**一、向上转型**

public class A {

public void a1() {

       System.out.println("Superclass");

}

}

A的子类B：

package a.b;

public class B extends A {

public void a1() {

       System.out.println("Childrenclass"); //覆盖父类方法

}

       public void b1(){} //B类定义了自己的新方法

}

1、向上转型： A a = new B()

a.a1()输出的是Childrenclass

2、由于向上转型，a对象会遗失和父类不同的方法，例如b1()

3、向上转型的作用，减少重复代码，父类为参数，调有时用子类作为参数，就是利用了向上转型。这样使代码变得简洁。体现了JAVA的抽象编程思想。

**二、构造过程**

单独的类：

1、 从父到子的static数据（static代码块，static成员）

2、父的构造块，父类的对象

3、父类的构造函数

4、子类的构造块，子类的对象

5、子类的构造函数

6.静态块，构造快的区别

静态块：用static申明，JVM加载类时执行，仅执行一次

构造块：类中直接用{}定义，每一次创建对象时执行

执行顺序优先级：静态块>main()>构造块>构造方法

有继承关系的类：

<http://www.cnblogs.com/SirSmith/p/5536288.html>

<http://blog.csdn.net/hhfeifeige/article/details/40073973>

**三、抽象类与接口**

**抽象类**

包含有抽象方法的为抽象类

抽象类不能被实例化

继承抽象类的子类必须实现其所有的抽象方法才能被实例化

**2.接口：互不相同的类具有相同的功能，解决单继承问题**

只能包含常量和抽象方法

不包含构造函数

不能被实例化

类实现接口，必须实现所有方法，否则为抽象类

只能有public接收

**四、内部类**

静态内部类可以包含static成员，而非static内部类却不可以包含static成员。

匿名内部类：适用于使用一次对象的场所，直接实例化，不用创建类名。

**五、泛型**

泛型集合，List<String> 就是在使用泛型

获得编译时的运行安全，不需要进行强制转换

**六、枚举**

常量的数据集合

switch支持枚举

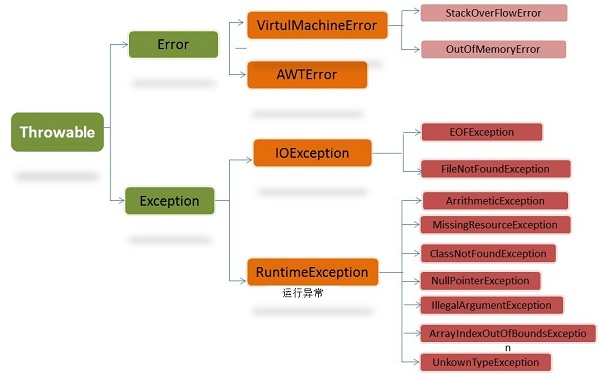
**六、异常**

编译异常必须捕获和抛出：IOException（流传输异常），SQLException（数据库操作异常）

运行时异常（非检查型异常）可以捕获和抛出也可以不做任何处理：NullpointException（空指针异常），ClassCastException（类型转换异常），IndexOutOfBoundsException（数组越界异常）

非运行时异常也叫检查型异常

finally异常无论发生都要执行



throw与throws的区别

throws出现在方法函数头；而throw出现在函数体。

throws表示出现异常的一种可能性，并不一定会发生这些异常；throw则是抛出了异常，执行throw则一定抛出了某种异常。

**七、线程**

线程与进程的差别

线程是cup进行调度和分派的最小单位，进程是系统进行资源调度和分配的最小单位

线程共享内存，需要采取同步措施，进程之间一般都有自己独立的内存

一般一个进程包含多个线程

线程执行切换开销小，进程之间的切换开销大

线程对象与线程的差别

线程的生命周期：创建、就绪、运行、阻塞、死亡

线程调度

setPriprity（）：设置优先级，越大越高

sleep（）：休眠进入阻塞，不释放锁

yield（）：让步，转为就绪，重新排队

join（）：等待上一个线程结束，不释放锁

===t（）：释放锁，进入等待队列，进入琐池

**八、二维数组a[][]**

行数：a.length;

列数：a[0].length;

**九、map的**

**十、String、StringBuffer、StringBuilder的区别**

string ：字符串常量，长度不可变

stringbuffer：线程安全，频繁插入修改试用，多线程

stringbuilder：线程非安全，频繁插入修改试用，单线程

**十一、**[**提升SQLite数据插入效率低、速度慢的方法**](http://blog.csdn.net/majiakun1/article/details/46607163)

所谓”事务“就是指一组SQL命令，这些命令要么一起执行，要么都不被执行。在SQLite中，每调用一次sqlite3\_exec()函数，就会隐式地开启了一个事务，如果插入一条数据，就调用该函数一次，事务就会被反复地开启、关闭，会增大IO量。

如果在插入数据前显式开启事务，插入后再一起提交，则会大大提高IO效率，进而加数据快插入速度。

开启事务只需在上述代码的前后各加一句开启与提交事务的命令。

**十二、理解java对象的序列化**

概念：java平台允许我们在内存中创建使用对象，但是一般情况下自由jvm运行时这些对象才是存在，如果要求在jvm停止运行后能够持久化指定的对象，并能够重新读取被保存的对象。

应用：对象持久化存储，网络中传递，通过序列化在进程间传递对象

android中序列化有两种方式：serializable，parcelable

**十三、重写与重载的区别**

重载：参数个数不同、类型不同，返回值可以相同可以不同，不能以返回值的不同区分重载。

重载中参数类型顺序不一样也是重载

public void demo(String a1,int a2){}

public void demo(int a1,String a2){}

重写：子类重写父类，相同的方法名、返回类型和参数

**十四、泛型的理解**

[**http://blog.csdn.net/davidluo001/article/details/50317397**](http://blog.csdn.net/davidluo001/article/details/50317397)

**十五、final的理解**

final在Java中是一个保留的关键字，可以声明成员变量、方法、类以及本地变量。一旦你将引用声明作final，你将不能改变这个引用。

（静态绑定）final也可以声明方法。方法前面加上final关键字，代表这个方法不可以被子类的方法重写。final方法比非final方法要快，因为在编译的时候已经静态绑定了，不需要在运行时再动态绑定。

使用final来修饰的类不能被继承。

接口中声明的所有变量本身是final的

final和abstract这两个关键字互斥，final类就不可能是abstract的。

没有在声明时初始化final变量的称为空白final变量(blank final variable)，它们必须在构造器中初始化，或者调用this()初始化

对于集合对象声明为final指的是引用不能被更改，但是你可以向其中增加，删除或者改变内容

final关键字的好处

final关键字提高了性能。JVM和Java应用都会缓存final变量。

final变量可以安全的在多线程环境下进行共享，而不需要额外的同步开销。

使用final关键字，JVM会对方法、变量及类进行

**十六、简单工厂模式**

利用的是向上转型，利用一个专门的类去实例化各种子类。

**十七、生产者消费者模式**

生产者消费者模式是通过一个容器来解决生产者和消费者的强耦合问题。生产者和消费者彼此之间不直接通讯，而通过阻塞队列来进行通讯，所以生产者生产完数据之后不用等待消费者处理，直接扔给阻塞队列，消费者不找生产者要数据，而是直接从阻塞队列里取，阻塞队列就相当于一个缓冲区，平衡了生产者和消费者的处理能力。

这个阻塞队列就是用来给生产者和消费者解耦的。纵观大多数设计模式，都会找一个第三者出来进行解耦，如工厂模式的第三者是工厂类，模板模式的第三者是模板类。在学习一些设计模式的过程中，如果先找到这个模式的第三者，能帮助我们快速熟悉一个设计模式。

**十八、引用的四种类型**

强引用StrongReferenc

软引用SoftReference：内存敏感的高速缓存，内存空间不足了，就会回收这些对象的内存，有用但非必须

弱引用WeakReference：垃圾回收器线程扫描它所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存

虚引用PhantomReference：一个对象仅持有虚引用，那么它就和没有任何引用一样

参考资料：<http://www.cnblogs.com/skywang12345/p/3154474.html>

**十九、==、equals()、hashcode**

==：两个对象在jvm中的地址

equals：首先比较对象地址，相同返回ture，不同比较两个对象的值，相同返回ture

hashcode：返回对象的32位jvm内存地址，对象存储的hashcode地址

当两个对象equals相等时，返回的hashcode必须是相同的

当两个对象equals不相等时，返回的hashcode不一定要相同

当对象放入到集合的时候，开始计算hashcode，如果没有冲突，直接放入。如果有冲突，则会调用equals方法，如果不相同则放入，如果相同，则放弃。

当我们定义了一个class为key，那么euqals 的方法需要我们重写，我们自己去定义什么时候两个对象相等。又由于当两个对象相等时hashcode需要相同，所以我们也需要重写hashcode的方法保证相同。

<http://www.cnblogs.com/liuyang0/p/6271343.html>

**二十、list删除不能用foreach，for循环的话从大到小删。可以利用迭代器**

Collection集合类中只能在Iterator中删除元素的原因，但是不能修改元素

k5.0以上的for-each也是利用内部的iterator来遍历集合的（跟以前的iterator一样）获得的Iterator是一个内部类产生的迭代器，这个迭代器在调用next方法时，会检查列表是否被修改过，如果被修改过，就会抛出ConcurrentModificationException异常。进一步说，当使用 fail-fast iterator 对Collection 或 Map 进行迭代操作过程中尝试直接修改 Collection / Map 的内容时，即使是在单线程下运xi,java.util.ConcurrentModificationException 异常也将被抛出。Iterator 是工作在一个独立的线程中，并且拥有一个 mutex 锁。 Iterator 被创建之后会建立一个指向原来对象的单链索引表，当原来的对象数量发生变化时，这个索引表的内容不会同步改变，所以当索引指针往后移动的时候就找不到要迭代的对象，所以按照 fail-fast 原则 Iterator 会马上抛出 java.util.ConcurrentModificationException 异常。　　所以 Iterator 在工作的时候是不允许被迭代的对象被改变的。但你可以使用 Iterator 本身的方法 remove() 来删除对象，Iterator.remove() 方法会在删除当前迭代对象的同时维护索引的一致性。

**二十一、匿名内部类调用的参数为何是final**

一般匿名内部类是一个抽象类或者接口，实例化的时候就必须实现抽象方法

匿名内部类与外部类不是一个class文件，仅仅保留了对外部类的引用，传入的参数变为了匿名内部类自身的属性拷贝了一份到匿名内部类里面去了，并不是引用，变化不会影响到外部类的参数，需要保持一致。

<http://blog.csdn.net/chenssy/article/details/13170015>

[**二十二、**](http://blog.csdn.net/chenssy/article/details/13170015)**hashmap**中能否用自定义的类当key可以，但是需要重写equals()与hashcode()

**二十三、线程安全的list**：vector，其他的都不安全

**二十五、**

**二十六：泛型**

E – Element (在集合中使用，因为集合中存放的是元素)

T – Type（Java 类），也可以表示元素

K – Key（键）

V – Value（值）

N – Number（数值类型）

？ – 表示不确定的java类型（无限制通配符类型）

**二十七：finally：释放资源，网络连接，数据库连接**

**二十八：静态内部类与成员内部类的区别**

静态内部类不能访问外部类的非静态变量，只能访问外部类的静态成员

静态内部类不依赖外部类进行实例化，成员内部类需要依赖外部类实例化

成员内部类不能有静态成员，成员内部类可以访问外部类的非静态成员

成员内部类可以访问外部类的成员。

**二十九：jvm类的加载过程**

通过一个类的**全限定名（包名与类名）**来获取定义此类的**二进制字节流**（Class文件）。而获取的方式，可以通过jar包、war包、网络中获取、JSP文件生成等方式。

将这个字节流所代表的**静态存储结构**转化为**方法区**的运行时数据结构。这里只是转化了数据结构，并未合并数据。（方法区就是用来存放已被加载的类信息，常量，静态变量，编译后的代码的运行时内存区域）

在**内存中**生成一个代表这个类的java.lang.Class对象，作为方法区这个类的各种数据的访问入口。这个Class对象并没有规定是在Java堆内存中，它比较特殊，虽为对象，但存放在方法区中。

<http://blog.csdn.net/zhangliangzi/article/details/51319033>

**三十：虚拟机中堆的参数设置**

-Xmx –Xms：指定最大堆和最小堆

-Xmn：设置新生代大小

-XX:NewRatio：新生代（eden+2\*s）和老年代（不包含永久区）的比值

-XX:SurvivorRatio：（幸存代）设置两个Survivor区和eden的比值

-XX:+HeapDumpOnOutOfMemoryError

　　　　OOM时导出堆到文件，根据这个文件，我们可以看到系统dump时发生了什么。

-XX:+HeapDumpPath导出OOM的路径

6.-XX:PermSize  -XX:MaxPermSize 设置永久区的初始空间和最大空间

**三十一：虚拟机中栈的参数**

Xss：设置栈空间的大小。通常只有几百K

**三十二：& | 前后都执行 && || 前面先执行，满足后在执行右边，**

<<      :     左移运算符，num << 1,相当于num乘以2

>>      :     右移运算符，num >> 1,相当于num除以2

>>>    :     无符号右移，忽略符号位，空位都以0补齐

**三十三：collection与collections的区别**

collection为集合接口;List Set

collections为集合的操作类，不能实例化

**三十四：封装、继承、多态的理解**

封装：保护数据，隐藏内部实现细节，暴漏指定方法。

继承：复用父类方法

多态：引用变量在运行时才指定其指向的变量以及调用的方法，最常用的向上转型。

**三十五：foreach与for比较**

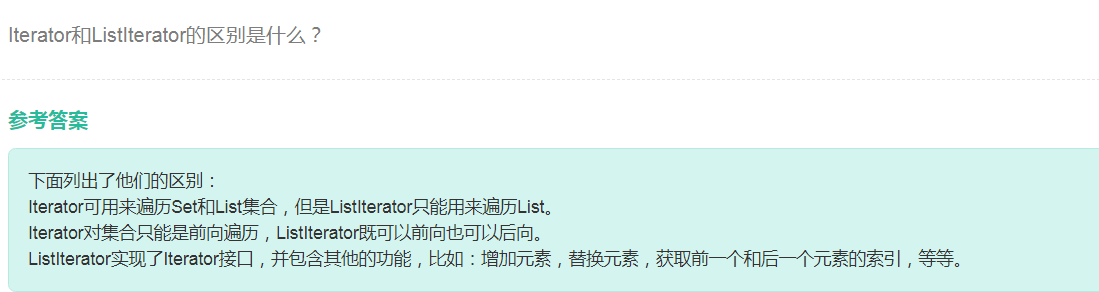
在固定长度或长度不需要计算的时候for循环效率高于foreach.

在不确定长度,或计算长度有性能损耗的时候,用foreach比较方便

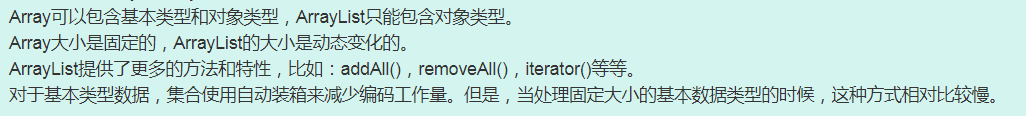
foreach的时候会锁定集合中的对象.期间不能修改.除非是iterator的remove

**三十六：**

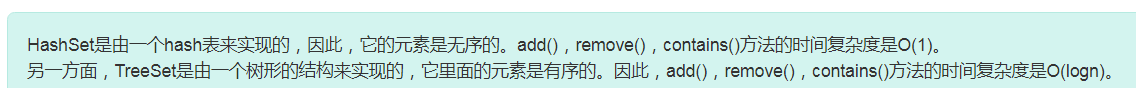
**Iterator与ListIterator的区别**



**Aarray与ArrayList的区别**



**HashMap与TreeMap的区别**



Java中能够产生事件的对象都可以称为事件源

**三十七：cookie与session的区别**

cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户的状态，就使用response向客户端浏览器颁发一个cookie。客户端浏览器会把cookie保存起来。当浏览器再次请求该网站时，浏览器就会把请求地址和cookie一同给服务器。服务器检查该cookie，从而判断用户的状态。服务器还可以根据需要修改cookie的内容。

session是另一种记录客户状态的机制。不同的是cookie保存在客户端浏览器中，而session保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上，这就是session。客户端浏览器再次访问时只需要从该session中查找该客户的状态就可以了。

三十八：java中常用的包

java.lang包：Thread、String、StringBuffer、Runnable、Math、Object、System

java.util包：集合，工具类、随机数生成、日期时间、事件模型等

java.io包：文件系统、数据流