常用DOS命令

d: 回车 盘符切换

dir(directory):列出当前目录下的文件以及文件夹

md (make directory) : 创建目录

rd (remove directory): 删除目录

cd (change directory)改变指定目录(进入指定目录)

cd.. : 退回到上一级目录cd\: 退回到根目录

del (delete): 删除文件,删除一堆后缀名一样的文件\*.txt

exit : 退出dos命令行

cls : (clear screen)清屏

方向键上下：显示刚才写的命令

cd \s:能删除带内容的目录

jvm充当java与其他语言的翻译，针对不同的其他语言使用不同的jvm

JRE为运行环境

JDK为开发环境，含JRE

Java语言严格区分大小写

Java程序的开发执行流程：(命令行模式下)

A:编写java源程序(.java)

B:javac xxx.java(编译生成class文件)

C:java xxx（执行class文件）

path环境配置是为了使javac应用范围从bin文件夹扩大到任意位置，记录的是exe文件的路径

classpath记录的是class文件的路径

main()是程序的起点

java中的int类和boolen类不相容，所以不能用int x=1；while（x）{}来判断

println会在最后面加入换行

java中的对象放在可回收的堆中

java中的一切都必须呆在类中

面向对象设计扩展功能不需要改动之前已经测试好的程序代码

创建类时可能同时需要创建独立、测试用的类

java程序在执行期是会互相交谈的一组对象

变量必须有类型

变量必须有名字

除非加上f，否则带小数点的值都被当做double来处理

名称以字母、下划线、$符号开头，不能用数字开头

安装jdk和jre时不用装系统盘，编辑系统变量时出现问题可能是要把%JAVA\_HOME%用安装jdk的绝对路径替换

Dog myDog=new Dog中等号左边为声明一个引用变量，右边为创建对象，等号连接对象和引用

Book b=new Book();Book c=b;的话c不会创建新对象，而是c也引用到b所指向的对象，b和c引用到同一个对象，若是c=null则表示c没有引用到任何事物，但还是个可以被指定引用到其他Book的引用变量

数组是对象，创建对象（如Dog）类型的数组时要切记数组中放的是引用变量（遥控器），还需要创建对象（狗）！！！如Dog [] pets=new Dog[3];pet[0]=new Dog();这句不能漏掉!!!!

变量有两种，一种是primitive主数据类型，一种是引用。

变量声明必须有类型和名称

搞清楚变量引用（类似指针，遥控器）和真正的对象（在堆中）的指向关系

java中的参数是通过值传递的，也就是拷贝传递，方法无法改变调用方所传入的的参数

封装时可以把对象中的实例变量设置为private，把方法（如实例变量的get和set）设置为public，这样可以通过在方法中对参数进行检查并判断是否执行来保证实例变量的安全性

实例变量都有默认值，如0,0.0，null等等，实例变量声明在类内，而不是在方法中

局部变量声明在方法中，没有初始值，所以必须初始化，如果在变量被初始之前就使用的话编译器会报错

==可用来比较两个主数据类型，或用于判断两个引用是否引用同一个对象

equals（）来判断两个对象在意义上是否相等

遵循良好的面向对象编程，不要让单一的类执行太多的任务

可以按照伪代码、测试码、真实码的顺序开发。极限编程了解一下！

++和--前置（++x）和后置（x++）要注意运算顺序，前置时先++在引用x，后置时先引用再++。如int x=0;int z=++x;则x和z都是1；intx=0;int z=x++则z=0，x=1

加强版的for循环是for(String name:nameArray){}其中：表示in的意思，依次将nameArray中的元素赋值给name变量，要注意name的类型要和nameArray中的元素类型一致

Interger.parseInt方法能将String转换为int类

一般数组在创建时需要确定大小，而ArrayList不需要，它是Java API中的一个类，可以自动调整大小

java函数库中的每个类都属于某个包，类由完整的名称，由包的名称和类的名称所组成，如java.util.ArrayList。除了java.lang包之外，其他包的类使用时都要指定全名，也可以用import指令

来说明所使用的包。import只是帮助省下每个类前面的包名称而已，不会使程序变大或变慢，与c语言的include不同

注意参数的传递，尤其是参数多的时候，注意参数的命名和大小写的规范

为不同的类设置合理的作用

子类会自动获得父类的功能，子类可以重新定义继承下来的方法来改变或者延伸此方法的行为

类的成员指的就是实例变量和方法

当有多重的继承关系时，调用对象引用的方法会调用到与该类型最接近的方法。若a是b的父类，b是c的父类，则调用c的方法时优先级为c，b，a。

判断一物x是否应该继承自另一物y时用IS-A来检验，看x is a y对不对,隐喻着x能做到y可以做的任何事情，并且可能会做出更多的行为

父类可以通过存取权限来决定子类能否继承某些特定的成员。限制程度由大到小为private、default、protected、public。public类型的成员会被继承，可以被任何代码存取，private类型的不会被继承，只有同一类的程序代码可以存取，是对类设限而不是对对象设限

default和protected权限是作用在包上的，default表示只能被同一个包上的类存取，protected和default等级基本一样，只有一处不同就是允许在不同包中的子类继承父类

继承下来的方法可以被覆盖，但是实例变量不会被覆盖

IS-A关系是单向的

继承的意义在于避免了重复的程序代码，定义出了共同的协议（父类）

final修饰符修饰类时表示是它的继承树的末端，不能被继承，修饰方法时这个方法不会被覆盖

多态下引用和对象可以是不同类型，引用类型可以是实际对象类型的父类。如 Animal myDog=new Dog（）；

参数传递时指定的是Animal类型的话我们也可以传一个Dog类型参数进去，多态参数就是将参数声明成父类类型，这样可以在运行时传入任何子类对象

子类覆盖父类的方法时要遵循规则：参数必须一样，返回类型必须要兼容（子类覆盖的方法的返回类型必须是父类方法返回类型的子类，即要保证能执行父类的一切），不能降低方法的存取权限

方法的重载和继承还有多态毫无关系，与覆盖方法也不一样。重载只是刚好有相同名字的不同方法。

重载的参数一定且必须不同，返回类型可以不同，，可以更改存取权限

抽象类（关键字 abstract），不能被初始化，即不能被new出来，但可以被作为引用类型，抽象类除了被继承过之外，是没有用途，没有值，没有目的的

抽象类可以带有抽象的和非抽象的方法

接口是100%的抽象类，即接口的所有方法都是抽象的

方法也可以是抽象的，抽象方法没有实体，（如 public abstract void eat();没有方法体）,有抽象方法的类一定要标记为抽象类

所有的类都是继承自java.lang.Object类，Arraylist的方法都是使用Object这个终极类，所以可以处理任何类

编译器是根据引用类型来判断有哪些method可以调用，而不是根据对象确实的类型，所以当ArrayList<Object>中装有对象时，不管这个对象原来是什么，只能把它当Object，取出引用时引用的类型也只能是Object，但是可以进行类型转换，如：Object o=al.get(index);Dog d=(Dog) o;d.roam;如果这个arraylist中的对象o真的是一个Dog对象，则可以通过这种转换来引用Dog中的方法

Java极其重视引用变量的类型！只有在引用变量的类确实有该方法时才能调用它

java中只能继承一个父类，不允许多重继承因为会有致命方块那样的问题，但是可以实现（implements）多个接口，因此接口的作用使得可以继承超过一个以上的来源

实现某个接口的类必须实现它所有的方法，不然就是抽象类，因为interface的所有方法都是abstract和public的

使用super关键词可以调用父类中的方法和变量

子类继承父类对接口方法的实现

对象存放在堆中，也就是可垃圾回收的堆

方法调用和局部变量存放在栈中，如果局部变量的类型是对象的话，则变量引用存放在栈上，而对象仍然存放在堆中

方法a调用方法b，方法b调用方法c，则栈中先放入a，再在a上放b，再在b上放c，c调用完毕释放，b调用完毕释放，然后a调用完毕再释放

对象的实例变量存放在堆中，给对象分配堆中的空间时会为实例变量留下足够的空间

构造函数名字和类相同，且没有返回类型，我们在new一个函数时就是在调用构造函数，构造函数不会被继承，java中可以有与类同名的方法却不会变成构造函数，差别在于有返回类型

构造函数也可以重载，当我们没有写构造函数时编译器会帮忙写出没有参数的构造函数，而如果我们写了构造函数编译器就不会写默认的构造函数了。

如果类有一个以上的构造函数，那么参数一定要不一样

在创建对象时，所有继承下来的构造函数都会执行，继承关系层次多时，会从最近的父类的构造函数一路调用至Object类的构造函数，然后从Object的构造函数开始逐渐往下完成构造函数的调用

因为小孩不能在父母之前出生，所以父类部分必须在子类创建完成之前完整成型

如果没写构造函数，编译器会加上没有参数的构造函数，如果没有super（）也就是父类的构造函数，编译器会加上没有参数的super（）也就是没有参数版本的父类构造函数

构造函数的第一个语句一定是super（）或this（）；且两者不可兼得。this（）用来从某个构造函数调用同一个类的另外一个构造函数

局部变量只会存活在声明该变量的方法中，实例变量的寿命和对象相同。对象生命周期要看引用变量的生命周期而定，当最后一个引用消失时，对象就会变成可回收的

boolean类型默认值是false

静态static方法指不需要依靠实例变量的行为，因此也不用创建对象，如Math中的方法

静态方法要用类名来调用，而不是对象引用变量来调用

静态方法不关心类的具体实例，也无法看到实例变量的状态。

静态变量会在该类的任何静态方法执行前就初始化。jvm会在有人尝试创建类的新实例或使用该类的静态方法或变量时加载类。静态变量就只在类被加载时初始化！静态变量和静态方法是在加载类的时候就执行的，非静态变量和方法还要实例化对象才能调用，此时还没生成，所以静态方法不能直接调用非静态方法，而可以通过在静态方法中实例化对象再通过对象调用非静态方法，如public class TT{

public static void main(String[] args){//不能直接调用fun方法，需要借助对象

TT t = new TT();

t.fun();

}

void fun(){

System.out.println("hehe");

}

}

实例变量是每个实例一份，而静态变量是每个类一个，同一个类的所有实例（对象）共享一份静态变量，就是整个类的所有对象共用这一份拷贝，因此静态变量的值对所有的实例来说都相同

final变量表示一旦被初始化就不会改动。静态的final变量是常数，常数变量名称应该都是大写字母，如Math.PI

静态final变量没有初始化会报错，所以必须在声明或静态初始化程序中赋值

如果类只有静态方法（不需要实例），可以把类的构造函数标记为private以避免被初始化

早期的java（5.0之前）如果要以对象方式处理primitive主数据类型，就要包装起来，如int i =22；Integer iWrap=new Integer（i）；

包装有静态的实用性方法，如Integer.parseInt（）方法把String转成int

现在的java有了autoboxing功能后就不用将primitive主数据类型和对应的对象分的那么清楚了，在参数啊返回值啊boolean表达式啊数值运算和赋值啊等一系列操作中例如int数据类型和Integer类型对象两者可以通用

+号是java中唯一重载过的运算符，可用于字符串的接续

java中数字格式化结构%与c类似，注意要把类型如d或f等摆在最后

日期格式化有自己的格式，注意用Calendar（日历）来进行有关时间的操作

静态方法在构造函数之前执行，见310页的习题

静态初始化程序为static{}，记住没有（）！

异常是一种Exception类型的对象，因此也是多态的

会抛出异常的方法必须要声明它有可能会这么做

方法可以抓住其他方法所抛出的异常，异常总是会丢回给调用方

Exception子类中不是RuntimeException子类的叫做检查异常，会由编译器进行检查

RuntimeException称为不检查异常，主要是一些逻辑异常

可能会抛出异常的方法必须用throws Exception来声明，可以声明多种异常

如果程序调用了已经声明会抛出异常的方法，就要告诉编译器已经注意到了这件事，一种方法是用try，catch块

另外一种方法是声明（duck回避掉），例如方法a调用b，方法b调用c，若方法c会抛出异常，则c和所有调用它的方法即abc都要声明异常。一旦c出现异常，c从栈中弹出，将异常抛给b，看b能否catch异常，若不能则b也弹出，将异常交给a，以此类推，即duck是通过声明但不处理异常的形式将处理异常的任务交给调用它的方法，实际是在踢皮球

如果try块失败了，try块中失败语句后续的部分就不会执行

finally块用来存放不管有没有异常都要执行的程序

如果try和catch块中有return指令，finally还是会执行，并且在finally执行完后回到return指令

catch块是从上往下依次执行指令的，因此注意异常的继承关系，避免有的语句根本不会执行到

try一定要有catch或finally，只带有finally的try必须要声明异常

弄清事件源（如按钮）、事件（事件对象，如鼠标被按下）和监听接口的关系

在事件源中添加（注册）监听接口（实际是个继承监听接口的对象），然后事件源中在发生事件时调用监听接口的事件处理方法

事件对象包含事件的信息

Graphics g所引用的对象实际上是Graphics2D（Graphics 的子类）的实例

内部类是嵌套在外部类的内部的。内部类可以使用外部类所有的方法和变量，就算是private的也一样，轻松的就像开自家冰箱

内部类好用的原因就在于多了这特殊的存取权

内部类的实例一定会绑在外部类的实例上，我们讨论的特殊的存取权是发生在外部对象和对应的内部对象之间的

内部类可以在某个类需要实现同一个接口多次等情形下使用，一个外部类可以有多个内部类，但一个内部类只能绑定一个外部类

布局管理器控制组件的大小和位置

组件（component）是可以嵌套的，例如JFrame a上有JPane b1、b2，JPane b1上有按钮c1，b2上有按钮c2等，这时布局管理器（Layout Managers）的运行流程是这样的：a的布局管理器管a上的b1和b2，b1的布局管理器管b1上的c1，b2上的布局管理器管b2上的c2，类似我的附庸的附庸不是我的附庸！运行时b1和b2的布局管理器先和c1、c2商量来布局c1和c2，然后a的布局管理器和b1、b2商量来布局b1和b2，所以布局的步骤是从小到大的。

布局管理器有三种主流的：Borderlayout（分东西南北中）、Flowlayout（从左往右，空间不够了往下换行）、Boxlayout（从上到下垂直排列）

JFrame框架默认用Borderlayout，Panel面板默认用Flowlayout布局，默认布局可以更改或直接删掉

若a上嵌着b，则b.setSize不起作用，因为a的布局管理器会调用别的方法

深刻理解行为存在于类中，状态存在于对象中

在不考虑数据库的情况下

数据的存储：一：如果只有自己的java程序会用到这些数据，可以用序列化（serialization）将序列化的对象写到文件中，程序用时能把对象再展开

二：如果数据需要被其他程序引用，可以写一个纯文本文件，用其他程序能解析的特殊字符写到文件中。

对象的序列化要必须要保证它能重新展开回复原来的状态，要保存对象的所有实例变量，所以序列化程序会把对象版图上的所有东西都存储起来。被对象的实例变量所引用的所有对象都会被序列化。

要让类能被序列化，就implements Serializable接口，这个接口不用实现任何方法，只需要声明即可。由接口机制可知实现这个接口的类的子类也能序列化。

序列化是全有全无的，整个对象版图都必须被正确序列化，否则就全部失败，因为只有部分状态被正确保存就很畸形，跟无脸人一样。

若对象a要序列化，但是a的某个实例变量b不能被序列化（比如存储b没有意义，因此b没有设置成能序列化），则可以将变量b添加关键字transient（瞬时的），这样a在序列化时会跳过b。transient的引用实例变量会以null或0、false等默认值返回，这代表整个对象版图中连接到这个特定实例变量的部分不会被存储。

deserialization（解序列化）用于还原对象，被还原的对象a会被配置在堆上，但不会执行构造函数（因为执行了就会把原来对象的状态抹去又变成新的）。要注意的是若解序列对象a的继承树上有某个不可序列化的祖先类x，则x和x之上的祖先类都会执行构造函数，而不管x的祖先类能否序列化

序列化对象不存储类的信息因为减少存储量。静态变量是每个类一个的，所以不会被序列化，对象被还原不会影响静态变量

序列化写入文件或解序列化读出都需要用到串流，用ObjectOutputStream将对象序列化，再用FileOutputStream写入文件（想象成两个串流在程序和文件间是串连工作的），用FileInputStream用来读文件，再用ObjectInputStream来将对象恢复

FileOutputStream和FileIntputStream是文件读写的链接，编程时可以自由设置用ObjectOutputStream、BufferedWriter等连接来和链接相连，完成文件读写

File对象代表磁盘上的文件或目录的路径名称，但不能读取或代表文件中的数据，可用于创建、浏览、删除目录

大部分用String文件名的串流都可以用File对象代替

缓冲区的好处在于可以等缓冲区满了再一并写入，避免多次的磁盘操作（磁盘操作比内存操作耗时大），提升效率

java的IO操作要放到try/catch块中执行

eclipse的快捷键：选中包、类或变量再按alt+shift+r可以重命名包、类、变量，其他用到这个包、类、变量的地方会自动更新

ctrl+d删除某行

自动生成getter和setter方法：右击空白处，选择【Source】,再选择【Generate Getters and Setters】,然后选择自己对哪些属性生成这样的方法

类有版本id称为serialVersionUID，老狗变不出新把戏，当对象被序列化时对象有serialVersionUID，若类有变动会有新的版本id，解序列化时只有对象和类的serialVersionUID一致才不会抛出异常。可以在类中人工设置serialVersionUID来处理旧的对象，但要确定修改程序的后果。

只要其中的元素是可序列化的，数组就是可序列化的

Socket是代表两台机器之间网络连接的对象，客户端建立socket连接需要知道服务器的ip和端口号

端口号是逻辑上用来表示应用程序的数字，取值范围为0~65535，0~1023的端口号都保留给了已知的特定服务

利用socket来读写数据和之前的串流并没有什么不同，同样要注意不同串流的连接使用

建立BufferedReader连接InputStreamReader（用于将字节转换为字符）和Socket的输入串流{socketname.getInputStream()可以从socket中取得低层串流（二进制流）}可以读取服务器的文本数据

用PrintWriter连接Socket的输出串流可以送出String给服务器

服务器使用ServerSocket来监听用户对特定端口的请求，接到用户的socket连接请求时会做一个新的socket连接来接收用户请求

Thread是java中用来表示线程的类，要建立线程就要创建Thread。线程是独立的线程，代表独立的执行空间

每个java应用程序会启动一个主线程，并且把main（）放在自己执行空间的最开始处

Runnable这个接口只有一个piblic void run（）的方法

对于Thread而言，它是工人，Runnable就是这个人的工作，要把Runnable传给Thread的构造函数才能启动新的线程

只有调用线程名.start()方法才会让这个线程进入可执行状态，在此之前这只是一个线程的实例，并不是真正的线程

一旦线程进入可执行状态，它会在可执行和执行中来回切换，有时会暂时不可执行（被堵塞状态），造成暂时不可执行的原因可能是线程要求sleep（）、线程调用某个被锁住（locked）的对象上的方法等等。

没有API可以调用线程调度器。不同的调度器有不同的行为，所以多线程程序的测试不能在单一的机器上进行。

线程执行完run方法后就不能再重新启动了，失去了线程的执行性而只剩下线程对象本身

两个以上的线程存取同一个对象的数据时会引起并发性的问题，竞争状态可能会导致数据的损毁。

使用synchronized关键词来修饰方法使它每次只能被单一的线程存取

每个Java对象都有一个锁，并且只有一把钥匙。通常对象没有上锁，并且也没有人在乎这件事。但是如果对象有同步化的方法，则线程只有在取得钥匙的情况下进入线程，也就是说没有其他线程已经进入的情况下才能进入。

synchronized同步化的目标是为了保护重要的数据，但是锁住的是存取数据的方法而不是数据

同步化可以不同步化整个方法而只同步化其中的几行代码，这样可以让被同步化的代码运行起来像原子一样不可分割（即运行中不会被干扰）

同步化的意义就是要让指定的某段工作在不能分割的状态下执行

类也有锁，所以要对静态方法做同步化时会使用到类本身的锁

泛型主要用于处理集合，目的是为了更好的类型安全性，可理解为类型作为参数，例如ArrayList<String> songList=new ArrayList<String>();中的<String>。这样当songList中add一个其他类型的元素时编译器会报错，提高了安全性。

泛型的类代表类的声明用到类型参数，泛型的方法代表方法的声明用到类型参数，ArrayList是类

注意对于泛型而言，extends表示“是一个”，此时适用于接口和类，与之前对extends的理解不同

public String toString()返回该对象的字符串表示。通常，ToString方法会返回一个“以文本方式表示”此对象的字符串。结果应是一个简明但易于读懂的信息表达式。可用于打印出对象

集合中的元素排序必须满足以下两点其中之一：

1.元素是实现Comparable的类型（有compareTo方法）

2.采用含有Comparator参数（有compare方法）的构造函数来构造集合

Collection框架中有三个主要的接口：Set、List和Map

Set：类似数学中集合的概念，不允许元素重复，且至多含有一个null的元素

List：是有序的，允许元素重复可含有多个null元素，可通过index来操作元素

Map：使用成对的键值和数据值，两个key可以引用相同的对象，但key不能重复

大部分java版本是根据对象内存位置计算hashcode，所以不会有相同的hashcode

Set中判断元素重复的标准是两个对象引用变量所引用的对象是equals的并且有相同的hashcode

所以元素重复的情况可以是两个对象引用变量引用到同一个对象，或两个对象引用变量引用到两个在意义上相同且hashcode也相同的对象

如果equals（）被覆盖过，则hashCode（）也必须被覆盖

equals（）的默认行为是执行==的比较，即测试两个引用是否对上heap中的同一个对象，如果equals（）没有被覆盖过，则两个对象永远不会被视为相同的，因为不同的对象有不同的字节组合。

a.equals(b)必须与a.hashCode()==b.hashCode等值

TreeSet是排序的Set，所以集合中的元素也要满足上述的两点其中之一

数组的类型是在执行期间检查的，集合类型检查只会发生在编译期间，代码见571页

所以对方法声明中取用参数ArrayList<Animal>,它就只会取用ArrayList<Animal>，若传入ArrayList<Dog>或ArrayList<Cat>就会在编译时报错，

若方法声明中取用参数Animal[],则它只会取用Animal[],若传入Dog[]或Cat[],则能通过编译但会在执行时报错

比如我们想象在方法是在Animal数组或集合中加入一个pig，则传入Dog的数组或集合都会出现错误

在方法参数中使用万用字符时，编译器会阻止任何可能破坏引用参数所指向集合的行为，即能调用元素但不能新增元素

命令行中编译时-d选项能指定编译后的类文件存放位置

eclipse中默认java文件（源代码）放在src中，class文件（编译后的二进制文件）放在bin中

可以用manifest.txt文件来描述哪个类带有main方法，将所有class文件和manifest.txt文件都包进可执行的JAR中是大部分完全在本机的Java应用程序部署的方式

把类放在包中防止重名，防止包重名的方法包括将域名写入包名中

类和包的文件目录结构应该与代码的逻辑结构一致，即类必须呆在完全对应于包结构的目录中才能包进包里

将项目组织一下以分开源代码和类文件到不同目录下

Java Web Start（JWS）是通过网络来发布的应用程序，用户通过浏览器下载后与可执行的JAR就一样了

客户端对象看起来像是在调用远程服务对象，实际上是在调用本地处理Socket和串流细节的代理（proxy）

Java的RMI（remote method invocation）会提供客户端辅助设施对象（stub）和服务器端的辅助设施对象（skeleton）

接口可以extends接口

若父类的构造函数声明了异常，则子类的构造函数也要声明异常，因为会调用父类构造函数

创建远程服务的过程有5个步骤：1创建Remote接口2实现Remote3用rmic产生stub和skeleton4启动RMI registry5启动远程服务

该过程就是一套流程，具体代码见623页

servlet是完全在HTTP服务器运行的Java

数组本身只是对象，Java中二维数组只是个数组的数组

枚举enum就是一种特殊的类

boolean只需要一位空间，但是计算机处理数据最小单位是一个字节，虚拟机中boolean转换为int处理，用四个字节存储。

byte是一个字节，-128~127

short两个字节，-32768~32767

int是四个字节，-2^31~2^31-1

long是八个字节

float是四个字节

double是八个字节

char是两个字节