**1.  面向对象编程的三大特性是什么，请简要阐述**

(1).继承：

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继 承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增 加新的方法使之更适合特殊的需要。

(2).封装：

封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。

(3).多态性：

多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

**2.  String 和StringBuffer的区别**

JAVA平台提供了两个类:String和StringBuffer，它们可以储存和操作字符串，即包含多个字符的字符数据这个String类提供了数值不可改变字符串而这个StringBuffer类提供的字符串进行修改当你知道字符数据要改变的时候你就可以使用StringBuffer典型地，你可以使用StringBuffers来动态构造字符数据

**3.  说出ArrayList,Vector, LinkedList的存储性能和特性**

 ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector由于使用了synchronized思路方法(线程安全) ，通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项前后项即可，所以插入速度较快

**4.  Collection 和 Collections的区别**

Collection是集合类的上级接口，继承与他的接口主要有Set 和List.  Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

**5.  HashMap和Hashtable的区别**

HashMap是Hashtable的轻量级实现(非线程安全的实现)，他们都完成了Map接口，主要区别在于HashMap允许空(null)键值(key),由于非线程安全，效率上可能高于HashtableJAVA中文站社区门户oC)| g| ax ，HashMap允许将Null作为一个entryde key 或者value,而Hashtable不允许 HashMap把Hashtable的contains思路方法去掉了，改成containsvalue和containsKey因为contains思路方法容易让人引起误解Hashtable继承自Dictionary类，而HashMap是Java1.2引进的Map interface的一个实现。

    最大的区别是，Hashtable的思路方法是Synchronize的，而HashMap不是，在多个线程访问Hashtable时，不需要自己为它的思路方法实现同步，而HashMap 就必须为的提供外同步。

    Hashtable和HashMap采用的hash/rehash算法都大概一样，所以性能不会有很大的差异。

**6.  final, finally, finalize的区别**

final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

**7.  Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型?**

**重载（Overload）：**

（1）方法重载是让类以统一的方式处理不同类型数据的一种手段。多个同名函数同时存在，具有不同的参数个数/类型。重载Overloading是一个类中多态性的一种表现。  
        （2）Java的方法重载，就是在类中可以创建多个方法，它们具有相同的名字，但具有不同的参数和不同的定义。调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法, 这就是多态性。  
        （3）重载的时候，方法名要一样，但是参数类型和个数不一样，返回值类型可以相同也可以不相同。无法以返回型别作为重载函数的区分标准。

**重写（Override）：**

      （1）  父类与子类之间的多态性，对父类的函数进行重新定义。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。在Java中，子类可继承父类中的方法，而不需要重新编写相同的方法。但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。方法重写又称方法覆盖。  
    （2） 若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖原有的方法。如需父类中原有的方法，可使用super关键字，该关键字引用了当前类的父类。  
   （3）子类函数的访问修饰权限不能少于父类的

重写方法的规则:  
1.参数列表必须完全与被重写的方法的相同,否则不能称其为重写而是重载.  
2.返回的类型必须一直与被重写的方法的返回类型相同,否则不能称其为重写而是重载.  
3.访问修饰符的限制一定要大于被重写方法的访问修饰符（public>protected>default>private）  
4.重写方法一定不能抛出新的检查异常或者比被重写方法申明更加宽泛的检查型异常．例如，  
父类的一个方法申明了一个检查异常IOException，在重写这个方法是就不能抛出Exception，只能抛出IOException的子类异常，可以抛出非检查异常．

而重载的规则：  
1.必须具有不同的参数列表；  
2.可以有不同的返回类型，只要参数列表不同就可以了；  
3.可以有不同的访问修饰符；  
4.可以抛出不同的异常；

重写和重载的区别在于：

重写多态性起作用，对调用被重载过的方法可以大大减少代码的输入量，同一个方法名只要往里面传递不同的参数就可以拥有不同的功能或返回值．

用好重写和重载可以设计一个结构清晰而简洁的类，可以说重写和重载在编写代码过程中的作用非同一般．

Overloaded的方法是可以改变返回值的类型。

**8.  error和exception有什么区别?**

error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。   
    exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。

**9.  abstract class和interface有什么区别?**

1.abstract class 在 Java 语言中表示的是一种继承关系，一个类只能使用一次继承关系。但是，一个类却可以实现多个interface。

2.在abstract class 中可以有自己的数据成员，也可以有非abstarct的成员方法，而在interface中，只能够有静态的不能被修改的数据成员(也就是必须是static final的，不过在 interface中一般不定义数据成员)，所有的成员方法都是abstract的。

　　3.abstract class和interface所反映出的设计理念不同。其实abstract class表示的是"is-a"关系，interface表示的是"like-a"关系。

　　4.实现抽象类和接口的类必须实现其中的所有方法。抽象类中可以有非抽象方法。接口中则不能有实现方法。

　　5.接口中定义的变量默认是public static final 型，且必须给其初值，所以实现类中不能重新定义，也不能改变其值。

　　6.抽象类中的变量默认是 friendly 型，其值可以在子类中重新定义，也可以重新赋值。

　　7.接口中的方法默认都是 public,abstract 类型的。

**10. List, Set, Map是否继承自Collection接口**

   List,Set是继承自Collection，Map不是。

**11. swtich是否能作用在byte上，是否能作用在long上，是否能作用在String上?**

swtich中只能用byte short int char

**12. char型变量中能不能存贮一个中文汉字?为什么?**

能够定义成为一个中文的,因为java中以unicode编码,一个char占16个bit,所以放一个中文是没问题的**。**

**13. 多线程有几种实现方法,都是什么?哪一种方式比较优秀?同步有几种实现方法,都是什么?**

多线程有两种实现方法，分别是继承Thread类与实现Runnable接口

实现Runnable接口比较好，因为实现类可以实现多个接口，而只能继承一个类。

同步的实现方面有两种，分别是synchronized,wait与notify

**14.  java中有几种类型的流？JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类**

字节流，字符流。字节流继承与inputStream/outputStream,字符流继承于inputStreamReader/outputSteamWriter

**15.java深度克隆和浅度克隆的区别**

**【浅克隆】，通常只是对克隆的实例进行复制，但里面的其他子对象，都是共用的。**

**Clone()方法属于浅度克隆**

**【深克隆】，克隆的时候会复制它的子对象的引用，里面所有的变量和子对象都是又、额外拷贝了一份。**