**面向对象编程（OOP）**

**封装**：封装给对象提供了隐藏内部变量和方法的能力，有3种修饰符：public，private（只有当前类可以访问）和protected（其他包不能访问）。通过隐藏对象的属性来保护对象内部的状态，提高了代码的可用性和可维护性，因为对象的行为可以被单独的改变或者是扩展禁止对象之间的不良交互提高模块化。

**多态**：通过一个父类可以引用不同的子类，只有在运行的时候才知道引用变量所指向的具体实例对象。向上转型：parent=new child()，child向上转为parent，只能访问parent中拥有的属性方法，优先使用child内重写的方法；（继承、重写、向上转型）

**继承：**给对象提供了从基类获取字段和方法的能力，代码的重用性，也可以在不修改类的情况下给现存的类添加新特性。（单继承）

**接口**：方法和属性的集合，没有方法的实现，这些方法可以在不同的地方被不同的类实现。变量：public,static,final显示初始化为常量，方法：public,abstract，么有方法体；没有构造函数.

1. Java虚拟机是一个可以执行Java字节码的虚拟机进程。Java源文件被编译成能被Java虚拟机执行的字节码文件。Java允许应用程序可以运行在任意的平台，而不需要程序员为每一个平台单独重写或者是重新编译。因为JVM知道底层硬件平台的指令长度和其他特性。（类加载器、执行引擎、运行时数据区）

2. JDK：java development kit；JRE：java runtime environment

3. final：常量，不能被继承的类，不能被覆盖的方法

4. static：全局静态，所有子类共享，不要实例化就可以使用，只分配一次内存

5. .数据类型是：byte/short/int/long/float/double/boolean/char，自动装箱是Java编译器在基本数据类型和对应的对象包装类型之间做的一个转化。比如：把int转化成Integer，double转化成double，反之就是自动拆箱。

6. 重载：在同一个类里面两个或者是多个方法的方法名相同但是参数不同的情况。

重写：子类重新定义父类的方法，必须有相同的方法名，参数列表和返回类型。

7. 当新对象被创建的时候，构造函数会被调用。每一个类都有构造函数。在程序员没有给类提供构造函数的情况下，Java编译器会为这个类创建一个默认的构造函数。可以为一个类创建多个构造函数。每一个构造函数必须有它自己唯一的参数列表。

8. 抽象方法：只有声明，而没有具体的实现，abstract修饰；一个类含有抽象方法即为抽象类，必须用public或者protected修饰，不能被直接实例化，需要继承后实现抽象方法

9. 接口和抽象类的区别：接口中所有的方法都是抽象的，而抽象类则可以同时包含抽象和非抽象的方法。类可以实现很多个接口，但是只能继承一个抽象类。类如果要实现一个接口，它必须要实现接口声明的所有方法，类可以不实现抽象类声明的所有方法。

10. 值传递：形式参数类型是基本数据类型，只是用实际参数的值初始化自己的存储单元内容，是两个不同的存储单元，所以方法执行中形式参数值的改变不影响实际参数的值。

引用传递：形式参数类型是引用数据类型参数，传地址

11. 创建线程：继承Thread类，重写run方法；实现Runnable接口，实现run方法；java单继承，利用扩展Thread类创建的多个线程，虽然执行的是相同的代码，但彼此相互独立，且各自拥有自己的资源，互不干扰。Runnable使代码能够被多个线程共享，而且代码和数据资源相对独立，从而特别适合多个具有相同代码的线程去处理同一资源的情况。

12.就绪(Runnable)运行中(Running)等待中(Waiting)睡眠中(Sleeping) I/O阻塞(Blocked on I/O)同步阻塞(Blocked on Synchronization)死亡(Dead)

13. 在Java语言中，每一个对象有一把锁。线程可以使用synchronized关键字来获取对象上的锁。监视器和锁在Java虚拟机中是一块使用的。监视器监视一块同步代码块，确保一次只有一个线程执行同步代码块。每一个监视器都和一个对象引用相关联。线程在获取锁之前不允许执行同步代码。

14. 两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程都陷入了无限的等待中。使用多线程的时候，一种非常简单的避免死锁的方式就是：指定获取锁的顺序，并强制线程按照指定的顺序获取锁。因此，如果所有的线程都是以同样的顺序加锁和释放锁，就不会出现死锁了。

15.Java集合类框架的基本接口：**Collection**：代表一组对象，每一个对象都是它的子元素。

Set：不包含重复元素的Collection。List：有顺序的collection，并且可以包含重复元素。

**Map**：可以把键(key)映射到值(value)的对象，键不能重复。集合类没有实现Cloneable和Serializable接口

16. 迭代器(Iterator)：提供了很多对集合元素进行迭代的方法，每一个集合类都包含了可以返回迭代器实例的迭代方法，迭代器可以在迭代的过程中删除底层集合的元素。

17. HashMap以键值对(key-value)的形式存储元素的，当调用put()方法的时候，会计算key的hash值，然后把键值对存储在集合中合适的索引上。如果key已经存在了，value会被更新成新值。HashMap的一些重要的特性是它的容量(capacity)，负载因子(load factor)和扩容极限(threshold resizing)。

18. HashMap和Hashtable都实现了Map接口，HashMap允许键和值是null，而Hashtable不允许键或者值是null。Hashtable是同步的，而HashMap不是。因此，HashMap更适合于单线程环境，而Hashtable适合于多线程环境。

19. Array可以包含基本类型和对象类型，ArrayList只能包含对象类型。Array大小是固定的，ArrayList的大小是动态变化的。ArrayList提供了更多的方法和特性，比如：addAll()，removeAll()，iterator()等等。对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。

20. ArrayList和LinkedList都实现了List接口，ArrayList是基于索引的数据接口，它的底层是数组。它可以以O(1)时间复杂度对元素进行随机访问。与此对应，LinkedList是以元素列表的形式存储它的数据，每一个元素都和它的前一个和后一个元素链接在一起，在这种情况下，查找某个元素的时间复杂度是O(n)。相对于ArrayList，LinkedList的插入，添加，删除操作速度更快，因为当元素被添加到集合任意位置的时候，不需要像数组那样重新计算大小或者是更新索引。LinkedList比ArrayList更占内存，因为LinkedList为每一个节点存储了两个引用，一个指向前一个元素，一个指向下一个元素。

21. Java提供了只包含一个compareTo()方法的Comparable接口。Java提供了包含compare()和equals()两个方法的Comparator接口。

22. PriorityQueue是一个基于优先级堆的无界队列，它的元素是按照自然顺序(natural order)排序的。在创建的时候，我们可以给它提供一个负责给元素排序的比较器，不允许null值，因为他们没有自然顺序，不是线程安全的，入队和出队的时间复杂度是O(log(n))。

23. 如何权衡是使用无序的数组还是有序的数组？有序数组最大的好处在于查找的时间复杂度是O(log n)，而无序数组是O(n)。有序数组的缺点是插入操作的时间复杂度是O(n)，因为值大的元素需要往后移动来给新元素腾位置，无序数组的插入时间复杂度是常量O(1)。

24. 根据应用的需要正确选择要使用的集合的类型对性能非常重要，比如：假如元素的大小是固定的，而且能事先知道，我们就应该用Array而不是ArrayList。有些集合类允许指定初始容量。因此，如果我们能估计出存储的元素的数目，我们可以设置初始容量来避免重新计算hash值或者是扩容。

25. Enumeration速度是Iterator的2倍，同时占用更少的内存。Iterator远远比Enumeration安全，因为其他线程不能够修改正在被iterator遍历的集合里面的对象。同时，Iterator允许调用者删除底层集合里面的元素，这对Enumeration来说是不可能的。

26. HashSet是由一个hash表来实现的，因此，它的元素是无序的。add()，remove()，contains()方法的时间复杂度是O(1)。TreeSet是由一个树形的结构来实现的，它里面的元素是有序的。因此，add()，remove()，contains()方法的时间复杂度是O(logn)。

垃圾收集器(Garbage Collectors)

27. 垃圾回收的目的是识别并且丢弃应用不再使用的对象来释放和重用资源。

28. System.gc()和Runtime.gc()用来提示JVM要进行垃圾回收，何时开始取决于JVM。

29. 在释放对象占用的内存之前，垃圾收集器会调用对象的finalize()方法，建议在该方法中释放对象持有的资源。

30. 如果对象的引用被置为null，垃圾收集器是不会立即释放对象占用的内存，而是表明在下一个垃圾回收周期中，这个对象将是可被回收的。

31. Java堆结构：JVM的堆是运行时数据区，所有类的实例和数组都是在堆上分配内存。它在JVM启动的时候被创建。对象所占的堆内存是由自动内存管理系统也就是垃圾收集器回收。堆内存是由存活和死亡的对象组成的。存活的对象是应用可以访问的，不会被垃圾回收。死亡的对象是应用不可访问尚且还没有被垃圾收集器回收掉的对象。一直到垃圾收集器把这些对象回收掉之前，他们会一直占据堆内存空间。

32. 在Java中，对象什么时候可以被垃圾回收？当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

异常处理

33. Java中有两种异常：受检查的(checked)异常和不受检查的(unchecked)异常。不受检查的异常不需要在方法或者是构造函数上声明，就算方法或者是构造函数的执行可能会抛出这样的异常，并且不受检查的异常可以传播到方法或者是构造函数的外面。相反，受检查的异常必须要用throws语句在方法或者是构造函数上声明。

34. Exception和Error都是Throwable的子类。Exception用于用户程序可以捕获的异常情况。Error定义了不期望被用户程序捕获的异常。

35. throw关键字用来在程序中明确的抛出异常，相反，throws语句用来表明方法不能处理的异常。每一个方法都必须要指定哪些异常不能处理，所以方法的调用者才能够确保处理可能发生的异常，多个异常是用逗号分隔的。

36. 无论是否抛出异常，finally代码块总是会被执行。就算是没有catch语句同时又抛出异常的情况下，finally代码块仍然会被执行。最后要说的是，finally代码块主要用来释放资源，比如：I/O缓冲区，数据库连接。

Java小应用程序(Applet)

37. java applet是能够被包含在HTML页面中并且能被启用了java的客户端浏览器执行的程序。Applet主要用来创建动态交互的web应用程序。

38. Applet的生命周期Init：每次被载入的时候都会被初始化。Start：开始执行applet。Stop：结束执行applet。Destroy：卸载applet之前，做最后的清理工作。

39. applet被载入时：创建applet控制类的实例，然后初始化applet，最后开始运行。

40. Applet和普通的Java应用程序有什么区别？applet是运行在启用了java的浏览器中，Java应用程序是可以在浏览器之外运行的独立的Java程序。但是，它们都需要有Java虚拟机。进一步来说，Java应用程序需要一个有特定方法签名的main函数来开始执行。Java applet不需要这样的函数来开始执行。最后，Java applet一般会使用很严格的安全策略，Java应用一般使用比较宽松的安全策略。

41. Java中的Swing组件使用了MVC(视图-模型-控制器)设计模式。

JDBC

42. JDBC(Java Data Base Connectivity)是允许用户在不同数据库之间做选择的一个抽象层。[JDBC允许开发者用JAVA写数据库应用程序](http://www.javacodegeeks.com/2014/03/java-8-friday-java-8-will-revolutionize-database-access.html" \t "_blank)，而不需要关心底层特定数据库的细节。

43. JDBC驱动提供了特定厂商对JDBC API接口类的实现，必须要提供下面这些类的实现[Connection](http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html), [Statement](http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html), [PreparedStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html),[CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html), [ResultSet](http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html)和[Driver](http://docs.oracle.com/javase/7/docs/api/java/sql/Driver.html" \t "_blank)。

44.Class.forName()方法有什么作用？这个方法用来载入跟数据库建立连接的驱动。

45. 如果SQL语句被多次执行选用PreparedStatement是最好的,第一次执行消耗是很高的,它的性能体现在后面的重复执行.

远程方法调用(RMI)