Java多态中：

成员变量，编译看父类，运行也看父类

成员方法，编译看父类，运行看子类

instanceof:

用来比较一个引用型对象是不是属于某一类，只能对有继承或者实现关系的类才能使用

example：Boolean a = p instanceof Student 判断对象p是不是属于类Student，是的话返回ture,不是的话返回false

转型：

向上转型，可以调用父类和子类共有的类型

向下转型，需要强制，转过之后，就可以使用子类独有的方法

构造函数：

new一个对象其实实质就是在调用构造方法

所有的子类构造函数，不管重载多少个构造函数，第一行都要是super()，即先构造父类，如果不写的话，也是默认写了super()

构造方法第一行，写this()或者super()，但是两者不能同时存在，任选其一，必须要保证子类的所有构造方法调用到父类的构造方法即可

final:

修饰的是成员变量，固定的不是内存的默认值，即必须要手动赋值，绝对不是内存的默认值；或者在构造函数里赋值。

static:

在成员变量前面加入static，可以使这个数据成为对象的共享数据。

内存中，静态成员优先于非静态成员存在。在静态中不能使用非静态，因为现有的静态，再有非静态，所以静态里面无法预知到会有什么样的非静态会到来。

成员函数中要是没用到成员变量时，加上static，要是局部变量都是静态成员变量，那么也加上static。

内部类：

将一个类定义在另一个类的内部之中（成员内部类），或者其他类的内部成员方法之中（局部内部类）。

内部类可以调用外部类成员，包括私有；

外部类想要使用内部类成员，必须建立内部类对象。

包：

Java的包，其实就是我们电脑中的文件夹，包里存放的是类文件。当类的数目太多了，在项目中，将功能相近的类文件放到一个包里面，分类管理。

导入包时，必须引入到该包所在的最底层文件夹里才行。

权限：

default权限，且只能同包下的任何类可以使用；

protected权限，本包所有类里面都可以访问，或者其他包的子类里面也可以访问。

代码块：

静态代码块只执行一次static{ }

构造代码块{ }和构造函数Object( ){ }会在每次生成新的对象时，反复执行。

执行顺序：静态代码块 > 构造代码块 > 构造函数。

冲突修饰符：

abstract与private不能共用

abstract与static不能共用

abstract与final不能共用

静态方法：

如果类中的成员方法没有使用到类当中的成员变量，那么最好将这个方法设置为静态方法，方便于直接使用类名调用该方法，不用new对象造成空间的浪费。

抽象类：

抽象类是无法new对象的，抽象类要是想要作为参数，则必须要用子类的对象作为参数；同理，如果方法要返回一个抽象类的对象，那么也可以返回这个抽象类子类的对象。

toString:

打印一个对象就是打印这个对象的toString方法的返回值。

字符串的本质就是字符数组，一旦建立，终生不变。

s.toCharArray()：将s字符串转为数组。

StringBuffer:

因为String类过于消耗内存，所以StringBuffer类是为了方便字符串操作而产生的，提高原有字符串造作效率，是一种可变的字符串，其内部的字符数组是没有final的，是可变的。

常用方法：

append("string"):追加字符串string delete(start , end):删除从start位置到end位置的字符串 insert(start,string):在start位置插入字符串string replace(start,end string):将从start到end的字符串用string代替

append可以接收任意类型，但是返回的都是StringBuffer类型

Java中开始结束索引，都是包含了开始，但没有包含结束。

正则表达式：

[abc]是a，或者b，或者c

[^abc]不能是a、b、c中的任意一个

[a-zA-Z0-9\_]所有大小写字母或者数字或者下划线

[.]任意字符

[a]{5,10}字符a出现5之10次

1[34578][0-9]{9}第一个字符必须是1，第二个字符是34578中的任意一个，第三个字符是0-9中的任意一个，第三个字符根据要求重复9次。最终结果是一个手机号码的正则表达式。

a\*标识a出现多次

Boolean string.maches(规则)判断字符串string是否符合规则

String[] arry = string.split("-");将字符串string按照“-”来切割，将切割的结果放到数组里面。

post和get请求的区别：

https://www.cnblogs.com/logsharing/p/8448446.html

get会把数据放在url地址里，而post会把信息放到request body里面；get一般有传输大小2K,服务器端64K,而post没有限制。

GET产生一个TCP数据包；POST产生两个TCP数据包。

Data:

java中data类是以毫秒存储的。

SimpleDateFormat simple = new SimpleDateFormat("yyyy年MM月dd日HH点mm分ss秒.

format将Data类型改成字符串，parse将字符串改成Data类型。

BigInteger:

BigInteger类的对象可以表示任意长度的数字，通常使用字符串来作为构造函数的参数。

BigInteger a,b 如果表示a+b可以用a.add(b)，表示a-b可以用a.subtract(b)。

如果表示a\*b可以用a.multiply(b),表示a/b可以用a.divide(b)。

ArrayList:

集合是用来存储引用对象的可变数组，不能用于存储基本对象。

List中允许有重复元素，而且元素都是有序的；而Set中不允许有重复元素，而且元素都是无序的。

list.clear()清空集合；list.contains(object)判断一个对象是否存在集合里面。

list.size()返回集合的大小；list.toArray()将集合转换为数组。list.remove(object)删除对象，返回布尔值，如果有重复元素，则删除第一个遇到的元素。

Iterator<String> it = list.iterator();用来生成迭代器。it.hasnext() it.next() it.remove()。

Iterable:

增强for循环。

迭代器异常，并发修改异常。迭代器在工作时，不要去改变集合的长度，可以修改元素。

string.next()将指针向下移动一次，并且返回移动后指针所指向的对象。

LinkList:

擅长节点的插入删除，但是不擅长节点的查询。

Hash值：

Hash值相同，但是equal返回不一定为真，但是equal为真，则Hash值返回一定相同。

Map:

HashMap是无序的，且Map不允许出现重复key值，后出现的key会覆盖先出现的key,但是value值允许重复。

增强for不能遍历map

可变参数：

可变参数必须要声明参数格式，但是不要求参数的个数，例如int...a本质上就是一个数组。

在一个方法中，可变参数最多只能有一个，而且必须要写在参数列表的最后面。

Java异常：

当程序遇到异常时：

1，创建了异常的对象new ArrayIndexOfBoundsException()

2,将异常的对象抛出，抛出给方法的调用者（JVM）

当JVM收到异常后：

1，将异常信息，以红色的字体输出在控制台

2，将程序停止，结束

throw：后面加上异常类的对象，写在方法的内部

throws：后面加上异常类的类名，写在方法的外部

当调用了带有异常抛出的方法时，调用者就必须要处理这个方法抛出的异常。

try{} catch(Exception e){ } try将捕获到的异常抛给catch来处理。其实e就是等于抛出的异常对象。当catch处理完异常之后，程序就可以继续执行。

一个try可以带有多个catch。

抛出的异常如果是平级的，那么异常的捕获不分先后。但是如果异常中出现了上下级的异常，越高级的越是要放到下面。因为要是把高级的放到前面，那么低级的永远都不会执行了，实质就是多态的调用。

finally{ }必须要执行的代码块，不管有没有捕获到异常，里面的代码块都会被执行。

异常分为两大类：

编译异常：调用了抛出异常的方法，不处理编译失败（try throws）。

运行异常：抛出的异常时RuntimeException类或者他的子类，一旦出现这种异常，就要改源代码了。

运行异常的特点：

方法内部抛出的异常时运行异常，new XXXException

方法的声明上，不需要throws语句。

子类抛出的异常只能是和父类一样的异常，或者父类异常的子类，如果父类没有抛出异常，那么子类也不能抛出异常，只能用try catch来解决异常。

IO流：

Windows下的文件系统是不区分大小写的，用的是\;但是Linux系统是区分的，用的是/

File类的三种构造方法，File file = new File(String 路径名) ， File file = new File（String 父路径名， String 子路径名） ， File file = new File(File 父路径对象，String 子路径名)。

boolean mkdir()创建文件夹，mkdirs()创建多层（含单层）文件夹（推荐使用） boolean createNewFile()创建文件

boolean delete（）不走回收站，删除的内容无法从回收站里找回。

file.exists()如果file里面写的是相对路径，那么exists()函数将会从当前项目的根目录下去寻找文件是否存在。

FileFilter是过滤器，是一个抽象方法。可以得到满足要求的文件。

Outputstream:

write()写入数据，close()释放资源。

Inputstream:

read()读取数据，但是每次只会读取一个数据，而且会返回数据的ASCII值，当读取到文件的结尾后，会返回-1。

Writer.write()在往文件里面写完数据之后，必须要调用Writer.flush()这个方法刷新，否则无法将数据写入。其实close()方法也暗含着刷新这个功能。

写入的数据有byte变为char和string。

Reader.read()功能是读取文本文件，同上类似，但是只能读取字符和字符数组，无法读取字符串，因为对于一个文本文件，无法给字符串一个确切的定义。

OutputStreamWriter将字符转换为字节，目的就改变字符的不同编码表。是字符流向字节的桥梁。

InputStreamReader将字节转成字符流。

readline()方法只是返回一行的有效字符串，不会附加\r\n,如果这一行是空的，那么返回Null。

转换吗？转换流。InputStreamReader OutputStreamWriter

高效吗？缓冲区对象。BufferedXXX

InputStream

FileInputStream

BufferedInputStream

OuputStream

FileOutputStream

BufferedOuputStream

Writer

OutputStreamWriter

FileWriter

BufferedWriter

Reader

InputStreamReader

FileReader

BufferedReader

Properties类继承Hashtable，实现Map接口，可以和IO对象结合使用，实现数据的持久存储。

load(InputStream in)或者load(Reader r)传递任意字节或者字符的输入流。流对象读取文件中的键值对，保存到集合中。

ObjectOutputStream(FileOutStream fos)进行对象的序列化，即将对象写入文件中，ObjectInputStream(FileInputStream fis)进行对象的反序列化，即将文件中的对象读取出来。

序列化和反序列化只是针对对象的操作，并且类中的静态成员不属于对象成员，所以无法序列化。对于非静态成员，如果不想对其序列化，可以加上修饰词transient。

打印流PrintStream字节流和PrintWriter字符流

print方法会原样向文档里做输出，不会走码表。

在打印数组的时候，只有打印char数组才会遍历，打印其他的数组，只会打印出地址。

多线程：

当一个类继承了thread类，并且重写了run()方法就可以多线程了。

或者实现Runnable接口的类,创建tread对象，在构造函数中传runnable的实现类对象，调用tread类的方法start()。

多线程的安全性操作：

当一个线程进入操作数据时，无论是否休眠，别的线程只能等待。

数据库

Java中的类和数据库中的表有一个对应关系，表中的列即对应着类中的成员字段，表中的每行数据对应着Java中的对象。

查询中的通配符使用方法，例如：select name from students where name like '王\_' or '张%'; 其中\_匹配一个字符，%匹配多个字符。

查询在条件之间的数据可以用between…… and……;这个是包含头尾的。

先过滤where,再分组group by,再再过滤having,最后排序 oder by。

JDBC

JDBC操作数据库的步骤

1. 注册驱动.

2. 获得连接.

3. 获得语句执行平台

4. 执行sql语句

5. 处理结果

6. 释放资源.

preparedstatement可以防止SQL注入。

一般情况下，数据库的配置放到properties配置文件下，再使用IO读取文件，键值对存储到集合中；再以键值对的方式从集合中读取连结信息，完成数据库的连结。

Map中，只有LinkedHashMap是有序的。

为了解决频繁的数据库连接操作，可以使用连接池，功能类似于线程池，连接池也有close()方法，但是这个close()只是释放连接，并没有销毁连接。

项目中的分层设计

