# 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