线程收集器，收集时会暂停所有工作线程（我们将这件事情称之为Stop The World，下称STW），使用复制收集算法，虚拟机运行在Client模式时的默认新生代收集器。

2.ParNew收集器

ParNew 收集器就是Serial的多线程版本，除了使用多条收集线程外，其余行为包括算法、STW、对象分配规则、回收策略等都与Serial收集器一摸一样。对 应的这种收集器是虚拟机运行在Server模式的默认新生代收集器，在单CPU的环境中，ParNew收集器并不会比Serial收集器有更好的效果。

3.Parallel Scavenge收集器

Parallel Scavenge收集器（下称PS收集器）也是一个多线程收集器，也是使用复制算法，但它的对象分配规则与回收策略都与ParNew收集器有所不同，它是 以吞吐量最大化（即GC时间占总运行时间最小）为目标的收集器实现，它允许较长时间的STW换取总吞吐量最大化。

4.Serial Old收集器

Serial Old是单线程收集器，使用标记 - 整理算法，是老年代的收集器，上面三种都是使用在新生代收集器。

5.Parallel Old收集器

老年代版本吞吐量优先收集器，使用多线程和标记 - 整理算法，JVM 1.6提供，在此之前，新生代使用了PS收集器的话，老年代除Serial Old外别无选择，因为PS无法与CMS收集器配合工作。

6.CMS（Concurrent Mark Sweep）收集器

CMS 是一种以最短停顿时间为目标的收集器，使用CMS并不能达到GC效率最高（总体GC时间最小），但它能尽可能降低GC时服务的停顿时间，这一点对于实时或 者高交互性应用（譬如证券交易）来说至关重要，这类应用对于长时间STW一般是不可容忍的。CMS收集器使用的是标记 - 清除算法，也就是说它在运行期间会 产生空间碎片，所以虚拟机提供了参数开启CMS收集结束后再进行一次内存压缩。

8.

ServletConfig接口默认是哪里实现的？

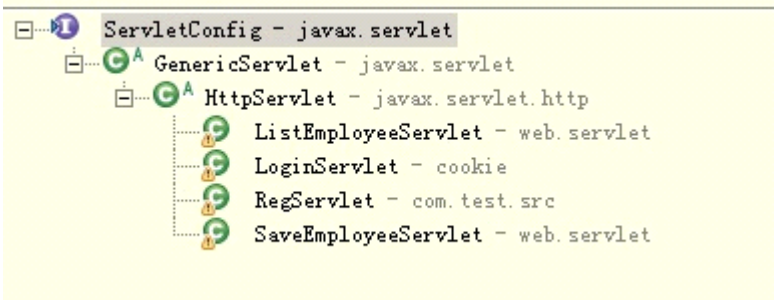
A. Servlet

B. GenericServlet

C. HttpServlet

D. 用户自定义servlet

答案：B



9.

在Struts框架中如果要使用Validation作验证的话，需要使用以下哪个Form？

A. ActionForm

B. ValidatorActionForm

C. ValidatorForm

D. DynaValidatorActionForm

答案：D

10

以下说法错误的是（ ）

A. 其他选项均不正确

B. java线程类优先级相同

C. Thread和Runnable接口没有区别

D. 如果一个类继承了某个类，只能使用Runnable实现线程

B、在java中线程是分优先等级的所以优先级不能相同，错误

C、Thread实现了Runnable接口是一个类不是接口，错误

D、实现多线程的三种方式，一种是继承Thread类使用此方式就不能继承其他的类了，还有两种是实现Runnable接口或者实现Callable接口

答案：BCD

11.

常用的servlet包的名称是？

A. java.servlet

B. javax.servlet

C. servlet.http

D. javax.servlet.http

答案：BD

12

对于线程局部存储TLS(thread local storage)，以下表述正确的是

A. 解决多线程中的对同一变量的访问冲突的一种技术

B. TLS会为每一个线程维护一个和该线程绑定的变量的副本

C. 每一个线程都拥有自己的变量副本，从而也就没有必要对该变量进行同步了

D. Java平台的java.lang.ThreadLocal是TLS技术的一种实现

答案：ABD