# 第一章

## 1、作为程序员要求掌握最基本的dos命令：

### \*在DOS命令窗口中操作先写DOS命令：

开始菜单>运行，键入cmd，吊起命令窗口-------这是win7系统的找到命令窗口的方式；

我当前用的是win10版本的电脑，所以需要win+X ，然后找到运行，然后键入cmd，吊起命令窗口即可；

win+R 就可以吊起运行窗口，键入cmd，就可以吊起命令行窗口；

### \*常见的DOS命令：

-- exit 退出当前的dos命令窗口

-- cls 清屏

-- dir 列出当前目录下所有的子文件和子目录

-- cd 这个不用多说

-- 绝对路径：表示该路径从某个磁盘的盘符下作为出发点的路径；

-- 相对路径：表示改路径从当前所在的路径下作为出发点的路径；

-- cd \ 直接回到根目录

## 2、JDK：什么是JDK：

\* Java开发工具包，做Java开发必须安装的一个工具包，该工具包需要从官网下载，目前SUN被\* \* Oracle收购了，所以下载需要去oracle下载：[www.oracle.com](http://www.oracle.com)

\* 目前JDK最高版本是：Java8/JDK1.8/JDK8

\* Java包括三大块：

-- JavaSE（Java标准版）

-- JavaEE（Java企业版）

-- JavaME（Java微型版）

其中JavaSE是基础，以后主攻方向是JavaEE方向。

## 3、Java语言特性

\* 简单性：相对而言的，例如java中不再支持多继承，c++是支持多继承的，多继承比较复杂， c++中有指针，Java中屏蔽了指针的概念。所以相对来说Java是简单的。

Java语言底层是c++实现的，不是c语言。

\* **面向对象：Java是纯面向对象的，更符合人的思维模式，更容易理解**。

\* **可移植性：Java程序可以做到一次编译，到处运行。也就是说java程序可以在windows操作系统中运行，不做任何修改，同样的java程序可以直接在Linux系统中运行，可移植性又被叫做跨平台。**

**win系统内核和Linux系统的内核肯定不同，这两个系统执行指令的方式也是不一样的。**

**结论：显然Java程序不能直接和操作系统打交道，因为java程序只有一份，操作系统执行原理都不同。**

**sun团队的处理办法：他们让Java程序运行在一台虚拟计算机当中，这个虚拟的计算机叫做Java虚拟机，简称JVM。Java虚拟机再和底层的操作系统打交道。**

\* **多线程**

\* 健壮性：和自动垃圾回收机制有关，简称GC机制。Java语言运行过程中产生的垃圾是自动回收的，不需要程序关心。

\* 安全性

........

## 4、JAVA的加载与执行

\* Java程序的运行包括两个非常重要的阶段

-- 编译阶段

-- 运行阶段

\* 编译阶段

-- 编译阶段主要的任务时间差Java源程序是否符合Java语法，

符合Java语法则能够生成正常的字节码文件（xxx.class）

不符合Java语法规则则无法生成字节码文件。

-- 字节码文件中不是纯粹的二进制，这种文件无法在操作系统中直接执行。

-- 编译阶段的过程：

-- 程序员需要在硬盘的某个位置；（位置随意）新建一个.java扩展名的文件，该文件被称为Java源文件，源文件当中编写的Java源程序/源代码。而这个源程序是不能随意编写，必须符合Java语法规则（语法规则是需要记忆的）

-- Java程序员需要使用JDK当中自带的Javac.exe 命令进行java的编译。

Javac如何使用，在哪里使用？

在DOC命令窗口中使用。Javac的使用规则：Javac Java源文件的路径。

Javac是一个java编译器工具/命令。

\* 一个java源文件可以编译生成多个.class文件。

\* 字节码文件/class 文件是最终要执行的文件，所以说class文件生成之后，java源文件删除并不会影响java程序的执行。但是一般java源程序不要删除，因为class文件最终执行效果可能不是我们想要的，那么这个时候需要回头再重新修改java源程序，然后将java源文件重新编译生成新的class文件，然后在运行这个class程序，生成新的效果。

\* 编译结束之后，可以将class文件拷贝到其他操作系统中运行。

\* 运行阶段【可以在其他操作系统当中，跨平台】

-- JDK安装之后，除了自带一个javac.exe之外，还有另外一个工具：java.exe

java.exe命令主要负责运行阶段。

-- java.exe在哪里用？怎么用？

在DOS窗口中使用。

java.exe怎么用？

java 类名

例如：硬盘上有一个A.class，那么执行命令： java A

-- 运行阶段的过程：

\* 打开DOS命令窗口

\* 输入：java A

\* java.exe命令会启动Java虚拟机（JVM），JVM会启动类加载器，classLoader

\* ClassLoader会去硬盘上搜索A.class文件，找到该文件则将该字节码文件装在到JVM当中

\* JVM将A.class字节码文件解释成二进制101010这样的数据

\* 然后操作系统执行二进制和底层硬件平台进行交互。

## 5、开始第一个Java程序

\* 保证计算机当中已经安装了文本编辑器

\* 安装JDK【JDK一般需要从oracle的官网下载】，这里讲解的是JDK7

\* JDK、JRE、JVM的关系搞清楚。

\* JDK目录的介绍：

JDK/bin：该目录下存放了很多命令，例如javac.exe 和 java.exe

javac.exe 负责编译

java.exe负责运行

\* 开发HelloWorld.java源程序【照抄，不问为什么】

\* 将HelloWorld.java源程序通过javac工具进行编译。

-- 首先需要解决的问题是：javac命令是否可用。

test：在命令行窗口中输入javac -version

看打印出什么结果，如果是打印出版本号，那么就是已经安装了javac；如果找不到这个命令，那么就是没有配置环境变量；

-- javac要想能随意使用，需要将javac所在的目录配置到环境变量path中：找到java所在的目录，然后复制路径，然后右键我的电脑，进入到高级系统设置，找到系统环境变量，然后新建目录，粘贴此新的目录，确定，退出DOS命令窗口，重新打开就解决上面的问题了。

\* javac 命令怎么用：

javac java源文件路径-----------------注意：前面一定得是javac

注意：路径包括相对路径和绝对路径，都可以。

\* 运行java程序：

- 需要使用java.exe命令

- 首先测试java命令是否可用。

- 使用方式：

java 类名

硬盘上有HelloWorld.class 那么类名就是HelloWorld

敲入：java HelloWorld

一定要注意：java命令后面跟的不是文件路径，是一个类的名字

- 首先需要先将DOS窗口中的目录切换到HelloWorld.class文件所在目录。

- 然后直接执行：java HelloWorld

字符串打印成功。

至此，第一个程序编译运行成功；

## 6、关于java源程序当中的注释：

\* 什么是注释？注释的作用是什么？

-- 出现在java源程序当中，对java源代码的解释说明；

-- 注释不会被编译到.class字节码文件当中

-- 一个号的开发习惯，应该是多编写注释，这样程序的可读性增强；

\* java中的注释写法规则：

-- 单行注释

// 单行注释，只注释当前行

-- 多行注释

/\*

多行注释

多行注释

\*/

-- javadoc注释

/\*\*

\* javadoc注释

\* javadoc注释

\* avadoc注释

\*/

注意：这种注释是比较专业的注释，该注释信息会被javadoc.exe 工具解析提取并帮助生成文档。

## 2020.03.25学习笔记

解决今天之前学习过程中产生的两个问题：

1、在编译的过程中，因为是用sublime工具编写的java程序，那么在编译文件的过程中，没有用ANSI转码，导致代码中含有中文的时候或者注释中有中文字符的标点的时候，编译不成功。

报错内容为：错误: 编码GBK的不可映射字符

解决办法：执行命令：javac -encoding UTF-8 文件名.java

完美解决。

但是这是要附加命令内容的，解决办法2：更换编写程序的工具，网上有关于nodpad++ 的工具，会自动保存为ANSI格式编码的文件，那么在编译过程中就不会报错了，因为自己电脑中没有使用该工具，所以没有实践是否成功解决该问题。

还有就是尝试了网上说的将sublime增加插件：crtl+shift+P ，然后install package，然后搜索ConvertToUTF8，下载并自动安装，完成后，但是重新编译java程序还是会报错。

所以先暂时用命令行来解决编译出错的问题。

2、在执行test.java文件时，报错：类HelloWorld是公共的，应在名为HelloWorld.java的文件中声明

百度之后有了答案：java源代码文件名错了，应该与声明的class类一致，将firstjava.java改为HelloWorld.java就可以了。但如果声明的class非public则文件名无需与类名相同。

## 7、public class 和class 的区别

\* 一个java源程序中可以定义多个class；

\* 一个jiava源文件中不一定有public 的class；

\* 一个class会定义生成一个xxx.class字节码文件

\* 一个java源文件当中定义公开类的话，public的class只能有一个，并且该类名称必须和java源文件名称一致；

\* 每一个class当中都可以编写main方法，都可以设定程序的入口，例如执行 B.class中的main方法，执行命令：java B 例如执行C.class中的main方法，执行命令：java C

\*注意：当在命令行中执行java Hello，那么要求Hello.java中必须要有Hello主方法，没有主方法运行就会报错。

## 0325总结第一章需要掌握的内容：

\*理解Java的加载与执行；

\*能够自己搭建java的开发环境；

\*能够独立编写HelloWorld程序，编译并运行；

\*掌握环境变量path的原理以及如何配置；

\*掌握环境变量classpath的原理以及如何配置；

\*java的注释；

\*public class 与 class 的区别

# 第二章：编程语言基础

## 1、标识符

### 1.1、什么是标识符？

-在java源程序中凡是程序员自己有权利命名的单词都是标识符。

-标识符在sublime编辑器中以绿色高亮展示；

-标识符可以标识什么元素呢？

\*类名

\*方法名

\*变量名

\*接口名

\*常量名

......

### 1.2、标识符的命名规则？

【不按照这个规则来，编译器就会报错，这是语法】

\*一个合法的标识符只能由数字、字母、下划线\_、美元符号$组成，不能含有其他符号；

\*不能数字开头；

\*严格区分大小写；

\*关键字不能做标识符；例如public、class、main等其他语法单词

......

### 1.3、标识符的命名规范？

【只是一种规范，不属于语法，不遵守规范编译器不会报错】

\*最好见名知意，举例：类名，方法名不能随便取，最好是什么服务，什么功能就取什么样的名字；

public UserService{

public void login(String username,String password){

}

}

\*遵守驼峰命名方式

SystemService

UserService

CustermorService

\*类名，接口名：首字母大写，后面每个单词首字母大写。

\*变量名，方法名：首字母小写，后面每个单词首字母大写。

\*常量名：全部大写

## 2、关键字

常见的关键字有哪些？

public public public public public public

class

static

void

int

char

if

else

while

do

default

byte

short

long

float

double

boolean

private

protected

swich

break

public

public

public

public

## 3、字面值

5、字面值：

\*字面值就是数据。

\*字面值是java源程序中的重要组成部分之一。包括标识符和关键字它们都是java源程序中的组成部分。

\*数据在现实世界中是分门别类的，所以数据在计算机编程语言中也是有类型的：【数据类型】

-10、100 是属于整数型字面值

-3.14 是属于浮点型字面值

-true、false 是属于布尔型字面值

-"abc"、"中国人" 是属于字符串型字面值

-'A'、'人' 是属于字符型字面值、

注意：

java语言当中所有的字符串数据类型都要用双引号括起来，双引号是半角；

java语言中所有的字符型数据类型都要用单引号括起来，单引号是半角；

示例代码：

public class ElementOfGrammar{ // ElementOfGrammar 就是一个类名，名字可以修改。

// main是一个方法名

public static void main(String[] args){

System.out.println("知识点都在注释里，要打开文件去看哦~~");

System.out.println("这个文件主要讲解的是java语法基础：包括标识符，标识符的命名规则，标识符的命名规范；关键字等等。");

System.out.println("这些东西都需要去记住，死记硬背的记住先。");

System.out.println("老婆做的小吊梨汤好好喝呀~~~");

}

}

## 4、变量

### 4.1

/\*

关于java语言中的变量

#### 1、什么是变量？

\*变量本质上来说是内存中的一块空间，这块空间有 数据类型，有 名字，有 字面值；

\*变量包涵三部分：数据类型、名称、字面值【数据】；

\*变量是内存中存储数据的最基本的单元；

#### 2、数据类型的作用？

\*不同的数据有不同的类型，不同的数据类型底层会分配不同大小的空间；

\*数据类型是知道程序在运行阶段应该分配多大的内存空间；

#### 3、变量要求：

变量中存储的具体数据必须要变量的数据类型一致，当不一致的时候，编译会报错。

#### 4、声明/定义变量的语法格式：

数据类型+空格+变量名;

\*数据类型

目前我们还没有学习数据类型，但是可以提前学习一种数据类型--整数型：int

\*变量名：

只要是合法的标识符就行。规范中要求：首字母小写，后面每个单词首字母大写；

例如：

int i;

int peopleAge;

int length;

int numOfBall;

其中，int是数据类型，i/peopleAge/length/numOfBall都是变量名称；

#### 5、变量声明之后，如何赋值？

语法格式：

变量名 = 字面值;

要求：字面值的数据类型必须要和变量的数据类型一致！

= 等号是一个运算符，叫做赋值运算符，赋值运算符先计算等号右边的表达式，表达式执行之后然后赋值给等号左边的变量。

#### 6、声明和赋值可以放到一起完成。

例如：int i = 10;

#### 7、变量赋值后，可以重新赋值，变量的值可变化：

int i = 10;

System.out.println(i);

i = 100;

System.out.println(i);

i =1000;

System.out.println(i);

#### 8、有了变量的概念后，内存空间得到了重复的使用；

int i = 10;

System.out.println(i);

...

...

...

System.out.println(i);

#### 9、通常访问一个变量包括两种访问形式；

\*第一种：读取变量中的具体数据 get/读取

\*第二种：修改变量中的具体数据 set/设置

例如：

int i = 10; //set

System.out.println(i); //get

#### 10、变量在一行中可以声明多个；

int a,b,c;

11、变量必须先声明，再赋值才能访问。

int i ; 程序执行到这里，内存空间并没有开辟出来，变量i并没有初始化。所以没有赋值之前是不能访问成功的。

\*/

public class VarTest01{

public static void main(String[] args){

// 声明一个变量

int i ;

// 给变量赋值；i变量在这里完成初始化，内存开辟；

i = 100;

// 将变量打印出来

System.out.println(i);

// 再次给变量赋值

i= 2020;

//打印变量

System.out.println(i);

//一行中声明多个变量；

int a,b,c=1999;

a = 18;

b = 19;

System.out.println(a);

System.out.println(b);

System.out.println(c);

}

}

### 4.2

/\*

关于java中的变量：

#### 1、在方法体中的java代码，是遵守自上而下的顺序依次执行的。逐行执行。

第一行；

第二行；

第三行；

特点：第二行的代码必须完整的执行结束后，第三行的代码才能执行。

#### 2、在同一个“作用域”当中，变量名不能重名，但是变量名可以重新赋值；

\*/

public class VarTest02{

public static void main(String[] args){

//声明一个变量i

int i ;

//给变量i赋值；

i = 156;

//打印变量i

System.out.println(i);

//以下代码为错误示例：顺序编写错误，应该先声明，然后赋值，再访问；

/\*

System.out.println(k);

int k;

k =177;

\*/

int i =188;

System.out.println(i);

}

}

### 4.3

/\*

关于java中的变量：

变量中的作用域

#### 1、什么是作用域？

变量的作用域，其实描述的就是变量的有效范围；

在什么范围之内是可以访问的，只要出了这个范围就不能访问了。

#### 2、变量的作用域就只用一句话概括：

出了大括号就不认识了。

\*/

public class VarTest03{

// 注意：这里的static不能去掉

static int k =19;

public static void main(String[] args){

//声明一个变量i

int i ;

//给变量赋值

i = 18;

//打印变量i的赋值

System.out.println(i); //这里打印是可以的

System.out.println(k); //这里打印是可以的

//以下会编写一个for循环语句

//这个for循环后面的控制语句的时候详细讲解；

for(int a =10;a<10;a++){ //a变量的作用域是整个for循环，for循环结束之后，a变量的内存就释放了

}

//这里无法访问a变量

//System.out.println(a); //这里打印是不可以的；

//这里再写一个for循环语句，这次的变量j写在for循环的外面，此事j的作用域是main方法

int j;

for(j=0;j<10;j++){

}

System.out.println(j);//这里编译是不会出错的；

}

public static void doSome(){

System.out.println(k);

}

}

### 4.4

/\*

#### 关于变量的分类：

根据变量声明的位置来分类：

\*局部变量

- 在方法体当中声明的变量叫局部变量

\*成员变量

- 在方法体外【类体之内】声明的变量叫做成员变量；

在不同的作用域当中，变量名是可以重名的；

在同一个作用域当中，变量名是不能重名的；

\*/

public class VarTest04{

//这里也属于成员变量；

int k=200;

public static void main(String[] args){

//i 就是局部变量

int i =10;

//java 遵循就近原则；

System.out.println(i);

}

public static void doSome(){

//这里也是局部变量

int j = 1999;

}

// 这里就是成员变量；

int i =100;

//类体中不能直接编写java语句【除声明变量之外】

//System.out.println(i); // 这一行会编译报错

}

# 数据类型：

/\*

关于java中的数据类型：

1、数据类型的作用是什么？

程序当中有很多数据，每一个数据都是有相关类型的，不同数据类型的数据占用空间大小不同。

数据类型的作用是知道JVM在运行程序的时候该数据分配多大的内存空间。

2、java中的数据类型包括两种：

\*基本数据类型

\*引用数据类型【后面再讲】

-类

-接口

-数组

-字符串

...

3、关于基本数据类型

\*基本数据类型分为四大类八小种：

第一类：整数型

byte,short, int, long

第二类：浮点型

float,double

第三类：布尔型

boolean

第四类：字符型

char

4、字符串"abc"不属于基本数据类型，属于引用数据类型，字符属于基本数据类型；

\*字符串使用双引号 "abc"

\*字符使用单引号 'a'

5、八种基本数据类型各自占用空间大小是多少？

基本数据类型 占用空间大小【单位：字节】

---------------------------------------------------------------

byte 1

short 2

int 4

long 8

float 4

double 8

boolean 1

char 2

6、计算机在任何情况下都只能识别二进制。例如：只认识10101010100010101011110101010111010101010......

【现代计算机底层采用交流电的方式，接通和断开就两种状态，计算机只识别1和0，其他不认识】

7、什么是二进制？

数据的一种表示形式。十进制表示满十进一原则，二进制表示满二进一原则。

例如：十进制

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16.....

二进制：

0 1 10 11 100 101 110 111 1000 1001...

8、字节（byte）：

1 byte = 8 bit （比特位） 【一个字节=8个比特位】 一个比特位表示一个二进制位：1/0

1 KB = 1024 byte

1 MB = 1024 KB

1 GB = 1024 MB

1 TB = 1024 GB

9、整数型当中的byte类型，占用1个字节，所以byte类型的数据占用8个比特位，那么byte类型的取值范围是什么？

\*/

# 上面的内容以后要补上。

# DAY11学习的内容回顾：

## 方法是什么？

方法是一段可以完成某个特定功能的并且可以被重复利用的代码片段。

方法的出现，让代码具有了很强的复用性。

## 方法最难实现的是：

根据业务怎么进行方法的抽取。

方法的返回值类型定义为 什么？

方法的名字叫什么？

方法的形式参数列表定义为 什么？

...

一个方法就是一个独立的功能。

## 方法的定义

[修饰符列表] 返回值类型 方法名(形式参数列表){

方法体；

}

## 方法的每一个细节学习：

4.1、修饰符列表：可选项，目前可写成：public static

4.2、怎么理解返回值？返回值是一个方法执行结束后返回的结果。

4.3、返回值类型都可以指定哪些类型？

4.4、返回值和“return”语句的关系。

4.5、方法名只要是合法的标识符就行，首字母小写，后面每个单词首字母大写，见名知意。

4.6、形式参数列表

4.7、方法体：方法体当中的代码遵循自上而下的顺序逐行执行。

5、JVM的内存结构中三块比较重要的空间

方法区：

存储代码片段，存储xxx.class字节码文件，这个空间是最先有数据的。

类加载器首先将代码加载到这里。

堆内存：

后面讲（面向对象的时候讲）

栈内存：

Stack当中存储什么？

每个方法执行时所需要的内存空间（局部变量）

## 关于数据结构中的栈数据结构：

原则：

先进后出；

后进先出；

栈数据结构相关的术语：

栈帧：永远指向栈顶部的元素（栈顶元素具有活跃权）。

栈顶元素

栈底元素

压栈，入栈，进栈，push

弹栈，出栈，pop

还讲到了数据结构：

有一本书：数据结构与算法；

数据结构和算法的选择很重要，选择对了对程序的执行效率大大提升，可以很好的优化程序。

1. 分析程序运行过程中的内存变化

方法只定义不调用是不会执行的。

方法调用时：压栈（在栈中给方法分配空间）

方法执行结束时：弹栈（将该方法占用的空间释放，局部变量的内存也释放掉）

# DAY12 学习回顾：

1. 方法重载overload

1.1什么情况下我们考虑方法重载机制？

当功能相似的时候，建议将方法定义为一致的。

这样代码既美观，又方便编程。

注意：如果功能不相似，坚决要让方法名不一致。

1.2、代码满足什么条件的时候构成了方法重载？

条件1：在同一个类中；

条件2：方法名相同；

条件3：形式参数列表不同（类型，个数，顺序）

注意：方法重载和返回值类型无关，和修饰符列表无关。

1.3、方法重载的优点？

代码美观，方便编程。

1. 方法递归

2.1、需要理解什么是方法递归？

方法自身调用自身。

2.2、使用递归的时候，必须添加结束条件，没有结束条件，会发生栈内存溢出错误。

StackOverflowError

原因：一直压栈，没有弹栈，栈内存不够用

2.3、会画出递归方法的内存结构图

递归的过程当中可以将图画出来。

2.4、能够使用循环代替递归的尽量使用循环，循环的执行耗费内存会少一些。递归耗费内存相对多一些，另 外递归使用不当很容易内存溢出，JVM停止工作。当然，极少数情况下，只能用递归，其他代码解决不 了问题。

2.5、当递归有结束条件，切结束条件合法的时候，就一定不会发生栈内存溢出吗？

也不一定，递归的太深了。

2.6、分享了一些递归方面的经验：

第一步：先检查结束条件是否正确。

第二步：如果正确，可以调整JVM的栈内存大小。（java -X）

1. 我们要一味地将变量缩减吗？代码缩减吗？

太计较变量的数量会有什么后果？（运行效率并不会变低）

后果1：代码的可读性会变差。

后果2：可读性差也会牵连到整个代码的开发效率。

其实计算机并不差这点内存.....

注意：在编码过程中，有一些变量是必须要定义的。

因为在后续的代码编码过程中还需要访问这个数据，重复的访问这个数据。

# DAY13课堂笔记：面向对象

## 1、面向过程和面向对象有什么区别？

从语言方面出发：

对于c语言来说，是完全面向过程的。

对于c++语言来说，是一半面向过程，一半是面向对象。（C++是半面向对象的。）

对于java语言来说，是完全面向对象的。

什么是面向过程的开发方式？

面向过程的开发方式主要的特点是：

注重步骤，注重的是实现这个功能的步骤。

第一步干什么；

第二步干什么；

…..

另外面向过程也注重实现功能的因果关系。

因为A所以有B；

因为B所以有C；

因为C所以有D;

….

面向过程中没有对象的概念。只是实现这个功能的步骤以及因果关系。

面向过程有什么缺点？（耦合度高，扩展力差。）

面向过程最主要是每一步与每一步的因果关系，其中A步骤因果关系到B步骤，A和B联合起来形成一个子模块，子模块和子模块质检又因为因果关系结合在一起，假设其中任何一个因果关系出现问题（错误），此时整个系统的运转都会出现问题（代码和代码质检的耦合度太高，扩展力太差）。

螺栓螺母拧在一起：耦合度高吗？

这个耦合度是低的，因为罗山螺母可以在拧开。（它们之间是 有接口的）

螺栓螺母宁在一起之后，在用焊条焊接在一起，耦合度高吗

这个耦合度就很高了，耦合度就是粘连程度。

耦合度高导致扩展力差，（集成显卡：计算机显卡不是独立的，是集成到主板上）

耦合度低导致扩展力强。（灯泡和灯口关系，螺栓螺母关系）

采用面向过程的方式开发一条计算机会怎样？

这台计算机将没有任何一个部件，所有的都是融合在一起的。

这台计算机僵尸一个实心儿的，没有部件的，一体机。

架设这台一体机的任何一个“部位”出问题，整个计算机就不能用了。

采用面向对象的方式开发一条计算机会是怎样？

内存条是一个对象，

主板是一个对象。

cpu是一个对象，

硬盘是一个对象，

然后这些对象组装在一起，形成一台计算机，假设其中CPU坏了，我们可以将cpu拆下来，换一个新的。

面向过程有什么优点？（快速开发）

对于小型项目（功能），采用面向过程的方式进行开发，效率较高。

不需要前期进行对象的提取，模型的建立，采用面向过程的方式可以直接开始干活

一上来直接写代码，编写因果关系。从而实现功能。

什么是面向对象的开发方式？

采用面向对象的方式进行开发，更符合人类的思维方式。

人类就是以“对象”的方式去认识世界的。

所以面向对象更容易让我们接受。

面向对象就是将显示世界分割成不同的单元，然后没一个单元都实现成对象，然后驱动依稀啊，让各个对象质检协作起来形成一个系统。

对象“张三”

对象“香烟”

对象“打火机”

对象“吸烟的场所”

然后将以上的4个对象组合在一起，就可以模拟一个人的抽烟场景。

其中“张三”对象可以更换为“李四”

其中“香烟”也可以更换品牌

其中“打火机”也可以更换。

都可以更换。

采用面向对象的方式进行开发：

耦合度低，扩展力强。

目前先听一下，需要三四年的时候才能彻底领悟面向对象。

面向过程主要关注的是：实现步骤以及整个过程。

面向对象主要关注的是：对象A，对象B，对象C，然后对象ABC随意组合，或者CBA组合….

## 2、当我们采用面向对象的方式贯穿整个系统的话，涉及到三个术语：

OOA：面向对象分析；

OOD：面向对象设计；

OOP：面向对象编程；

整个软件开发的过程，都是采用OO进行贯穿的。

实现一个软件的过程：

分析（A） 🡪 设计（D） 🡪 编程 （P）

## 3、面向对象包括三大特征：

封装

继承

多态

任何一个面向对象的编程语言都包括这三个特征

例如：

python也有封装，继承，多态

java 也有封装，继承，多态

注意：java只是面向对象编程语言中的一种

除了java之外，还有其他很多很多的编程语言也是面向对象的

## 4、类和对象的概念

面向对象当中最主要“一词”是：对象

什么是类？

类实际上在现实世界当中是不存在的，是一个抽象的概念。

是一个模板，使我们人类大脑进行“思考、总结、抽象”的一个结果。（主要是因为人类的大脑不一般才有了类的概念。）

类本质上是现实世界当中某些事物具有共同特征，将这些共同特征提取出来形成的概念就是一个“类”，“类”就是一个模板。

什么是对象？

对象是实际存在的个体。（真实存在的个体）

在JAVA语言中，要想得到“对象”，必须先定义“类”，“对象”是通过“类”这个模板创造出来的。

类就是一个模板：类中描述的是所有对象的“共同特征信息”

对象就是通过类创建出的个体。

这几个术语需要自己能够阐述出来：

类：不存在的，人类大脑思考总结出的一个模板（这个模板当中描述了共同特征）

对象：实际存在的个体。

实例：对象还有另外一个名字，实例。

实例化：通过类这个模板创建对象的过程，叫做：实例化。

抽象：多个对象具有共同特征，进行思考总结抽取共同特征的过程。

类--【实例化】-🡪 对象（实例）

对象--【抽象】-🡪 类

## 5、思考：“java软件工程师”在开发中起到的一个作用是什么？

我们为什么要做软件开发？

说大一些是为了人民服务，解决现实生活当中的问题。

软件开发既然是为了解决现实世界当中的问题，那么首先java软件必须能够模拟现实世界。

其实软件是一个虚拟的世界。

这个虚拟的世界需要和现实世界一一对应，这才叫模拟。

## 6、类的定义：

6.1、怎么定义一个类，语法格式是什么？

[修饰符列表] class 类名{

//类体 = 属性 + 方法

//属性在代码上以“变量”的形式存在（描述状态）

//方法描述动作/行为

}

注意：修饰符列表可以省略。

6.2、为什么属性是“以”变量的形式存在的？

假设我们要描述一个学生：

学生包括哪些属性：

学号：110

姓名：“张三”

性别：“男”

住址：“深圳宝安区”

答案：是因为属性对应的是“数据”，数据在程序中只能放到变量中。

结论：属性其实就是变量。

变量的分类？

变量根据出现位置进行划分：

方法体当中声明的变量：局部变量。

方法体外声明的变量：成员变量。（这里的成员变量就是属性）

## 7、关于编译的过程

按说应该先编译XueSheng.java 然后再编译XueShengTest.java

但是对于编译器来说，编译XueShengTest.java 文件的时候，会自动找XueSheng.class，如果没有，会

自动编译XueSheng.java 文件，生成XueSheng.class文件。

第一种方式：

javac XueSheng.java

javac XueShengTest.java

第二种方式：

javac XueShengTest.java

第三种方式：

javac \*.java

## 在语法级别上是怎么完成对象的创建呢？

类名 变量名 = new 类名();

这样就完成了对象的创建。

1. 什么是实例变量？

对象又被称为实例。

实例变量实际上就是：对象级别的变量。

Public class 明星类{

Double height

}

身高这个属性所有的明星对象都有，但是每一个对象都有“自己的身高值“

假设创建10个明星对象，height变量应该有10份。

所以这种变量被称为对象级别的变量，属于实例变量。

实例变量在访问的时候，是不是必须先创建对象？

1. 对象和引用的区别？

# DAY14课堂笔记：构造方法

## 画内存图注意事项：、

第一：在内存图上不要体现出代码，内存上应该主要体现“数据”。

第二：画图的时候，图上的图形应该有先后顺序，先画什么，再画什么，必须是有顺序的，而不是想起来什么画什么，程序代码是有执行顺序的，程序执行到哪里就画哪里。

## 为什么要画内存图（非常重要）？

第一：有了内存图，程序不运行，也能知道结果（可以推算出结果）

第二：有了内存图，有助于调试代码

画内存图是对java运行机制的一种理解，不知道运行机制，以后复杂的程序出现错误之后是不会调试的，调 不明白。

1. 程序在什么情况下会出现空指针异常？

空引用 访问“对象相关”的数据时，会出现空指针异常。

垃圾回收器主要针对堆内存。

## 方法在调用的时候参数是如何传递的？

实际上，在java语言中，方法调用时参数传递，和类型无关，都是讲变量中保存的那个“值”传过去，这个“值”可能是一个数字100；也可能是一个java对象的内存地址：0x1234。

## 构造方法

5.1、当一个类中没有提供任何构造方法，系统默认提供一个无参数的构造方法。

这个无参数的构造方法叫做缺省构造器。

5.2、当一个类中手动的提供了构造方法，那么系统将不再默认提供无参数构造方法。

5.3、无参数构造方法，和有参数的构造方法都可以调用。

Student x = new Student();

Student y = new Student(1234);

5.4、构造方法支持方法重载吗？

构造方法是支持方法重载的。

在一个类当中构造方法可以有多个。

并且所有的构造方法名字都是一样的。

方法重载特点：

在同一个类当中，方法名相同，参数列表不同。

5.5、对于实例变量来说，只要你在构造方法中没有手动给它赋值，同意都会默认赋值，默认赋系统值。

6、封装

6.1、面向对象的三大特征：

封装

继承

多态

有了封装，才能继承，有了继承，才能说多态。

6.2、面向对象的首要特征：封装

什么是封装？有什么用？

现实生活中有很多例子都是封装的，例如：

手机，电视机，笔记本电脑，照相机，这些都是外部有一个坚硬的壳儿，封装起来，保护内部的不见，保证内部的部件都是安全的，另外封装了之后，对于我们使用者来说，我们是看不见内部的复杂结构的，我们也不需要关心内部有多么复杂，我们只需要操作外部壳儿上的几个按钮就可以完成操作。

那么封装，有什么用呢？

作用有两个：

第一个：保证内部结构的安全；

第二个：屏蔽复杂，暴露简单。

在代码级别上，封装有什么用？

一个类体当中的数据，假设封装之后，对于代码的调用人员来说，不需要关心代码的复杂实现，只需要通过一个简单的入口就可以访问了，另外，类体中安全级别较高的数据封装起来，外部人员不能随意访问，来保证数据的安全性。

6.3、怎么进行封装，代码怎么实现？

第一步：属性私有化（使用private关键字进行修饰）

第二步：对外提供简单的操作入口。

# DAY15课堂笔记：

## 1、封装的代码实现两步：

第一步：属性私有化 加语法private

第二步：1个属性对外提供set和get方法，外部程序只能通过set方法修改，只能通过get方法读取，可以在set方法中设立关卡来保证数据的安全性。

再强调一下：

set和get方法都是实例方法，不能带static

不带static的方法称为实例方法，实例方法的调用必须先new对象。

set和get方法写的时候有严格的规范要求：（大家要按照规矩来）

set方法长这个样子：

public void set+属性名首字母大写(一个参数){

xxx = 1个参数;

}

get方法长这个样子：

public 返回值类型 get+属性名首字母大写(无参){

return xxx；

}

## 2、static关键字

2.1、static修饰的统一都是静态的，都是类相关的，不需要new对象，直接采用“类名.”访问

2.2、当一个属性是类级别的属性，所有对象的这个属性的值是一样的，建议定义为静态变量。

# DAY16课堂笔记

## this

* 1. this是一个关键字
  2. this可以使用在实例方法中，也可以使用在构造方法中
  3. this出现在实例方法中其实代表的当前对象
  4. this不能使用在静态方法中
  5. this. 大部分情况下可以省略，但是用来区分局部变量和实例变量的时候不能省略
  6. this() 这种语法只能出现在构造方法第一行，表示当前构造方法调用本类其他的构造方法，目的是代码复用。

## 总结所有的变量怎么访问，总结所有的方法怎么访问？

总结一下到目前为止我们都接触过什么了。

## 继承：extends

3.1、什么是继承，有什么用？

继承：在现实世界当中也是存在的，例如：父亲很有钱，儿子不用很努力也很有钱。

继承的作用：

基本作用：子类继承父类，代码可以得到复用（这个不是重要的作用，是基本作用）

主要（重要）作用：因为有了继承关系，才有了后期的方法覆盖和多态机制。

3.2、继承的相关特性

3.2.1、B类继承A类，则A类为超类（superclass）、父类、基类

B类则称为子类（subclass）、派生类、扩展类

class A{}

class B extends A {}

我们平时聊天说的比较多的就是：父类和子类

3.2.2、java中的继承只支持单继承，不支持多继承，c++中支持多继承，这也是java体现简单性的一点， 换句话说，java中不允许这样写代码：

class B extends A,C {}

3.2.3、虽然java中不支持多继承，但有的时候会产生间接继承的效果，例如：

class C ectends B, class B extends A

也就是说，C会直接继承B，其实C还间接继承A。

3.2.4、java中规定，子类继承父类，除构造方法不能继承外，剩下都可以继承。但是私有的属性无法在 子类中直接访问。（父类中private修饰的不能在子类中直接访问，可以通过间接的手段来访问）

3.2.5、java中的类没有显示的继承类，则默认继承Object类，Object类是java语言提供的根类（老祖 宗类），也就是说一个对象与生俱来就有Object类型中所有的特征。

3.2.6、继承也存在一些缺点，例如：CreditAccount 类继承Account类会导致它们之间的耦合度非常高， Account类发生改变之后会马上影响到CreditAccount 类

# DAY17课堂笔记

## 1、继承extends

1.1、测试：子类继承父类之后，能使用子类对象调用父类方法吗？

实际上以上的这个问题问的不对：

\* 本质上，子类继承父类之后，是将父类继承过来的方法归为自己所有

\* 实际上调用的也不是父类的方法，是他子类自己的方法（因为已经继承过来了，就是

\* 属于自己的）

1.2、在实际开发中，满足什么条件的时候，可以使用继承？

凡是采用“is a“ 能描述的，都可以继承。

例如：

cat is a animal : 猫是一个动物

dog is a animal : 狗是一个动物

creditAccout is a Account ：信用卡账户是一个银行账户

……..

假设以后的开发中有一个A类，有一个B类，A类和B类确实也有重复的代码，那么他们两个之间就可以继承吗？不一定，还是要看一看它们之间是否能够使用is a 来描述

class Customer{

String name;//名字

//setter and getter

}

class Product{

String name;//名字

//setter and getter

}

class Product extends Customer{

}

以上的继承就属于很失败的，因为：Product is a Customer，是有违伦理的。

1.3、任何一个类，没有显示继承任何类，默认继承Object，那么Object类当中有哪些方法呢？

以后慢慢的一定要适应看JDK的源代码（多看看牛人写的程序自己才会变成牛人）

先模仿后超越。

java为什么比较好学？

是因为java内置了一套庞大的类库，程序员不需要从0开始写代码，程序员可以基于这套庞大的类库进行“二次“开发（开发速度较快，因为JDK 内置的这套类库实现了很多基础的功能）

例如：String 是SUN编写的字符串类，System是SUN编写的系统类。

这些类都可以哪来直接使用。

JDK源代码在什么位置：

windows：C:\LProgram Files\Java\jdk-13.0.2\lib\src.zip

现在是否能看懂以下代码：

System.out.println(“hello world”)

Sysrtem.out中，out后面没有小括号，说明out是变量名。

另外System是一个类名，直接使用类名System.out，说明out是一个静态变量。

System.out 返回一个对象，然后采用“对象.“ 的方式访问println() 方法

我们研究了一下Object类当中有很多方法，大部分看不懂，其中有一个叫做toString() 的，我们进行了测试，发现：

System.out.println(引用);

档直接输出一个“引用“的时候，println() 方法会先自动调用”引用.toString()” , 然后输出toString()方法的执行结果。

## 2、方法覆盖

2.1、什么时候考虑使用方法覆盖？

父类中的方法无法满足子类的业务需求，自雷有必要对继承过来的方法进行覆盖。

2.2、什么条件满足的时候构成方法覆盖？

第一：有继承关系的两个类

第二：具有与相同方法名、返回值类型、形式参数列表

第三：访问权限不能更低

第四：抛出异常不能更多

2.3、关于Object类中toString() 方法的覆盖？

toString() 方法存在的作用就是：将java对象转换成字符串形式

大多数的java类toString(）方法都是需要覆盖的，因为Object类中提供的toString() 方法输出的是一 个java对象的内存地址。

至于toString() 方法具体怎么进行覆盖？

格式可以自己定义，或者听需求的

2.4、方法重载和方法覆盖有什么区别？

方法重载发生在同一个类当中。

方法覆盖是发生在就有继承关系的父子类之间。

方法重载是一个类中，方法名相同，参数列表不同。

方法覆盖是具有继承关系的父子类，并且重写之后的方法必须和之前的方法一致：方法名一致，参数列表一致，返回值类型一致。

## 3、多态的基础语法

3.1、向上转型和向下转型的概念。

向上转型：子-🡪父 (upcasting)

又被成为自动类型转换：Animal a = new brid();

向下转型：父-🡪子 (downcasting)

又被成为强制类型转换：Cat c = (Cat)a; 需要添加强制类型转换符

什么时候需要向下转型？

需要调用或者执行子类对象中特有的方法。

必须进行向下转型，才可以调用。

向下转型有风险吗？

容易出现ClassCastException (类型转换异常)

怎么避免这个风险？

instanceof 运算符，可以在程序运行阶段动态的判断某个引用指向的对象，是否为某一种类型。

养成好习惯，向下转型之前一定要使用instanceof 运算符进行判断

不管是向上转型还是向下转型，首先他们质检必须要有继承关系，这样编译器就不会报错。

3.2、什么是多态？

多种形态，多种状态，编译和运行有两个不同的状态。

编译器叫做静态绑定。

运行期叫做动态绑定。

Animal a = new Cat();

//编译的时候编译器发现a的类型是Animal，所以编译器会去Animal类中找move() 方法

//找到了，绑定，编译通过。但是运行的时候和底层堆内存当中的对象有关。

//真正执行的时候会自动调用“堆内存中真实对象”的相关方法。

a.move();

多态的典型代码：父类型的引用指向子类型的对象（java中允许这样写代码）

3.3、什么时候必须进行向下转型？

调用子类特有的方法时。

# DAY18课堂笔记：

## 1、多态在开发中有什么作用：

非常重要：五颗星。

多态在开发中的作用是：

降低程序的耦合度，提高程序的扩展力。

面向对象的三大特征：

封装、继承、多态

真的是一环扣一环。

有了封装，有了这种整体的概念之后。

对象和对象之间产生了继承。

有了继承之后，才有了方法的覆盖和多态。

这里涉及到一个软件的开发原则：

\* 软件开发原则有七大原则（不属于java，这个开发原则属于整个软件行业）：

\* 其中有一条最基本的原则：OCP(开闭原则)

目的是：降低程序耦合度，提高程序扩展力。

面向抽象编程，不建议面向具体编程。

## 2、解释之前遗留的问题：

私有方法无法覆盖；

方法覆盖只是针对于“实例方法“，”静态方法覆盖“没有意义（这是因为方法覆盖通常和多态联合使用才有意义）

总结两句话：私有不能覆盖。

静态不谈覆盖。

在方法覆盖中，关于方法的返回值类型。

什么条件满足之后，会构成方法的覆盖呢？

1、发生具有继承关系的两个类之间。

2、父类中的方法和子类重写之后的方法：

具有相同的方法名，相同的形式参数列表、相同的返回值类型。

学习了多态机制之后：

”相同的返回值类型“可以修改一下吗？

对于返回值类型是基本数据类型来书，必须一致；

对于返回值类型是引用数据类型来说，重写之后返回值类型可以变得更小。但是意义不大。实际开发中，没人这样写。

## 3、super关键字

super能出现在实例方法和构造方法中；

\* super的语法是：”super“、”super()“

\* super不能使用在静态方法中；

\* super. 大部分情况下是可以省略的

\* super. 什么时候不能省略？

父类和子类中有同名属性，或者说有同样的方法。

想在子类中访问父类的，super，不能省略。

\*

\* super() 只能出现在构造方法第一行，通过当前的构造方法去调用”父类“中其他的

\* 构造方法，目的是：创建子类对象的时候，先初始化父类型特征。

super的使用：

super.属性名 【访问父类的属性】

super.方法名(实参) 【访问父类的方法】

super.(实参) 【调用父类的构造方法】

# DAY19课堂笔记：

## 1、关于java的继承开发环境：

eclipse、IntelliJ IDEA等。

其中目前主流的继承开发环境是：IntelliJ IDEA

## 2、安装IDEA工具

## 3、IDEA工具的使用：

第一次打开的时候：会弹出一个窗口（Import idea settings）

这个表示导入idea的设置，由于第一次使用并没有任何设置。

所以这里选择：do not import setting… 不导入设置

第二步：接受条款。

第三步：don’t send

第四步：一直下一步（最终选择免费使用30天）

第五步：可能会让你填写email等信息，这里不填写，继续continue

第六步：弹出welcome窗口

点击create new project

注意：在IDEA当中一个project相当于eclipse当中的一个workspace

第七步：新建一个Empty Project

新建一个空的工程，选择创建工程窗口下面最后面衣箱：Empty Project

第八步：给空的工程起一个名字：javase

存储到：C:\Users\Administrator\IdeaProjects\javase

点击finish

第九步：自动弹出一个每日提示，这个每日提示可以取消掉。

第十二步：在new Module 窗口上点击左上角的java，然后next

第十三步：给module起一个名字：chapter15

第十四步：编写代码，在src目录下新建类，写代码，并运行

4、关于IDEA工具的快捷键以及一些简单的设置

4.1、字体设置

file 🡪 settings 🡪 输入font 🡪 设置字体样式以及字号大小

4.2、快速生成main方法

psvm

4.3、快速生成System.out.println()

sout

4.4、注意：IDEA是自动保存，不需要ctrl+s

4.5、删除一行：ctrl + y

4.6、怎么运行：

代码上右键🡪run

或者点击左侧的绿色箭头

或者control + shift + F

4.7、左侧窗口中的列表怎么展开？怎么关闭

左箭头关闭

右箭头展开

上下箭头移动

4.8、idea中退出任何窗口，都可以使用esc键（esc就是退出）

4.9、任何新增/新建/添加的快捷键是：

alt + insert

4.10、窗口变大，变小：

ctrl + shift + F12

4.11、切换java程序：从HelloWorld切换到User

alt + 右箭头

alt + 左箭头

4.12、切换窗口：

alt + 标号

alt + 1 （打开，关闭）

alt + 2

4.13、提示方法的参数，ctrl + p

# DAY20课堂笔记：

## 1、final关键字：

1.1、final修饰的类无法继承。

1.2、final修饰的方法无法覆盖。

1.3、final修饰的变量只能赋一次值。

1.4、final修饰的引用一旦指向某个对象，则不能再重新指向其他对象，但该引用指向的对象内部的数据是可以修改的。

1.5、final修饰的实例变量必须手动初始化，不能采用系统默认值。

1.6、final修饰的实例变量一般和static联合使用，称为常量。

public static final PI = 3.14;

## 2、抽象类和接口以及抽象类和接口的区别。

### 2.1、抽象类

第一：抽象类怎么定义？在class前添加abstract关键字就行了

第二：抽象类是无法实例化的，无法创建对象的，所以抽象类是用来被子类继承的。

第三： final 和 abstract 不能联合使用，这两个关键字是对立的。

第四：抽象类的子类可以是抽象类，也可以是非抽象类。

第五：抽象类虽然无法实例化，但是抽象类可以有构造方法，这个构造方法是供子类使用的。

第六：抽象类中不一定有抽象方法，但是如果有抽象方法，抽象方法必须出现在抽象类中。

第七：抽象方法怎么定义：

public abstract void doSome();

第八（\*\*\*\*\*重点）：一个非抽象的类，继承抽象类，必须将抽象类中的抽象方法进行覆盖/重写/实现。

到目前为止：只是学习了抽象类的基础语法，一个类到底声明为抽象还是非抽象，这个以后慢慢来，写代码多了，自然就理解了

面试题： （判断）java语言中凡是没有方法体的方法都是抽象方法。

不对，错误的。

理由：

Object类中就有很多方法都没有方法体，都是以“;“结尾的，但他们都不是抽象方法，例如：

public native int hashCode();

这个方法底层调用了C++写的动态链接库程序。

前面修饰符列表中没有：abstract 有一个native，表示调用JVM本地程序

### 2.2、接口的基础语法

1、接口也是一种"引用数据类型"。编译之后也是class文件。  
\* 2、接口是完全抽象的。（抽象类是半抽象）或者也可以说接口是特殊的抽象类。  
\* 3、接口怎么定义？语法是什么？  
\* [修饰符列表] interface 接口名{}  
\*  
\* 4、接口支持多继承，一个接口可以继承多个接口。  
\* 5、接口中只包含两部分内容，一部分是：常量；一部分是：抽象方法。接口中没有其他内容了，  
\* 只有两部分内容。  
\* 6、接口中所有的元素都是public修饰的  
\* 7、接口中的抽象方法定义时：public abstract修饰符可以省略；  
\* 8、接口中的方法不能有方法体。  
\* 9、接口中的常量的public static final 可以省略。

# DAY21课堂笔记：

接昨天的学习：

10、一个非抽象的类，实现接口的时候，必须将接口中所有的方法加以实现。

11、一个类可以实现多个接口

12、继承extends 和 实现implements 可以共存，extends在前，implements在后。

13、使用接口，写代码的时候，可以使用多态（父类型引用指向子类型对象）

### 2.3、接口在开发中的作用

注意：接口在开发中的作用，类似于多态在开发中的作用。

多态：面向抽象编程，不要面向具体编程。降低程序的耦合度，提高程序的扩展力。

接口是不是完全抽象的？是

而我们以后正好要求，面向抽象编程。

面向抽象编程这句话以后可以修改为：面向接口编程。

有了接口就有了可插拔，可插拔表示扩展力很强，不是焊接死的

主板和内存条之间有插槽，这个插槽就是接口，内存条坏了，可以重新买一个换下来，这叫做高扩展性（低耦合度）

接口在现实世界中是不是到处都是？

螺栓和螺母质检有接口

灯泡和灯口之间有接口

笔记本电脑和键盘质检有接口（usb接口，usb接口是不是某个计算机协会制定的协议？）

接口有什么用？扩展性好，可插拔

接口是一个抽象的概念。

分析：

中午去饭馆吃饭，这个过程中有接口吗？

接口是抽象的。

菜单是一个接口。（菜单上有一个抽象的照片：西红柿炒鸡蛋）

谁面向接口调用，（顾客面向菜单点菜，调用接口）

谁负责实现这个接口（后台的厨师负责把西红柿鸡蛋做好，是接口的实现者）

这个接口有什么用呢？

这个饭馆的菜单让“顾客“和“后厨“解耦了

顾客不用找后厨，后厨不用找顾客，他们之间完全依靠这个抽象的菜单沟通。

总结一句话：

面向接口编程，可以降低程序的耦合度，提高程序的扩展力，符合OCP开发原则。

接口的使用离不开多态机制。（接口+多态才可以达到降低耦合度）

接口可以解耦合，解开的是谁和谁的耦合？

任何一个接口都有调用者和实现者。

接口可以讲调用者和实现者解耦合。

调用者面向接口调用。

实现者面向接口编写实现。

以后进行大项目的开发，一般都是将项目分离成一个模块一个模块的，模块与模块之间采用接口衔接，降低耦合度。

### 2.4、 类型和类型之间的关系：

is a

has a

like a

is a :

Cat is a Animal(猫是一个东旭)

凡是能满足is a 的标识“继承关系“

has a：

I have a Pen（我有一支笔）

凡是能满足has a 关系的表示“关联关系”

like a ：

cooker like a Foodmenu（厨师像一个菜单一样）

凡是能满足like a 关系的标识“实现关系”

实现关系通常是：类实现接口

### 2.5、抽象类和接口有什么区别？

在这里只说一下抽象类和接口在语法上的区别。

之余以后抽象类和接口应该怎么进行选择，通过后面的项目去体会/学习

抽象类是半抽象的；

接口是完全抽象的；

抽象类中有构造方法；

接口中没有构造方法；

接口接口之间支持多继承；

类和类之间只能单继承；

一个类可以同时实现多个接口；

一个抽象类只能继承一个类（单继承）

这里先透露一个信息：

以后接口使用的比抽象类多，一般抽象类用的还是比较少。

接口一般都是对“行为”的抽象。

## 3、package和import（不用IDEA工具的时候）

3.1、package

第一：出现在java源文件第一行；

第二：带有包名怎么编译？

javac -d . xxx.java

第三：怎么运行

java 完整类名

补充：以后说类名的时候，如果带包名描述，表示完整类名。

如果没有带包名，描述的话，表示简类名。

java.util.Scanner 完整类名

Scanner 简类名

3.2、import

import什么时候不需要？

java.lang 不需要

同包下不需要

其他一律需要；

怎么用？

import 完整类名；

import 包名.\*；

# DAY22课堂笔记：

1、访问控制权限

1.1、访问控制权限都有哪些？

4个

private 私有

protected 受保护

public 公开

默认（啥也不写）

1.2、以上的4个访问控制权限：控制的范围是什么？

private 表示私有的，只能再本类中访问

public 表示公开的，在任何位置都可以访问

“默认”表示只能再本来，以及同包下访问

protected表示只能在本类、同包、子类中访问

访问控制修饰符 本类 同包 子类 任意位置

-----------------------------------------------------------------------------------

public 可以 可以 可以 可以

protected 可以 可以 可以 不行

默认 可以 可以 不行 不行

private 可以 不行 不行 不行

范围从大到小排序：

public > protected > 默认 > private

1.3、访问控制权限修饰符可以修饰什么？

属性（public > protected > 默认 > private 4个都能用）

方法（public > protected > 默认 > private 4个都能用）

类（public 和 默认能用，其他不能用）

接口（public 和 默认能用，其他不能用）

2、JDK类库的根类：Object

2.1、这个老祖宗类中的方法我们需要先研究一下，因为这些方法都是所有子类通用的。

任何一个类默认继承Object。就算没有直接继承，最终也会间接继承。

2.2、Object类当中有哪些常用的方法？

我们去哪里找这些方法呢？

第一种方法：去源代码当中（但是这种方式比较麻烦，源代码也比较难）

第二种方法：去查阅java的类库的帮助文档。

什么是API？

应用程序编程接口

整个JDK的类库就死一个javase的API

每一个API都会配置一套API帮助文档。

SUN公司提前写好的这套类库就是API（一般每一份API都对应一份API帮助文档）

目前为止我们只需要知道这几个方法即可：

protected Object clone() //负责对象克隆的

int hashCode() //获取对象哈希值的一个方法

Boolean equals (Object obj) //判断两个对象是否相等

String toString() //将对象转换成字符串形式

protected void finalize() //垃圾回收器负责调用的方法

2.3、toString()方法

以后所有类的toString方法是需要重写的。

重写规则，越简单越明了越好。

System.out.println(引用); 这里会自动调用“引用“的toString方法；

String类是SUN写的，toString方法已经重写了

2.4、equals方法

以后所有类的equals方法也需要重写，因为Object中的equals方法比较的是两个对象的内存地址，我们应该比较的是内容，所以需要重写。

重写规则：自己定，主要看是什么和什么相等时表示两个对象相等。

基本数据类型比较用：==

对象和对象比较：调用equals方法

String类是SUN编写的，所以String类的equals方法重写了。

以后判断两个字符串是否相等，最好不要使用== ，要调用字符串对象的equals方法。

注意：重写equals方法要彻底。

2.5、finalize()方法

这个方法是protected修饰的，在

# DAY23课堂笔记：

## 1、数组

1.1、数组的优点和缺点，并且要理解为什么？

第一：空间存储上，内存地址是连续的；

第二：每个元素占用的空间大小相同；

第三：知道首元素的内存地址；

第四：通过下标可以计算出偏移量；

通过一个数学表达式，就可以快速计算出某个下标位置上元素的内存地址；

直接通过内存地址定位，效率非常高；

优点：检索效率高；

缺点：随机增删效率较低，数组无法存储大量数据；

注意：数组最后一个元素的增删效率不受影响。

1.2、一维数组的静态初始化和动态初始化

静态初始化：

int[] arr = {1,2,3,4};

Object obj = {new Object(), new Object(), new Object()};

动态初始化：

int[] arr = new int[4]; //4个长度，每个元素默认值0

Object obj = new Object[4]; //4个长度，每个元素默认值null

1.3、一维数组的遍历

for(int i = 0 ; i < arr.length; i++){

System.out.println( arr[i] );

}

1.4、二维数组的静态初始化和动态初始化

静态初始化：

int[][] arr= {

{1,3,2,4},

{2324,54,34,23},

{234,54,32,6,5}

};

Object[][] obj = {

{new Object(), new Object(), new Object()},

{ new Object(), new Object(), new Object(),new Object()},

{ new Object(), new Object(), new Object() }

};

动态初始化：

int[][] arr = new arr[3][4];

Object[][] arr = new Object[4][5];

Animal类型数组，里面可以存储Animal类型对象，以及Animal类型的子类型都可以

Animal[][] animls = new Animal[5][6];

…….

1.5、二维数组的遍历

for(int i = 0; i < arr.length; i ++){

for(int j = 0; j < arr[i].length; j ++){

System.out.println(arr[i][j]);

}

换行

System.out.println();

}

1.6、main方法上“String[] arga“参数的使用（非重点，了解一下，以后一般都是有界面的，可视化）

1.7、数组的拷贝：System.arraycopy() 方法的使用

数组有一个特点：长度一旦确定，不可变

所以数组长度不够的时候，需要扩容，扩容的机制是：新建一个大数组

将小数组中的数据拷贝到大数组，然后小数组对象被垃圾回收；

1.8、对数组中存储引用数据类型的情况，要会画它的内存结构图

# DAY24课堂笔记：

## 1、解决昨天的两个作业题；

1.1、数组模拟栈：弹栈，压栈

1.2、酒店管理系统的模拟

## 2、数组

2.1、常见的算法：

排序算法：

冒泡排序算法；

选择排序算法；

查找算法：

二分法查找；

以上算法在以后的java实际开发中我们不需要使用的。

因为java已经封装好了，直接调用就行。

只不过以后面试的时候，可能有机会碰上。

2.2、算法实际上在java中不需要精通，因为java中已经封装好了，

要排序就调用方法就行。例如：java中提供了一个数组工具类：

java.util.Arrays

Arrays是一个工具类

其中有一个sort（） 方法，可以排序。静态方法，直接使用类名调用就行。

## 3、冒泡排序：

参与比较的数据：9 8 10 7 6 0 11

第一次循环：

8 9 10 7 6 0 11（第一次比较：交换）

8 9 10 7 6 0 11 （第二次比较：不交换）

8 9 7 10 6 0 11（第三次比较：交换）

8 9 7 6 10 0 11（第四次比较：交换）

8 9 7 6 0 10 11（第五次比较：交换）

8 9 7 6 0 10 11（第六次比较：不交换）

参与比较的数据：8 9 7 6 0 10

第二次循环：

8 9 7 6 0 10（第一次比较：不交换）

8 7 9 6 0 10（第二次比较：交换）

8 7 6 9 0 10（第三次比较：交换）

8 7 6 0 9 10（第四次比较：交换）

8 7 6 0 9 10（第五次比较：不交换）

参与比较的数据：8 7 6 0 9

第三次循环：

7 8 6 0 9（第一次比较：交换）

7 6 8 0 9（第二次比较：交换）

7 6 0 8 9（第三次比较：交换）

7 6 0 8 9（第四次比较： 不交换）

参与比较的数据：7 6 0 8

第四次循环:

6 7 0 8（第一次比较：交换）

6 0 7 8（第二次比较：交换）

6 0 7 8（第三次比较：不交换）

参与比较的数据：6 0 7

第五次循环:

0 6 7（第一次比较：交换）

0 6 7（第二次比较：不交换）

参与比较的数据：0 6

第六次循环：

0 6（一次比较：不交换）

## 4、选择排序；

## 5、二分法查找（折半查找）

第一：二分法查找简历再排序的基础之上；

第二：二分法查找效率要高于“一个挨着一个“的这种查找方式；

第三：二分法查找原理：

10（0下标） 23 56 89 100 111 222 235 500 600（下标9） arr数组

目标：找出600的下标：

（0+9）/2 🡪 4（中间元素的下标）

arr[4] 这个元素就是中间元素：arr[4] 是100；

100<600

说明被查找的元素在100的右边

那么此时开始下标变成：4+1

（5+9）/2 🡪 7 （中间元素的下标）

arr[7] 对应的是：235

235 < 600

说明被查找的元素在235的右边

开始下标又进行了改变：7+ 1

（8+9）/2 = 8

arr[8] –》 500

500《600

开始元素的下标又发生了变化：8+ 1==9

（9+9） /2 = 9

arr[9] 是600，正好和600相等，此时找到了

# DAY25课堂笔记

## 1、String类

### 1.1、对String在内存存储方面的理解：

第一：字符串一旦创建不可变；

第二：双引号括起来的字符串存储在字符串常量池中；

第三：字符串的比较必须使用equals方法；

第四：String已经重写了toString() 和 equals()方法；

### 1.2、String的构造方法

String s = “”;

String s = new String(“”);

String s = new String(byte数组);

String s = new String(byte数组，起始下标，长度)；

String s = new String(char数组)；

String s = new String(char数组，起始下标，长度)；

### 1.3、String类常用的21个方法

## 2、StringBuffer/StringBuiler

2.1、StringBuffer/StringBuiler可以看做可变长度字符串；

2.2、StringBuffer/StringBuiler初始化容量16；

2.3、StringBuffer/StringBuiler是完成字符串拼接操作的，方法名：qppend

2.4、StringBuffer是线程安全的，StringBuiler是非线程安全的

2.5、频繁进行字符串拼接不建议使用“+“

## 3、八种基本数据类型对应的包装类：

3.1、包装类存在有什么用：

方便编程

3.2、八种包装类的类名是什么：

Byte

Short

Integer

Long

Float

Double

Boolean

Charcter

3.3、所有数字的父类Number

3.4、照葫芦画瓢：学习Integer，其他的模仿Integer

3.5、什么是装箱，什么是拆箱？

# DAY26课堂笔记：

## 1、八种基本数据类型对应的包装类。

1.1、什么是自动装箱和自动拆箱，代码怎么写？

1.2、Integer类常用方法

1.3、Integer String int 三种类型互相转换

## 2、日期类：

2.1、获取系统当前时间：

Date d = new Date();

2.2、日期格式化：Date 🡪 String

yyyy—MM—dd HH:mm:ss SSS

SimpleDateFormat sdf = new SimpleDateFormat(“yyyy—MM—dd HH:mm:ss”);

String s = sdf.format(new Date());

2.3、String 🡪 Date

SimpleDateFormat sdf = new SimpleDateFormat(“yyyy—MM—dd HH:mm:ss”);

Date d = sdf.parse(“2008-08-08 08:08:08”);

2.4、获取毫秒数：

long begin = System.currentTimeMillis();

## 3、数字类：

3.1、DecimalFormat数字格式化

###,###.## 表示加入千分位，保留两位小数

###,###.0000 表示加入千分位，保留四位小数，不够补0

3.2、BigDecimal

财务软件中通常使用BigDecimal

## 4、随机数

4.1、怎么产生int类型的随机数:

Random r = new Random();

int i = r.nextInt();

4.2、怎么产生某个范围之内的int类型随机数。

Random r = new Random();

int i = r.nextInt(101); //产生[0-100]的随机数

## 5、枚举

5.1、枚举是一种引用数据类型

5.2、枚举编译之后也是class文件

5.3、枚举类型怎么定义：

enum 枚举类型名{

枚举值1，枚举值2，枚举值3

}

5.4、当一个方法执行结果超过两种情况，并且是一枚一枚可以列举出来

的时候，建议返回值类型设计为枚举类型

## 6、异常处理机制

6.1、java中异常的作用是：增强程序健壮性

6.2、java中异常以类和对象的形式存在

# DAY27课堂笔记：

## 1、java的异常处理机制：

1.1、异常在java中以类和对象的形式存在，那么异常的继承结构是怎样的？

我们可以使用UML图来描述一下继承结构；

画UML图有很多工具，例如：Rational Rose (收费的) 、starUML等…..

Object

Object下有Throwable（可抛出的）

Throwable下有两个分支：Error（不可处理，直接退出JVM）和Exception（可处理的）

Exception下有两个分支：

Exception的直接子类：编译时异常（要求程序员在编写程序阶段必须预先对这些异常进行处理，如果不处理，编译器报错，因为得名编译器异常）

RuntimeException：运行时异常（在编写程序阶段程序员可以预先处理，也可以不管，都行）

1.2、编译时异常和运行时异常，都是发生在运行阶段，编译阶段异常是不会发生的。

编译时异常因为什么而得名？

因为编译时异常必须在编译（编写）阶段预先处理，如果不处理编译器报错，因此得名。

所有异常都是在运行阶段发生的。因为只有程序运行阶段才可以new对象。

因为异常的发生就是new异常对象。

1.3、编译时异常和运行时异常的区别？

编译时异常一般发生的概率比较高。

举个例子：

你看到外面下雨了，倾盆大雨的。

你出门之前会预料到：如果不打伞，我可能会生病（生病是一种异常）

而且这个异常发生的概率很高，所以我们出门之前要拿一把伞。

”拿一把伞“就是对”生病异常“发生之前的一种处理方式

对于一些发生概率比较高的异常，需要在运行之前对其进行预处理。

运行时异常一般发生的概率比较低。

举个例子：

小明走在大街上，可能会被天上的飞机轮子砸到；

被飞机轮子砸到也算是一种异常。

但是这种异常发生概率较低。

在出门之前你没必要提前对这种发生概率较低的异常进行预处理。

如果你预处理这种异常，你将活得很累。

假设java中没有对异常进行划分，没有分为：编译时异常和运行时异常。

所有的异常都需要在编写程序阶段对其进行预处理，将是怎样的效果呢？

首先，如果这样的话，程序肯定是绝对的安全的。

但是程序员编写程序太累，代码到处都是处理异常的代码。

1.4、编译时异常还有其他名字：

受检异常：CheckedException

受控异常

1.5、运行时异常还有其他名字：

未受检异常：UnCheckedException

非受控异常

1.6、再次强调：所有异常都是发生在运行阶段的。

1.7、java语言中对异常的处理包括两种方式:

第一种方式：在方法声明的位置上，使用throws关键字，抛给上一级。

谁调用我，我就抛给谁，抛给上一级；

第二种方式：使用try…catch 语句进行异常的捕捉

这件事发生了，谁也不知道，因为我给抓住了；

举个例子：

我是某集团的一个销售员，因为自己的失误，导致公司损失了1000元，

”损失1000元“可以看做是一个异常发生了，我有两种处理方式：

第一种方式：我把这件事告诉我的领导（异常上抛）

第二种方式：我自己掏腰包把这个钱补上（异常的捕捉）

思考：

异常发生之后，如果我选择了上抛，抛给了我的调用者，调用者需要对这个异常继续处理，那么调用者处理这个异常同样有两种处理方式。

1.8、注意：java中异常发生之后如果一直上抛，最终抛给了main方法，main方法继续向上抛，抛给了JVM，JVM知道这个异常发生，只有一个结果，终止JVM程序的执行。

## 2、什么是UML？有什么用？

UML是一种统一建模语言

一种图标式语言（画图的）

UML不是只有java中使用，只要是面向对象的编程语言，都有 UML

一般画UML图的都是软件架构师或者说是系统分析师，这些级别的人员使用的；

软件设计人员使用UML

在UML图中可以描述类和类之间的关系，程序执行的流程，对象的状态等。

盖大楼和软件开发一样，一个道理；

盖楼之前，会先由建筑师画图纸，图纸上一个一个符号都是标准符号；

这个图纸画完，只要是搞建筑的都能看懂，因为这个图纸上标注的这些符号都是一种“标准的语言“

在jaca软件开发中国，软件分析师/设计师负责设计类，java软件开发人员必须要能看懂。

# DAY28课堂笔记

1、集合概述

1.1、什么是集合？有什么用？

数组其实就是一个集合。集合实际上就是一个容器，可以来容纳其他类型的数据。

集合为什么说在开发中使用较多？

集合是一个容器，是一个载体，可以一次容纳多个对象。

在实际开发中，假设链接数据库，数据库当中有10条数据。

那么假设吧这10条数据查询出来，在java程序中会将10条数据封装成10个java对象，然后

将10个java对象放到某一个集合当中，将集合传到前端，然后遍历集合，将一个数据一个数据展现出来。

1.2、集合不能直接存储基本数据类型，另外集合也不能直接存储java对象，集合当中存储的都是java对象的内存地址。（或者说集合中存储的都是引用。）

1.3、在java中每一个不同的集合，底层会对应不同的数据结构。往不同的集合中存储元素，等于将数据放到了不同的数据结构当中。什么是数据结构？数据存储的结构就是数据结构。不同的数据结构，数据存储方式不同。例如：

数组，二叉树，链表，哈希表…..

以上这些都是常见的数据结构。

往集合c1中放数据，可能是放到数组上了。

往集合c2中放数据，可能是放到二叉树上了。

使用不同的集合等同于使用了不同的数据结构。

在java集合这一章节，你需要掌握的不是精通数据结构。java已经将数据结构实现了，已经写好了这些常用的集合类，你只需要掌握怎么使用？在什么情况下选择哪一种合适的集合去使用即可。

new ArrayList(); 创建一个集合，底层是数组。

new LinkedList(); 创建一个集合对象，底层是链表。

new TreeSet(); 创建一个集合对象，底层是二叉树。

1.4、集合在java JDK中哪个包下？

java.util.\*;

所有的集合类和集合接口都在java.util包下。

1.5、为了能让大家掌握集合这块的内容，最好能将集合的继承结构图背会！！！

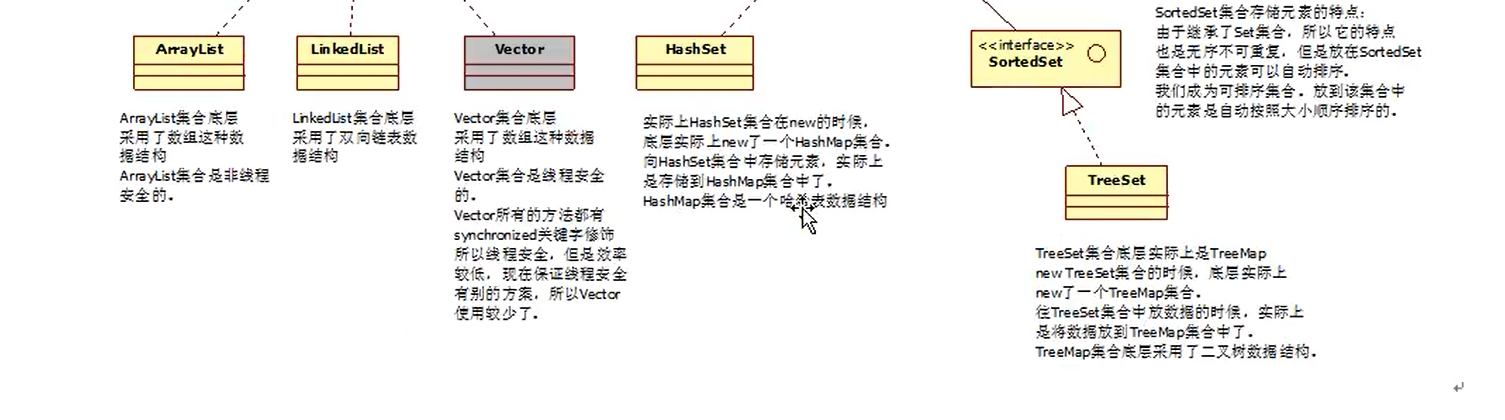
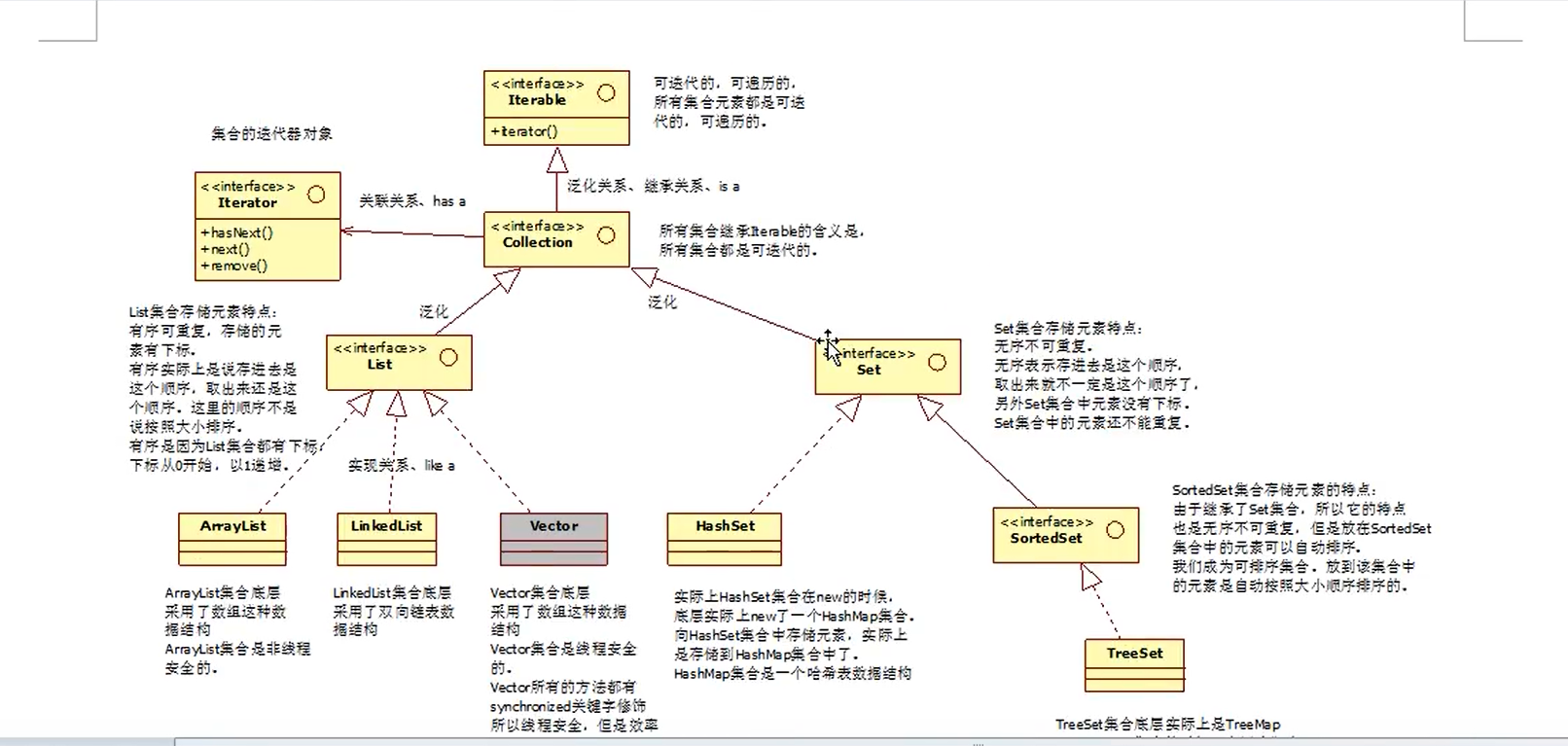
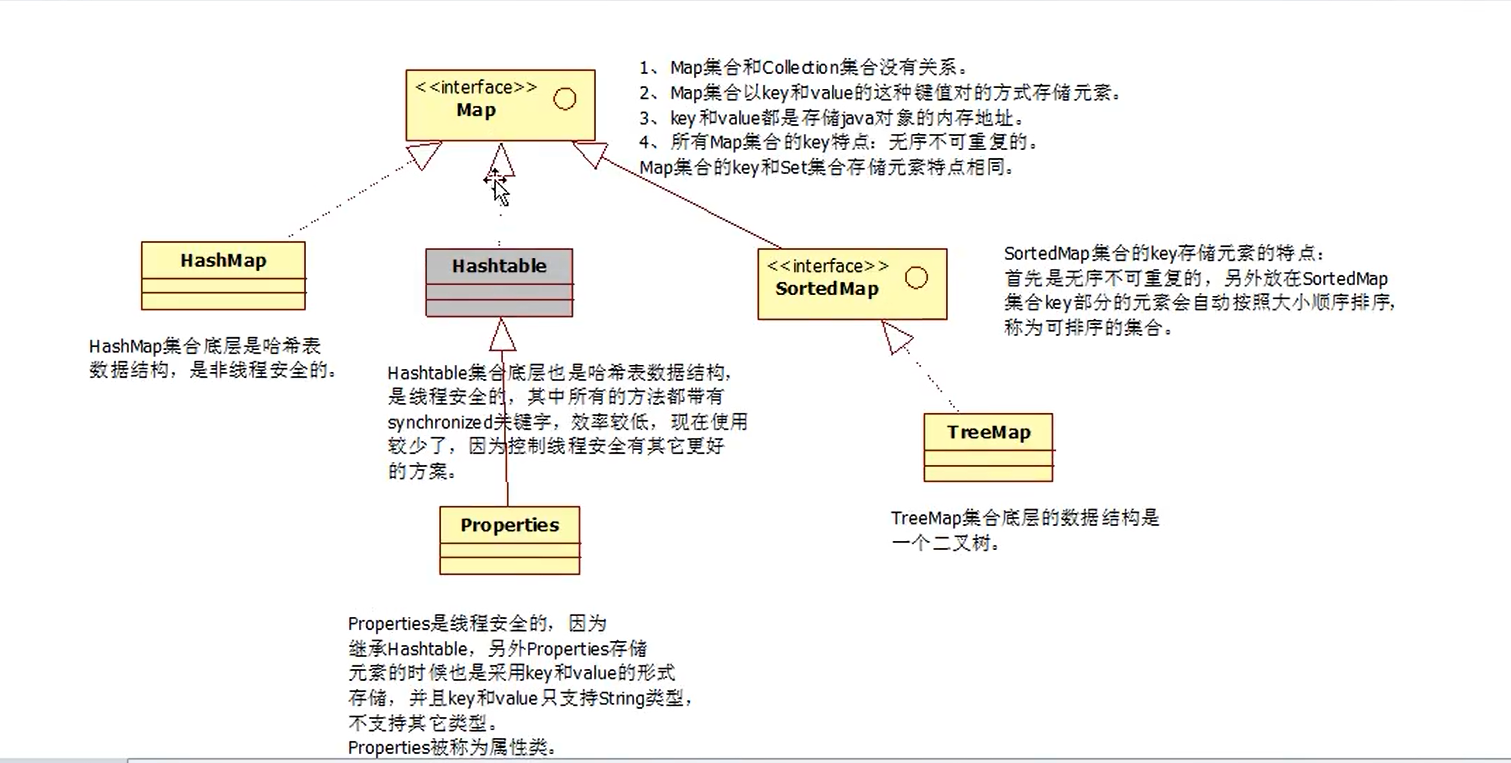
1.6、在java中集合分为两大类：

一类是单个方式存储元素：

单个方式存储元素，这一类集合中超级父接口： java.util.Collection;

一类是以键值对的方式存储元素：

以键值对的方式存储元素，这一类集合中超级父接口：java.util.Map;



总结：（所有的实现类）：

ArrayList：底层是数组。

LinkedList：底层是双向链表。

Vector：底层是数组，线程安全的，效率较低，使用较少。

HashSet：底层是HashMap，放到HashSet集合中的元素等同于放到HashMap集合Key部分了。

TreeSet：底层是TreeMap，放到TreeSet集合中的元素等同于放到TreeMap集合Key部分了。

HashMap：底层是哈希表。

Hashtable：底层也是哈希表，只不过线程安全的，效率较低，使用较少。

Properties：是线程安全的，并且Key和Value只能存储字符串String。

Treemap：底层是二叉树，TreeMap集合的Key可以自动按照大小顺序排序。

List集合存储元素的特点：

有序可重复；

有序：存进去的顺序和取出来的顺序是一致的，每一个元素都有下标；

可重复：存进去1，可以再存储一个1；

Set(Map)集合存储元素的特点：

无序不可重复；

无序：存进去的顺序和取出来的顺序不一定相同，另外set集合中的元素没有下标；

不可重复：存进去1，不能再存进去1了；

SortedSet（SortedMap）集合存储元素的特点：

首先是无序不可重复的，但是SortedSet集合中的元素是可排序的。

无序：存进去的顺序和取出来的顺序不一定相同，另外set集合中的元素没有下标；

不可重复：存进去1，不能再存进去1了；

可排序：可以按照大小顺序排列。