**在**内存中，堆区就是通过new、malloc、realloc分配的内存块，编译器不会负责它们的释放工作，需要用程序区释放。分配方式类似于数据结构中的链表。

在数据结构中堆

是一种经过排序的树形数据结构，每一个节点都有一个值，通常所说堆的数据结构是二叉树，堆的存取是随意的。所以堆在数据结构中通常可以被看做是一棵树的数组对象。而且堆需要满足一下两个性质：堆中某个节点的值总是不大于或不小于其父节点的值；堆总是一棵完全二叉树。

在内存中，栈区可存放函数的参数值、局部变量等，由编译器自动分配和释放，通常在函数执行完后就释放了，其操作方式类似于数据结构中的栈。栈内存分配运算内置于CPU的指令集，效率很高，但是分配的内存量有限。

在数据结构中，栈是限定仅仅在表尾进行插入和删除操作的线性表，把允许插入和删除的一端称之为栈顶，另外一端称之为栈底。特点：后进先出，称之为后进先出线性表。

静态区的全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。程序结束后，由系统释放。

Static是静态关键字，可修饰局部变量、全局变量和函数，被static修饰的变量属于类变量，可以通过类名.变量名直接引用，而不需要new出一个类来

被static修饰的方法属于类方法，可以通过类名.方法名直接引用，而不需要new出一个类来

被static修饰的变量、被static修饰的方法统一属于类的静态资源，是类实例之间共享的，换言之，一处变、处处变。JDK把不同的静态资源放在了不同的类中而不把所有静态资源放在一个类里面

volatile是一个类型修饰符，它是被设计用来修饰被不同线程访问和修改的变量。如果没有volatile，基本上会导致这样的结果：要么无法编写多线程程序，要么编译器失去大量优化的机会。用它声明的类型变量表示可以被某些编译器未知的因素更改。volatile 提醒编译器它后面所定义的变量随时都有可能改变，因此编译后的程序每次需要存储或读取这个变量的时候，都会直接从变量地址中读取数据。如 果没有 volatile 关键字，则编译器可能优化读取和存储，可能暂时使用寄存器中的值，如果这个变量由别的程序更新了的话，将出现不一致的现象。所以遇到这个关键字声明的变量，编译器对访问该变量的代码就不再进行优化，从而可以提供对特殊地址的稳定访问。