封装

封装，**就是把客观事物封装成抽象的类，**并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。**一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体**。在一个对象内部，**某些代码或某些数据可以是私有的，不能被外界访问。**通过这种方式，对象对内部数据提供了不同级别的保护，以防止程序中无关的部分意外的改变或错误的使用了对象的私有部分。

继承

继承，指可以让**某个类型的对象获得另一个类型的对象的属性和方法**。它支持按级分类的概念。继承是指这样一种能力：**它可以使用现有类的所有功能**，并在无需重新编写原来的类的情况下对这些功能进行扩展。通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。继承的过程，就是从一般到特殊的过程。要**实现继承，可以通过“继承”（Inheritance）和“组合”（Composition）来实现**。继承概念的实现方式有二类：**实现继承与接口继承**。实现继承是指直接使用基类的属性和方法而无需额外编码的能力；接口继承是指仅使用属性和方法的名称、但是子类必须提供实现的能力。

多态

多态，是指**一个类实例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。这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，

它们（那些操作）可以通过相同的方式予以调用。（比如输入的形参可以不同等去实现同一个方法从而得到不同的表现形式）

六大基本原则：SPR, OCP, LSP, DIP, ISP,LoD

单一职责原则SRP(Single Responsibility Principle)

是**指一个类的功能要单一，不能包罗万象**。如同一个人一样，分配的工作不能太多，否则一天到晚虽然忙忙碌碌的，但效率却高不起来。

开放封闭原则OCP(Open－Close Principle)

一个模块在扩展性方面应该是开放的而在更改性方面应该是封闭的。比如：一个网络模块，原来只服务端功能，而现在要加入客户端功能，那么应当在不用修改服务端功能代码的前提下，就能够增加客户端功能的实现代码，这要求在设计之初，**就应当将服务端和客户端分开，公共部分抽象出来。**

里式替换原则LSP(the Liskov Substitution Principle LSP)

子类应当可以替换父类并出现在父类能够出现的任何地方。（比如父类public，子类一定是public）比如：公司搞年度晚会，所有员工可以参加抽奖，那么不管是老员工还是新员工，也不管是总部员工还是外派员工，都应当可以参加抽奖，否则这公司就不和谐了。

依赖倒置原则DIP(the Dependency Inversion Principle DIP)

A.高层次的模块不应该依赖于低层次的模块，**他们都应该依赖于抽象。**

B.抽象不应该依赖于具体实现，具体实现应该依赖于抽象。

具体依赖抽象，上层依赖下层。高层模块就是调用端，底层模块就是具体实现类。（应该让底层模块定义抽象接口并且实现，让高层模块调用抽象接口，而不是直接调用实现类。）

本文摘要：
1.DNS域名解析；
2.建立TCP连接；
3.发送HTTP请求；
4.服务器处理请求；
5.返回响应结果；
6.关闭TCP连接；
7.浏览器解析HTML；
8.浏览器布局渲染；

5类状态码
100 信息状态码，正在请求
200 成功状态码
300 重定向
400 客户端错误
500 服务器错误

1：都是面向对象的语言，都支持封装、继承和多态

2：Java不提供指针来直接访问内存，程序内存更加安全

3：Java的类是单继承的，C++支持多重继承；虽然Java的类不可以多继承，但是接口可以多继承。

4：Java有自动内存管理机制，不需要程序员手动释放无用内存

5: JAVA的应用在高层，C++在中间件和底层

6: JAVA离不开业务逻辑，而C++能够分开业务为JAVA们效劳

自动内存管理
Java程序中所有的对象都是用new操作符建立在内存堆栈上，这个操作符类似于c++的new操作符。下面的语句由一个建立了一个类Read的对象，然后调用该对象的work方法：
Read r＝new Read()；
r.work()；
语句Read r＝new Read()；在堆栈结构上建立了一个Read的实例。Java自动进行无用内存回收操作，不需要程序员进行删除。而c++中必须由程序员释放内存资源，增加了程序设计者的负担。Java中当一个对象不被再用到时，无用内存回收器将给它加上标签以示删除。JAVA里无用内存回收程序是以线程方式在后台运行的，利用空闲时间工作。

面试题1:String s = new String("hello")和String s = "hello"有区别吗?

有区别。前者会创建两个对象，后者创建一个对象。（分析请看String类的特点介绍）

StringBuffer长度和内容可变，String内容和长度不可变。如果使用StringBuffer做字符串的拼接，不会浪费太多的资源。

StringBuffer是同步的，数据安全的，但是效率低； StringBuilder是不同步的，数据不安全，相比于来说，效率高。

1：都是面向对象的语言，都支持封装、继承和多态

2：Java不提供指针来直接访问内存，程序内存更加安全

3：Java的类是单继承的，C++支持多重继承；虽然Java的类不可以多继承，但是接口可以多继承。

4：Java有自动内存管理机制，不需要程序员手动释放无用内存

5: JAVA的应用在高层，C++在中间件和底层

6: JAVA离不开业务逻辑，而C++能够分开业务为JAVA们效劳

7: java言语给开发人员提供了更为简约的语法；取消了指针带来更高的代码质量；完整面向对象，共同的运转机制是其具有自然的可移植性。

8: java 是运转在JVM上的，之所以说它的可移植性强，是由于jvm能够装置到任何的系统

重载： 发生在同一个类中，方法名必须相同，参数类型不同、个数不同、顺序不同，方法返回值和访问修饰符可以不同，发生在编译时。

重写： 发生在父子类中，方法名、参数列表必须相同，返回值范围小于等于父类，抛出的异常范围小于等于父类，访问修饰符范围大于等于父类；如果父类方法访问修饰符为private则子类就不能重写该方法。

- READ UNCOMMITTED（读未提交） 脏读
- READ COMMITTED（读提交） 不可重复读
- REPEATABLE READ（可重复读） 幻读
- SERIALIZABLE（序列化）

ConcurrentHashMap

正是通过Segment分段锁技术，将数据分成一段一段的存储，然后给每一段数据配一把锁，当一个线程占用锁访问其中一个段数据的时候，其他段的数据也能被其他线程访问，能够实现真正的并发访问。

这样结构会使Hash的过程要比普通的HashMap要长，影响性能，但写操作的时候可以只对元素所在的Segment进行加锁即可，不会影响到其他的Segment，ConcurrentHashMap提升了并发能力。

最小生成树

Prim普利姆算法

普利姆算法的核心思想是：从任意一个顶点出发将该顶点加入顶点集U并解锁相邻的边，并找到满足得到最小生成树的那条最小的边，将该边的另一个顶点加入顶点集U并继续解锁相邻的边，直到所有顶点加入顶点集U生成一棵最小生成树。

普利姆算法的执行步骤：

第一步：选择图中某一顶点开始，加入顶点集U，并解锁相邻的所有边

第二步：从解锁边中选择满足条件的一条权值最小的边，将新的顶点加入顶点集U；

第三步：循环第二步，直到所有顶点加入顶点集。

- **应用层：**由用户自己规定，**规定各个应用之间消息传递的形式**等，包括各机互访协议，分布式数据库协议等。常见的应用层协议有HTTP协议和FTP等。
- **表示层：**在满足用户需求的基础上，尽可能的节省传输费用而设置的，比如传输压缩文件，jpeg或者加密文件等格式。
- **会话层：**用于建立和拆除会话。
- **传输层：**负责将来自会话层的消息传递给网络层，**常见的传输层协议有TCP和UDP等协议。**
- **网络层：**规定通信网内的路由选择等方式，建立用户间的信息报传输设施。常见的网络层协议有IP，ICMP以及ARP等协议。

- **数据链路层：**与建立数据传输链路相关。
- **物理层：**规定一些机电性能，也包括工作方式如双工、单工或半双工，建立通信的启动和终止等。

【1】请求报文

请求报文包含四个部分：

- 请求行：包含HTTP版本号、请求方法、URI……
- 请求首部字段
- 请求内容实体
- 空行

【2】响应报文

响应报文包含四部分：

- 状态行：响应状态码、HTTP版本信息
- 响应首部字段
- 响应内容实体
- 空行

HTTPS
因为Http是明文传输，不安全
Https 这个s是secure，更加安全，http + ssl来实现

Volatile
每个线程都分配有单独的处理器缓存
当修改一个线程的变量时，更新主内存，然后向cpu总线发一个修改信号。

秒杀
单一职责。单独业务逻辑，单独数据库
URL动态化，通过前端代码获取url后台校验才能通过
秒杀前，一般按钮都是**置灰**的

LinkedHashMap 按插入顺序排序

- 管道(pipe)
- 有名管道(namedpipe)
- 信号量(semaphore)
- 消息队列(messagequeue)
- 信号(sinal)
- 共享内存(shared memory)
- 套接字(socket)