Java数据结构与算法：

1. 综述
2. 数组
3. 排序

1. ：综述
2. 数据结构是指数据在计算机内存空间中或磁盘中的组织形式。
3. 正确选择数据结构会使程序的效率大大提高。
4. 数据结构的例子有数组、栈和链表
5. 算法是完成特定任务的过程
6. 在Java中，算法经常通过类的方法实现
7. 一些数据结构的用途是作为程序员的工具，他们帮助执行算法
8. 数据库是指由许多类似的记录组成的数据存储的集合。
9. 一条记录经常表示现实世界中的一个事物，例如一名雇员或者一个汽车零件。
10. 一条记录被分成字段。每个字段都存储了由这个记录所描述事物的一条特性
11. 一个关键字是一条记录中的一个字段，通过他可以对数据执行许多操作。
12. 可以搜索数据库以便找到关键字字段有定值的所有记录。这个值被称为关键字

# ：数组

线性查找：

1. 缺省情况下是线性查找
2. 查找：在有序数组中，当找到一个比待查数据大的数据时就退出查找。
3. 插入：插入一个能处于数组中部的数据项值，这需要将大于这个数据项的所有数据项向后移动。
4. 删除：删除位于数据项中部的数据，向前移动所有比删除数据项下标大的数据项。但有序数组中如果没有找到待删除的数据项时，删除算法可以在中途退出。（break的用法）

有序数组：

1. 优点：查找速度快；在查找频繁的情况下十分有用；
2. 缺点：在插入操作中由于所有的数据都需要移动以腾开空间，所以速度较慢；删除操作较慢，因为数据项必须向前移动来填补已删除的数据项。

无序数组：

1. 优点：插入速度快；
2. 缺点：查找及删除操作较慢

总结：

1.Java的数组是对象，由new操作符创建

2.无序数组可以提供快速的插入，但查找和删除较慢

3.将数组封装到类中可以保护数组不被随意更改

4.类的接口由类用户可访问的方法（有时还有字段）组成。

5.类的接口被设计成使类用户的操作更加简单

6.有序数组可以使用二分查找

7.以B为底A的对数（大概）是在结果小于1之前的用B除A的次数

8.线性查找需要的时间与数组中数据项的个数成正比

9.二分查找需要的时间与数组中数据项的个数的对数成正比

10.大O表示法为表示算法的速度提供了一种方便的方法

11.O（1）级时间的算法是最好的。O(logN)次之，O（N）一般，O（N^2）最差

# 简单排序

冒泡排序：

每次选择出所有未经排序数据项的最大值，放在数组末尾。

1.冒泡排序算法运行起来非常慢，但在概念上它是算法中最简单的。

选择排序：

顾名思义:每次选择出所有未排序数据中的最小值，放在数组开始处。

\*\*排序类型\*\*：插入排序、选择排序、冒泡排序、对象排序

1.简单排序的排序算法都假定了数组作为存储结构

2.排序包括比较数组中数据项的关键字或移动相应的数据项

3.简单排序的复杂度均为O（N的平方）。

4.不变性是指在算法运行时保持不变的条件

5.冒泡排序算法是效率最差的算法，但他最简单

6.插入排序是复杂度为O（n的平方）中应用最多的。

7.如果具有相同关键字的数据项，经过排序他们的顺序保持不变，这样的排序就是稳定的

8.这四种排序方法除了需要初始数组之外，都只需要一个临时变量

# 栈和队列

与数组不同的数据结构类型：

1. 数组与其他数据结构（链表、树）一样，都适用于数据库中作数据记录，他常用于记录那些对应于现实世界的对象和活动的数据。这些结构便于数据的访问：他们易于进行插入删除、删除和查找特定数据项的操作。
2. 栈和队列主要作为构思算法的辅助工具，而不是完全的数据存储工具。这些数据结构的生命周期比那些数据库类型的结构要短得多，在程序操作执行期间他们才能被创建，通常用他们去执行某项特殊的任务，当完成任务之后，他们就会被销毁。

受限访问：

1. 在数组中，若知道数据项的下标，便可以立即访问数据项；若通过顺序搜索数据项，访问到数组中的各数据项。在栈和队列中，访问是受限制的，既在特定时刻只有一个数据项可以被读取或者被删除。栈和队列结构的设计中增强了这种访问。访问其他数据项（理论上）是不允许的。
2. 栈。队列和优先级队列是比数组和其他数据存储结构更为抽象的结构。主要通过接口对栈、队列和优先级队列进行定义，这些接口表明通过他们可以完成的操作，而他们的主要实现机制对用户来说是不可见的。

栈：

1. 栈只允许访问一个数据项：既最后插入的数据项。移除这个数据项后才能访问倒数第二个插入的数据项，以此类推。这种机制在不少编程环境中都很有用。
2. 栈也是那些应用了相当复杂的数据结构算法的便利工具
3. 大部分微处理器运用基于栈的体系结构。当调用一个方法 时，把他的返回地址和参数压入栈，当方法结束返回时，那些数据出栈。栈操作就嵌入在微处理器中。
4. 一些比较老的便携式计算器也用于基于栈的体系结构。
5. 数据项入栈和出栈的时间复杂度都为常数O（1），这就是说，栈操作所耗的时间不依赖于栈中数据项的个数，因此操作时间很短。栈不需要比较和移动操作。

队列：

1. 在计算机（或网络）操作系统中，有各种队列安静的工作着。打印作业在打印队列中等待打印。当在键盘上敲击时，也有一个存储键入内容的队列。同样，如果使用文字处理程序敲击一个键，而计算机又暂时要做其他的事，敲击的内容不会丢失，他会在队列中等待，直到文字处理程序有时间来读取他，利用队列保证了键入内容在处理时其顺序不会改变。

总结：

1.栈、队列和优先级队列是经常用于简化某些程序操作的数据结构

2.在这些数据结构中，只有一个数据项可以被访问

3.栈允许访问最后一个插入的数据项

4.栈中重要的操作是在栈顶插入（压入）一个数据项，以及冲栈顶移除（弹出）一个数据项

5.队列只允许方位第一个插入的数据项

6.队列的重要操作是在队尾插入数据项和在队头移除数据项

7.队列可以实现为循环队列，它基于数组，数组下标可以从数组末端回绕到数组的开始位置。

8.优先级队列允许访问最小（或者有时最大）的数据项

9.优先级队列的重要操作是有序的插入新数据项和移除关键字最小的数据项

10.这些数据结构可以用数组实现，也可以用其他机制（例如链表）来实现

11.普通算术表达式是用中缀表达式表达的，这种命名的原因是操作符在两个操作数的中间

12.在后缀表达式中，操作符跟在两个操作数的后面

13.算术表达式求值通常是先转换成后缀表达式，然后再求后缀表达式的值

14.在中缀表达式转换到后缀表达式以及求后缀表达式的值得过程里，栈都是很有用的工具。

# 链表

1. 类型为Link的next字段仅仅是对另一个Link对象的“引用”，而不是一个对象。
2. 一个引用是一个队某个对象的参照设置，它是一个计算机内存中的对象地址，然而不需要知道它的具体值；只要把它当成一个奇妙的数，它就会告诉你对象在哪里。
3. 所有的引用，在给定的计算机/操作系统中，所有的引用，不管它指向谁，大小都是一样的。因此，对编译器来说，知道这个字段的大小并由此构造出整个Link对象，是没有任何问题的。
4. 在Java语言中，int和double等简单类型的存储于对象的存储是完全不同的。含有简单类型的字段不是引用，而是实实在在的数值。
5. 链表之间的联系是关系而不是位置。在一个数组中，每一项占用一个特定的位置。这个位置可以用一个下标号直接访问。在链表中，寻找一个特定元素的唯一方法就是沿着这个元素的链一直向下寻找。
6. 在表头插入和删除速度很快。仅需要改变一两个引用值，所以花费O(1)的时间。
7. 平均起来，查找、删除和指定链结点后面插入都需要搜索链表中的一半链结点。需要O（N）次比较。在数组中执行这些操作也需要O（N）次比较，但是链表仍然要快一些，因为当插入和删除链结点，链表不需要移动任何东西。增加的效率是很显著的，特别是当复制时间远远大于比较时间的时候。
8. 链表比数组优越的另外一个重要方面是链表需要多少内存就可以用多少内存，并且可以扩展到所有可用内存。数组的大小在它创建的时候就固定了；所以经常由于数组太大而导致效率低下，或者数组太小导致空间溢出。向量是一种可扩展的数组，他可以通过可变长度解决这个问题。但是他经常只允许以固定大小的增量扩展（例如快要溢出的时候，就增加一倍数组容量）。这个解决方案在内存使用效率上来说还是要比链表低。
9. 栈和数组都可以用数组来实现。
10. 在有序链表插入和删除某一项最多需要O（N）次比较，（平均N/2）,因为必须沿着链表上一步一步才能找到正确的位置。然而，可以在O（1）的时间内找到或删除最小值，因为塔总在表头。如果一个应用频繁的存取最小项，且不需要快速的插入，那么有序链表是一个有效的方案选择。优先级队列可以用有序列表来实现。
11. 有序链表可以用于一种高效的排序机制。假设有一个无需数组。如果从这个数组中取出数据，然后一个一个的插入了有序链表，他们自动的按排序排列。他们从有序表中删除，重新放入数组，那么数组就会排序好了。

总结：

1. 链表包含一个linkedList对象和许多Link对象。
2. linkedList对象包含一个引用，这个引用通常叫做first，它指向链表的第一个链结点。
3. 每个Link对象包含数据和一个引用，通常叫做next，它指向链表的下一个节点。
4. Next字段为null值意味着链尾的结尾。
5. 在表头插入链结点需要把新链结点的next字段指向原来的第一个链结点，然后把first指向新链接点。
6. 在表头删除链结点要把first指向first.next.
7. 为了遍历链表，从first开始，然后从一个链结点到下一个链结点，方法是每个链结点的next字段找到下一个链结点。
8. 通过遍历链表可以找到拥有特定值的链结点。一旦找到，可以显示、删除或用其他方式操纵该链结点。
9. 新链结点可以插在某个特定值的链结点的前面或者后面，首先要遍历找到这个链结点。
10. 双端链表在链表中维护一个指向最后一个链结点的引用，他通常和first一样，叫做last
11. 双端链表允许在表尾插入数据项。
12. 抽象数据类型是一种数据存储类，不涉及他的实现。
13. 栈和队列都是ADT，他们既可以用数组实现，又可以用链表实现。
14. 有序链表中，链结点按照关键值升序（有时是降序）排列。
15. 在有序链表中插入需要O(N)的时间，因为必须找到正确的插入点。最小值链结点的删除需要O（1）的时间。
16. 双向链表中，每个链结点包含对前一个链结点的引用，同时有对后一个链结点的引用。双向链表允许反向遍历，并可以从表尾删除。
17. 迭代器是一个引用，他被封装在类对象中，这个引用指向相关联的链表中的链结点。
18. 迭代器方法允许使用者沿链表移动迭代器，并访问当前指示的链结点。
19. 能用迭代器遍历链表，在选定的链结点（或所有链结点）上执行某些操作。