1. Java 的内存管理？

1) 内存的分配是由程序完成的，程序员需要通过关键字 new 为每个对象申请内存空间 (基本类型除外)，所有的对象都在堆 (Heap)中分配空间。

2) 对象的释放是由垃圾回收机制决定和执行的，这样做确实简化了程序员的工作。但同时，它也加重了 JVM 的工作。因为，GC 为了能够正确释放对象，GC 必须监控每一个对象的运行状态，包括对象的申请、引用、被引用、赋值等，GC 都需要进行监控。

2. JAVA 的垃圾回收机制

垃圾：没有被其他对象所引用的对象，是无用的。

标记垃圾的算法：1.引用计数法 2. 可达性分析算法

可达性分析：从'GC Roots'的对象作为起点，从'GC Roots'到目标节点如果没有任何引用链则为不可用。

垃圾回收算法

1 标记清除算法：最基础的收集算法，效率高，但是会导致产生大量不连续的碎片

2 复制算法：它将可用内存按容量划分成为大小相等的两块，每次只使用其中的一块，当这一块的内存用完了，就将还存活着的对象复制到另外一块，然后删除已用内存

3 标记整理+清除算法

4 分代回收（Generational Collection）算法

对象分为 Eden、Survivor.from、Survivor.to、Old 四类

详细参考链接：https://www.cnblogs.com/jiangtunan/p/11025521.html

3. Java 反射

JAVA 反射的核心是 JVM 在运行时才动态加载类和调用方法/访问属性，而不是在编译时，并且可以修改属性值

每个类都有一个 Class 对象，包含了与类有关的信息。当编译一个新类时，会产生一个同名的.class 文件，该文件内容保存着 Class 对象。

反射可以提供运行时的类信息，并且这个类可以在运行时才加载进来。

Java 反射几个重要的类：

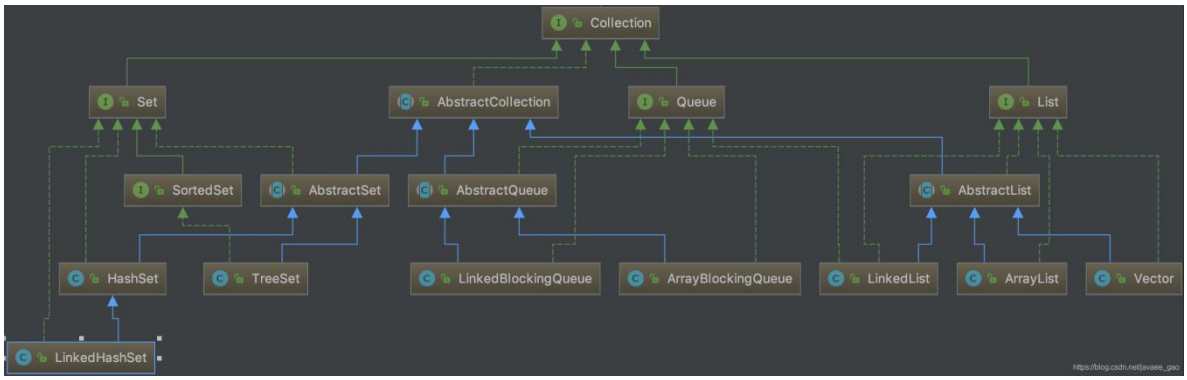
Field ：可以使用 get() 和 set() 方法读取和修改 Field 对象关联的字段；

Method ：可以使用 invoke() 方法调用与 Method 对象关联的方法；

Constructor ：可以用 Constructor 的 newInstance() 创建新的对象。

4. Java 的 Collection

Collection 继承与实现图：



Collection 分为 Set、List、Queue

Set 保证集合内的元素不重复

Set 分为：

·TreeSet: 基于红黑树，支持有序性操作，例如根据一个范围查找元素的操作。

·HashSet：基于 Hash 值映射查找，支持快速查找。

·LinkedHashSet

List 分为：

·ArrayList：基于动态数组实现，支持随机访问

·Vector：和 ArrayList 类似，但它是线程安全的

·LinkedList：基于双向链表实现，只能顺序访问，可以用作栈，队列，双向队列

Queue 分为：

LinkedList/PriorityQueue：

5. Java 的 Map

Map 分为 TreeMap、HashMap、HashTable、LinkedHashMap

1. HashTable

HashTable 的大致实现：

```
public class Hashtable<K,V>
    extends Dictionary<K,V>
    implements Map<K,V>, Cloneable, java.io.Serializable{
    private transient HashtableEntry<?,?>[] table;
    private transient int count;
    ...
```

内部静态类：HashtableEntry，为存储的节点

本质是链表,所以 HashTable 类似于桶状数组，拉链法解决冲突

```
    private static class HashtableEntry<K,V> implements Map.Entry<K,V> {
        final int hash; //哈希值
        final K key;
        V value;
```

```
        HashtableEntry<K,V> next;
    }
}
```

2. HashMap

大致实现:

```
public class HashMap<K,V> extends AbstractMap<K,V> implements Map<K,V>, Cloneable, Serializable{
    transient Entry[] table;
    ....
    内部静态类 Entry<K,V>
    同 HashTable 一样同样使用拉链法，但是使用的是链表的头插法
    static class Entry<K,V> implements Map.Entry<K,V> {
        final K key;
        V value;
        Entry<K,V> next;
        int hash;
    }
```

3. HashMap Vs. HashTable

1.HashMap 是支持 null 键和 null 值的，而 HashTable 在遇到 null 时，会抛出 NullPointerException 异常。

2.我们说 HashTable 是同步的，HashMap 不是，也就是说 HashTable 在多线程使用的情况下，不需要做额外的同步，而 HashMap 则不行。

3.如果你不需要线程安全，那么使用 HashMap，如果需要线程安全，那么使用 ConcurrentHashMap。HashTable 已经被淘汰了，不要在新的代码中再使用它。

4.HashMap 和 HashTable 都是基于哈希表来实现键值映射的工具类。

6.Java 中**接口和类的区别？**

1) 抽象类表示“是一个（IS-A）”关系的抽象，接口表示“能（CAN-DO）”关系的抽象，接口是一组方法声明的集合，是行为的抽象

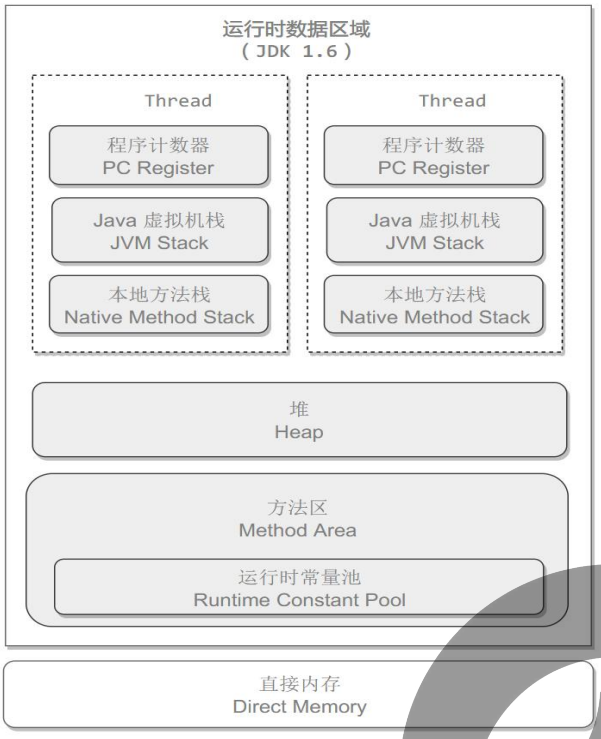
2) 接口不能实例化，类可以实例化

3) 接口没有构造函数，类有构造函数

4) 接口不能进行运算符重载，类可以运算符重载

5) 接口的成员没有任何修饰符，成员都是公共的，类的成员有修饰符

7.JAVA 虚拟机



Java 虚拟机（JVM）是 Java Virtual Machine 的缩写，它是一个虚构出来的计算机，用来实现跨平台。一般的高级语言如果要在不同的平台上运行，至少需要编译成不同的目标代码。而引入 Java 语言虚拟机后，Java 语言在不同平台上运行时不需要重新编译。Java 语言使用模式 Java 虚拟机屏蔽了与具体平台相关的信息。

· 程序计数器：记录正在执行的虚拟机字节码指令的地址

· Java 虚拟机栈 ：每个 java 方法在执行的同时，会创建一个栈帧用于存储局部变量表，操作数栈，常量池引用

· 本地方法栈：本地方法栈为本地方法服务。

· 堆：所有对象都在这里分配内存，是垃圾收集的主要区域("GC 堆")

· 方法区：用于存放已被载的类信息、常量、静态变量、即时编译器编译后的代码等数据。

· 运行时常量池：Class 文件中的常量池（编译器生成的字面量和符号引用）会在类加载后被放入这个区域。

8.Java 程序从源文件创建到程序运行要经过哪两大步骤？

1) 源文件由编译器编译成字节码（ByteCode）

2) 字节码由 java 虚拟机解释运行。因为 java 程序既要编译同时也要经过 JVM 的解释 运行，所以说 Java 被称为半解释语言

9.Synchronized 同步锁？

Synchronized 同步锁可以修饰一段代码块，可以修饰方法，也可以修饰类。主要分为对象锁和类锁。方法锁属于对象锁。

1) 对象锁也叫实例锁，对应 synchronized 关键字，防止在同一个时刻多个线程访问同一个对象的 synchronized 块。当多个线程访问多个实例时，它们互不干扰，每个对象都拥有自己的锁。

2) 类锁对应的关键字是 static sychronized，是一个全局锁，无论多少个对象共享同一个锁（也可以锁定在该类的 class 上或者是 classloader 对象上），同样是保障同一个时刻多个线程同时访问同一个 synchronized 块，当一个线程在访问时，其他的线程等待。

10.Public，private，protect，default 区别？

Public：被其修饰的类、属性以及方法不仅可以跨类访问，而且允许跨包访问；

Private：被其修饰的类、属性以及方法只能被该类的对象访问，其子类不能访问，更不能允许跨包访问；

Protect：被其修饰的类、属性以及方法只能被类本身的方法及子类访问，即使子类在不同的包中也可以访问；

Default：默认访问模式，该模式下，只允许在同一个包中进行访问。

11.run()方法和 start()方法的区别？

start()方法是真正启动 new 出来的线程的，若直接 run()，其实这个 run()是运行在当前

线程之中的，而不是其自身的线程之中。

12. Java 中的 final 关键字

final 可以应用于类、方法以及变量，一旦你将引用声明作 final，你将不能改变这个引用了，编译器会检查代码，如果你试图将变量再次初始化的话，编译器会报编译错误。不能够对 final 变量再次赋值，final 方法不能被重写，final 类不能被继承。好处是提高了性能，final 变量可以安全的在多线程环境下进行共享，而不需要额外的同步开销。

13. 泛型是什么？可以用在哪些地方？

泛型是指通用自定义数据类型。可以用在类，接口，方法中。

14.EJB (Enterprise JavaBean)

JavaEE 服务器端组件模型，即是把已经编写好的程序（即：类）打包放在服务器上执行。

EJB 容器可以接受三类 EJB：

a) 会话 Bean（Session Beans）；

b) 实体 Bean（Entity Beans）；

c) 消息驱动 Bean（Message Driven Beans，MDBs）。

15. 实体 Bean 和会话 Bean 的区别？

实体 bean 实实在在的东西，比如说一本书，有书名，编号，多少页等等。没有逻辑性。可以被持久化的，好比这本书应该放在图书馆里。

会话 bean 指一次会话。客户与店家的对话想买哪本书，有逻辑性的。

它的逻辑性最终

决定了会从书库里拿到哪一本书。不能持久化。

16 多线程中 sleep()、wait()、yield()、join()、stop()、suspend()？

sleep()：在指定时间内让当前正在执行的线程暂停执行，但不会释放锁，属于 Thread 类；

wait()：使当前线程处于等待状态，会释放当前的锁资源，直到被 notify()或 notifyAll()唤醒，可以不指定时间，属于 Object 类；

yield()：只是使当前线程重新回到就绪态，只能使同优先级或更高优先级的线程有机会执行；

join()：其他线程等待当前线程执行完毕；

stop()：可以停止正在执行的线程，但不安全，会解除线程获取的所有锁；

suspend()：调用时目标线程会挂起，但是却仍然有之前获取的锁，其他线程均不能访问该线程锁定的资源，直到其恢复运行，这样容易发生死锁。

17. 线程实现的方法？

继承 Thread 类或者实现 Runnable 接口。

18. Java 中@Override 的含义？

@Override 是伪代码，表示重写(当然不写也可以)，不过写上有如下好处: 可以当注释用,方便阅读；编译器