面试题详解

基础：

面向对象特征：

封装：把描述一个对象的属性和行为封装成一个类，把具体的业务逻辑封装成一个方法，其次封装的意义还能够有效的保护属性通过访问修饰符私有化属性和公有化方法。

继承：实现代码的复用，所有的子类所共有的行为和属性抽取成一个父类，所有的子类继承该父类可具备父类的属性行为和方法，继承具有单一性和传递性。

多态：多态同一个行为具有多个不同表现形式或形态的能力。是指一个类实例（对象）的相同方法在不同情形有不同表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口

final, finally, finalize 的区别：

final：final可以修饰类，变量，方法，修饰类表示该类不能被继承，修饰方法表示该方法不能被重写，修饰变量表示该变量是一个常量不能被重新赋值

Finally：finally一般作用在try-catch代码块中，在处理异常的时候，通常我们将一定要执行的代码方法finally代码块中，表示不管是否出现异常，该代码块都会执行，一般用来存放一些关闭资源的代码。

Finalize：finalize是一个方法，属于Object类的一个方法，而Object类是所有类的父类，该方法一般由垃圾回收器来调用，当我们调用System的gc()方法的时候，由垃圾回收器调用finalize(),回收垃圾。

int 和 Integer 有什么区别：

1. integer是int的包装类，int是Java中一种基本数据类型。
2. integer变量必须实例化后才能使用，而int变量不需要。
3. integer的默认值是null，int的默认值是0.
4. integer实际是对象的引用，当new一个integer时，实际上生成一个指针指向此对象；而int是直接储存数据。

重载和重写的区别：

重载：就是在同一类中允许同时村咋一个以上同名的方法，只要这个方法参数个数或者类型不同即可

重写：是子类重新定义父类中已经定义的方法，即子类重写父类方法

抽象类和接口有什么区别：

1. 抽象类可以有构造方法，接口不能有构造方法
2. 抽象类中可以有普通成员变量，接口中不能有普通成员变量
3. 抽象类中可以包含静态方法，接口中不能包含静态方法
4. 一个类可以实现多个接口，但是只能继承一个抽象类
5. 接口可以被多重实现，抽象类只能被单一继承
6. 如果抽象类实现接口，则可以把接口中方法映射到抽象类中作为抽象方法而不必实现，而在抽象类的子类中实现接口中的方法

接口和抽象类的相同点：

1. 都可以被继承
2. 都不能被实例化
3. 都可以包含方法声明
4. 派生类必须实现未实现的方法

反射的用途及实现：

用途：Java中反射主要用途是开发各种通用框架，通过反射机制，可以在运行期间获取对象的类型信息，

实现：

1. 通过getClass()方法。
2. 通过Class.forName() 方法。
3. 使用类.class。
4. 通过类加载器实现getClassLoader()。

自定义注解的场景以及实现：

场景：

类属性自动赋值

验证对象属性完整性

代替配置文件功能，像spring中基于注解的配置。

可以生成文档，像java代码中的@see@param等等

HTTP 请求的 GET 与 POST 方式的区别：

1. get把参数包含在url中，post通过request body传递参数，
2. get请求在url中传递的参数是有长度限制的，post没有
3. get在浏览器回退时是无害的，而post会再次提交请求
4. get比post更加不安全，get请求参数会暴露在url中，所以不能用来传递敏感信息
5. get产生一个TCP数据包；而post请求会产生两个TCP数据包

session 与 cookie 区别：

1. cookie数据存放在客户的浏览器上，session数据放在服务器上。
2. cookie不是很安全，别人可以分析存放在本地cookie进行cookie欺骗，考虑到安全的话应该使用session。
3. session会在一定时间内保存在服务器上，当访问增多，就会比较占用服务器性能，如果考虑减轻服务器压力，可以选用cookie。
4. 单个cookie保存数据大小不能超过4k，

JDBC 流程：

1. 注册驱动
2. 获取连接
3. 获取语句的对象
4. 执行sql语句
5. 有结果集的的话处理结果集
6. 关闭资源
7. 处理异常

MVC 设计思想：

MVC英文即Model-View-Controller，是一种软件设计典范，将一个应用分成三层，即模型层，视图层，控制层，

equals 与 == 的区别：

1、==，如果作用于基本数据类型的变量，则是比较其储存的值是否相等，

如果作用于引用类型变量，则比较的是所指向的对象的地址是否一致

2、equals继承object类，比较的是引用类型的变量所指向的对象的地址是否一致。如果对equals方法进行重写的话，比较的是所指向的对象的内容是否一致。

集合：

List 和 Set 区别：

1. List，set都继承自collection接口，但是List集合元素放入有序，元素可重复性，Set集合元素放入无序，元素不可重复，如果有重复的元素会自动覆盖掉。
2. Set查询元素效率低下，但是增删改效率高。
3. List和数组类似，可以动态增长查询快，但是增删改效率低。
4. List可以插入多个null元素，但是set只允许插入一个null元素

List 和 Map 区别：

1. List是存储单列数据的集合，Map是存储双列数据的集合。
2. List中存储的数值是有序的，并且允许重复，Map中存储的数据没有顺序的，其键是不能重复的，值是可以重复的。
3. List继承Collection接口，Map不继承Collection接口

Arraylist 与 LinkedList 区别：

1. arraylist是基于动态数组的，linkedlist是基于双向链表的
2. 对于随机访问，arraylist是优于linkedlist
3. 对于增删改操作，linkedlist优于arraylist
4. Linkedlist比arraylisy更加占用内存，因为linkedlist的节点除了储存数据还存储了引用。

ArrayList 与 Vector 相同点和不同点：

不同：

1. arraylist是线程不安全的，vector是线程安全的，vector类对集合中的元素操作时都加了synchronized来保证线程安全。
2. vector与arraylist扩容并不一样，vector默认扩容是增长一倍的容量，arraylist是增上50%的容量。

相同：

1. Vector与ArrayList本质上都是一个Object[] 数组、
2. ArrayList和Vector都是继承了相同的父类和实现了相同的接口

HashMap 和 Hashtable 的区别：

1. HashMap线程不安全（如果想使用一个HashMap线程安全的，可以使用Collections.synchronizedMap(map)或者ConcurrentHashMap类），Hashtable线程安全。
2. HashMap内部允许一个null键和任意数量的空值，Hashtable不允许空的键或值
3. Hashtable从java语言面世以来就已经存在，而HashMap是在java集合框架诞生作为其中一部分而出现的（jdk1.2）。

HashSet 和 HashMap 区别：

1. HashSet实现Set接口，HashMap实现了Map接口。
2. HashSet仅仅用来存储对象，储存之前要确保重写了HashCode()方法和equals()方法确保一致，HashMap存储键值对，不允许存储重复的键
3. HashSet使用add()方法来向set中添加元素，HashMap使用put()方法来将元素放入map集合中。
4. HashMap中使用键对象来计算hashcode值，HashSet使用成员变量对象来计算hashcode值，对于两个对象来说hashcode可能相同，所以equals()方法用来判断对象的相等性，如果两个对象不同，那么返回false。

HashMap 和 ConcurrentHashMap 的区别：

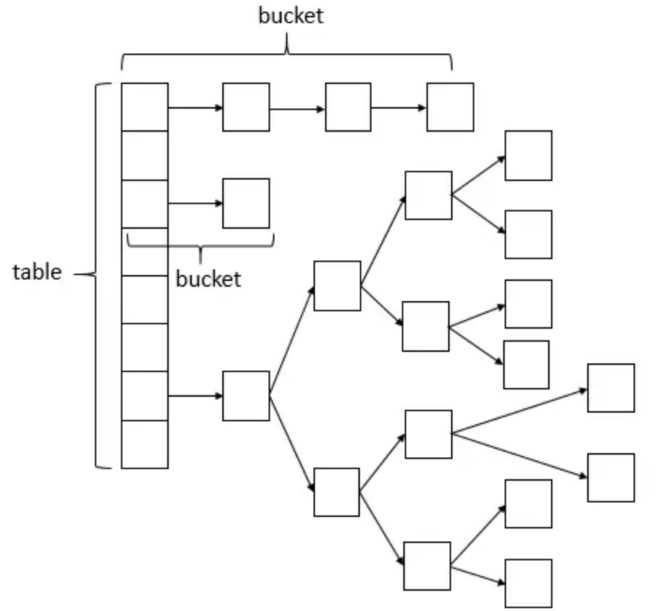
1. HashMap线程不安全，ConcurrenHashMap线程安全
2. ConcurrenHashMap之所以线程安全，是因为它引入了一个分段锁概念，具体可以理解为把一个大的Map拆分成N个小的HashTable，然后根据key.hashCode()来决定把key放到哪个HashTable中。

HashMap 的工作原理：

Hash是基于hashin(散列法)的原理，我们使用put(key,value)存储对象到HashMap中，使用get(key)从HashMap中获取对象。当我们给put()方法传递键和值的时候，先对key调用hashCode方法，来计算hash值，返回的hash值用来找bucket对象，来放entry键值对。

ConcurrentHashMap 的工作原理：

ConcurrentHashMap是java1.5中引入的一个线程安全的支持高并发的HashMap集合类，在jdk1.8中ConcurrentHashMap抛弃了分段锁技术的实现，直接采用了CAS+synchronized保证并发更新的安全性，底层采用了数组加链表加红黑树的存储结构，其包含核心静态内部类Node。



数据结构图