接口有什么用

1、通过接口可以实现不相关类的相同行为，而不需要了解对象所对应的类。
2、通过接口可以指明多个类需要实现的方法。
3、通过接口可以了解对象的交互界面，而不需了解对象所对应的类。
另：Java 是单继承，接口可以使其实现多继承的功能。

说说 http,https 协议

HTTP：
是互联网上应用最为广泛的一种网络协议，是一个客户端和服务器端请求和应答的标准（TCP），用于从 WWW 服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

HTTPS：
是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版，即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。

区别：
1、https 协议需要到 ca 申请证书，一般免费证书较少，因而需要一定费用。
2、http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。
3、http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。
4、http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全。

说说 tcp/ip 协议族

TCP/IP 协议族是一个四层协议系统，自底而上分别是数据链路层、网络层、传输层和应用层。每一层完成不同的功能，且通过若干协议来实现，上层协议使用下层协议提供的服务。
1、数据链路层负责帧数据的传递。
2、网络层责数据怎样传递过去。

3、传输层负责传输数据的控制（准确性、安全性）
4、应用层负责数据的展示和获取。

tcp 五层网络协议

物理层：
为数据端设备提供传送数据的通路，数据通路可以是一个物理媒体，也可以是多个物理媒体连接而成。

数据链路层：
为网络层提供数据传送服务。

网络层：
路由选择和中继、激活,终止网络连接、在一条数据链路上复用多条网络连接,多采取分时复用技术 、差错检测与恢复、排序,流量控制、服务选择、网络管理 。

传输层：
传输层是两台计算机经过网络进行数据通信时,第一个端到端的层次，具有缓冲作用。

应用层：
应用层向应用程序提供服务

TCP 与 UDP 的区别

1、基于连接与无连接
2、TCP 要求系统资源较多，UDP 较少；
3、UDP 程序结构较简单
4、流模式（TCP）与数据报模式(UDP);
5、TCP 保证数据正确性，UDP 可能丢包
6、TCP 保证数据顺序，UDP 不保证

cookie 和 session 的区别，分布式环境怎么保存用户状态

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的 COOKIE 并进行 COOKIE 欺骗，考虑到安全应当使用 session。
3、session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用 COOKIE。
4、单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。

分布式环境下的 session（举例两种）：

服务器 session 复制
原理：任何一个服务器上的 session 发生改变（增删改），该节点会把这个 session 的所有内容序列化，然后广播给所有其它节点，不管其他服务器需不需要 session，以此来保证 Session 同步。
优点：可容错，各个服务器间 session 能够实时响应。
缺点：会对网络负荷造成一定压力，如果 session 量大的话可能会造成网络堵塞，拖慢服务器性能。

session 共享机制
使用分布式缓存方案比如 memcached、redis，但是要求 Memcached 或 Redis 必须是集群。

GIT 和 SVN 的区别

1、GIT 是分布式的，SVN 不是。
2、GIT 把内容按元数据方式存储，而 SVN 是按文件。
3、GIT 分支和 SVN 的分支不同。
4、GIT 没有一个全局的版本号，而 SVN 有。
5、GIT 的内容完整性要优于 SVN。
（一般问会不会用，知道这些区别貌似也没卵用）

请写一段栈溢出、堆溢出的代码

递归调用可以导致栈溢出
不断创建对象可以导致堆溢出

代码如下：

```
public  class  Test  {

      public  void  testHeap(){
            for(;;){
                     ArrayList  list  =  new  ArrayList (2000);
               }
      }
      int  num=1;
      public  void  testStack(){
            num++;
            this.testStack();
       }

      public  static  void  main(String[] args){
            Test   t   =  new  Test ();
            t.testHeap();
            t.testStack();
      }
}
```