面试题——（异常、网络、设计模式）

1、throw 和 throws 的区别？

答：

throw则是指抛出的一个具体异常类型

throws是用来声明一个方法可能抛出的所有异常信息

2、final、finally、finalize 有什么区别？

答：

final 是用来修饰类、方法、变量

finally 只能用在 try catch 语法中，表示这段语句最终一定会被执行

finalize 是基础类 java.lang.Object 的一个方法，它的设计目的是保证对象在被垃圾收集前完成特定资源的回收。

3、try-catch-finally 中哪个部分可以省略？

答：try-catch-finally 其中 catch 和 finally 都可以被省略，但是不能同时省略，也就是说有 try 的时候，必须后面跟一个 catch 或者 finally

4、try-catch-finally 中，如果 catch 中 return 了，finally 还会执行吗？

答：一定会， catch 中 return会等finally中的代码执行完之后才会执行

5、常见的异常类有哪些？

答：NullPointerException 空指针异常

ClassNotFoundException 指定类不存在

NumberFormatException 字符串转换为数字异常

IndexOutOfBoundsException 数组下标越界异常

ClassCastException 数据类型转换异常

FileNotFoundException 文件未找到异常

NoSuchMethodException 方法不存在异常

IOException IO异常

SocketException Socket异常

6、http 响应码 301 和 302 代表的是什么？有什么区别？

答：301表示网页永久性转移到另一个地址

302表示临时性转移

区别：301是永久的重定向，搜索引擎在抓取新内容的同时也将旧的网址替换为重定向之后的网址

302重定向是临时的重定向，搜索引擎抓取新的内容而保留旧的网址

7、forward 和 redirect 的区别？

答：forward是转发，redirect是重定向

地址栏URL显示：forward URL不会发生变化，redirect会发生变化

数据共享：forward可以共享request里面的数据，redirect不可以

效率：forward比redirect效率高

8、简述 tcp 和 udp的区别？

答：

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| 是否基于连接 | 是 | 否 |
| 数据传输可靠性 | 可靠 | 不可靠 |
| 数据传输模式 | 面向字节流 | 面向报文 |
| 传输效率 | 慢 | 快 |

9、tcp 为什么要三次握手，两次不行吗？为什么？

答：如果采用两次握手，那么只要服务器发出确认数据包就会建立连接，但由于客户端此时并未响应服务器端的请求，那此时服务器端就会一直在等待客户端，这样服务器端就白白浪费了一定的资源，若采用三次握手，服务器端没有收到来自客户端的再次确定，则会知道客户端并没有要求建立请求，就不会浪费服务器资源。

10、说一下 tcp 粘包是怎么产生的？

答：TCP粘包可能发生在发送端或者接收端，分别来看两端各种产生粘包的原因：

发送端粘包：发送端需要等缓冲区满才发送出去，造成粘包；

接收端粘包：接收方不及时接收缓冲区的包，造成多个包接收；

11、OSI 的七层模型都有哪些？

答：物理层：利用传输介质为数据链路层提供物理连接，实现比特流的透明传输。

数据链路层：负责建立和管理节点间的链路。

传输层：向用户提供可靠的端到端的差错和流量控制，保证报文的正确传输。

网络层：通过路由选择算法，为报文或分组通过通信子网选择最适当的路径。

会话层：向两个实体的表示层提供建立和使用连接的方法。

表示层：处理用户信息的表示问题，如编码、数据格式转换和加解密等。

应用层：直接向用户提供服务，完成用户希望在网络上完成的各种工作。

12、get 和 post 请求有哪些区别？

答：get请求会被浏览器主动缓存，而post不会。

Get传递参数有大小限制，而post没有。

Post参数传输更安全，get的参数会明文显示在URL上，post不会

13、如何实现跨域？

答：实现跨域有以下几种方案：

使用JSONP跨域;

在单个接口使用注解@CrossOrigin运行跨域;

服务器端运行跨域设置CORS等于\*;

14、说一下 JSONP 实现原理？

答：jsonp：JSON with Padding，它是利用script标签的src连接可以访问不同源的特性，加载远程返回的“JS函数”来执行的。

15、说一下你熟悉的设计模式？

答：策略模式：策略模式的主要目的主要是将算法的定义和使用分开，也就是将算法的行为和环境分开，将算法的定义放在专门的策略类中，每一个策略类封装一个实现算法。而使用算法的环境中针对抽象策略编程，而不是针对实现编程，符合依赖倒置原则。

迭代器模式：提供一种方法顺序访问一个聚合对象中的各种元素，而又不暴露该对象的内部表示。

单例模式：保证一个类在内存中对象的唯一性，节省系统开销。

工厂方法模式：解耦代码

16、简单工厂和抽象工厂有什么区别？

答：简单工厂：工厂类（SimpleFactory）拥有一个工厂方法（create），接受了一个参数，通过不同的参数实例化不同的产品类。对于增加新的产品，无能为力。

工厂方法：针对每一种产品提供一个工厂类。通过不同的工厂实例来创建不同的产品实例。抽象工厂：用来生产不同产品族的全部产品，对于增加新的产品，无能为力；支持增加产品族