|  |  |
| --- | --- |
| 弯道和树的图片  Learning Java  Java基础篇 | 摘要  Java 20天学习  作者  Forest |

**目录**

[Day 1 2017/3/23、2017/3/29 7](#_Toc480398113)

[计算机存储单元 7](#_Toc480398114)

[注意事项： 7](#_Toc480398115)

[Editplus设置 7](#_Toc480398116)

[常用的dos命令 8](#_Toc480398117)

[tips： 8](#_Toc480398118)

[1. complier/interpreter 8](#_Toc480398119)

[2.报错类型之非法字符 8](#_Toc480398120)

[3.配置环境变量的原因 8](#_Toc480398121)

[Day 2 2017/3/24、2017/3/29 10](#_Toc480398122)

[注释(解读和排错) 10](#_Toc480398123)

[关键字 10](#_Toc480398124)

[标识符 10](#_Toc480398125)

[常量 11](#_Toc480398126)

[变量 12](#_Toc480398127)

[数据类型(本质是关键字) 12](#_Toc480398128)

[基本数据类型 12](#_Toc480398129)

[引用数据类型 13](#_Toc480398130)

[数据类型的转换 14](#_Toc480398131)

[隐式转换 14](#_Toc480398132)

[强制转换 14](#_Toc480398133)

[字符与字符串参与运算（ASCII码表） 14](#_Toc480398134)

[运算符 14](#_Toc480398135)

[Day 3 2017/3/25 16](#_Toc480398136)

[自习 16](#_Toc480398137)

[课堂练习——冒泡排序法 16](#_Toc480398138)

[Day 4 2017/3/26 17](#_Toc480398139)

[逻辑运算符 17](#_Toc480398140)

[三元（三目）运算符 17](#_Toc480398141)

[三元运算符和if条件语句的区别 17](#_Toc480398142)

[键盘录入（Java语言的独特之处） 17](#_Toc480398143)

[流程控制语句 18](#_Toc480398144)

[顺序结构语句 18](#_Toc480398145)

[选择结构语句 18](#_Toc480398146)

[if 18](#_Toc480398147)

[switch 20](#_Toc480398148)

[Special points： 20](#_Toc480398149)

[Day 5 2017/3/27 25](#_Toc480398150)

[要记住的tips 25](#_Toc480398151)

[流程控制语句 25](#_Toc480398152)

[循环结构 25](#_Toc480398153)

[for（要求精通） 25](#_Toc480398154)

[while循环 26](#_Toc480398155)

[if和while的死循环 26](#_Toc480398156)

[控制跳转语句 27](#_Toc480398157)

[break 27](#_Toc480398158)

[continue 27](#_Toc480398159)

[return 27](#_Toc480398160)

[函数（方法） 27](#_Toc480398161)

[方法概述 27](#_Toc480398162)

[返回值为空（面试题） 28](#_Toc480398163)

[main方法 28](#_Toc480398164)

[如何写方法 29](#_Toc480398165)

[注意事项 29](#_Toc480398166)

[方法的重载overload 29](#_Toc480398167)

[方法的数据参数传递问题（面试题） 30](#_Toc480398168)

[课上闪闪发光的练习题们： 30](#_Toc480398169)

[1. 30](#_Toc480398170)

[2. 31](#_Toc480398171)

[3. 34](#_Toc480398172)

[4 循环语句{前加了； 35](#_Toc480398173)

[5 while ( !e ) 36](#_Toc480398174)

[6.一行输出5个数后，换行 36](#_Toc480398175)

[(基础班)自习 2017/3/30 38](#_Toc480398176)

[循环语句经典思想——统计思想 38](#_Toc480398177)

[第三种循环语句——do while 38](#_Toc480398178)

[三种循环语句的区别 38](#_Toc480398179)

[代码优化 38](#_Toc480398180)

[控制跳转语句的标号 39](#_Toc480398181)

[方法之输出星形及其调用 39](#_Toc480398182)

[数组 39](#_Toc480398183)

[存在的原因： 39](#_Toc480398184)

[数组概念 39](#_Toc480398185)

[格式 39](#_Toc480398186)

[数组初始化 40](#_Toc480398187)

[动态初始化 40](#_Toc480398188)

[静态初始化 40](#_Toc480398189)

[越界和空指针（重要） 40](#_Toc480398190)

[数组的遍历 40](#_Toc480398191)

[数组获取最值(获取数组中的最大值最小值) 41](#_Toc480398192)

[数组元素反转(就是把元素对调) 41](#_Toc480398193)

[Day 3 2017/4/1 42](#_Toc480398194)

[课上闪闪发光的练习题们： 42](#_Toc480398195)

[1.if 42](#_Toc480398196)

[2. 43](#_Toc480398197)

[Day 4 2017/4/3 45](#_Toc480398198)

[随机数 45](#_Toc480398199)

[Day5 2017/4/5 46](#_Toc480398200)

[注意： 46](#_Toc480398201)

[課上闪闪发光的练习题们： 46](#_Toc480398202)

[1.打印水仙花数，同时数出水仙花数的个数 46](#_Toc480398203)

[Day 6 2017/4/6 47](#_Toc480398204)

[Eclipse断点调试 47](#_Toc480398205)

[作用： 47](#_Toc480398206)

[常用操作： 47](#_Toc480398207)

[注意 47](#_Toc480398208)

[综合练习 47](#_Toc480398209)

[Day 7 2017/4/8 49](#_Toc480398210)

[面向对象 49](#_Toc480398211)

[思想概述 49](#_Toc480398212)

[类与对象 49](#_Toc480398213)

[类 49](#_Toc480398214)

[属性 49](#_Toc480398215)

[行为 49](#_Toc480398216)

[对象 49](#_Toc480398217)

[类的使用 49](#_Toc480398218)

[注意事项： 50](#_Toc480398219)

[1.基本类与测试类 50](#_Toc480398220)

[2.成员变量和局部变量的区别（面试题） 50](#_Toc480398221)

[3.成员变量的其他注意点 50](#_Toc480398222)

[4.成员变量放在方法后面也是可以的 50](#_Toc480398223)

[5.面试题 50](#_Toc480398224)

[6.面试题 51](#_Toc480398225)

[Private关键字 51](#_Toc480398226)

[特点 51](#_Toc480398227)

[存在的意义 51](#_Toc480398228)

[set/get方法 51](#_Toc480398229)

[基本类的标准代码: 52](#_Toc480398230)

[封装的概述和好处: 52](#_Toc480398231)

[this关键字 52](#_Toc480398232)

[构造方法 53](#_Toc480398233)

[作用 53](#_Toc480398234)

[格式 53](#_Toc480398235)

[构造方法注意事项与重载 53](#_Toc480398236)

[类名作为方法的形式参数 53](#_Toc480398237)

[案例代码十四: 53](#_Toc480398238)

[类名作为返回值案例 54](#_Toc480398239)

[案例代码十五: 54](#_Toc480398240)

[Day 8 2017/4/9 55](#_Toc480398241)

[String 55](#_Toc480398242)

[构造方法（3种） 55](#_Toc480398243)

[判断功能（4种） 55](#_Toc480398244)

[获取功能（5种） 56](#_Toc480398245)

[其他功能（2种） 57](#_Toc480398246)

[StringBuilder 57](#_Toc480398247)

[构造方法 57](#_Toc480398248)

[成员方法 57](#_Toc480398249)

[String和StringBuilder的互相转换 58](#_Toc480398250)

[Day 9 2017/4/10 59](#_Toc480398251)

[集合概述 59](#_Toc480398252)

[集合的功能 59](#_Toc480398253)

[增加元素 59](#_Toc480398254)

[删改查元素 60](#_Toc480398255)

[学生管理系统案例 60](#_Toc480398256)

[day 10 2017/4/13 64](#_Toc480398257)

[IO流 64](#_Toc480398258)

[就业班 66](#_Toc480398259)

[Day 1 2017/4/19 66](#_Toc480398260)

[Static 66](#_Toc480398261)

[概述 66](#_Toc480398262)

[特点 66](#_Toc480398263)

[注意事项 66](#_Toc480398264)

[静态的优缺点 66](#_Toc480398265)

[静态应用 67](#_Toc480398266)

[Math类使用 67](#_Toc480398267)

[类变量与实例变量辨析（官方版） 68](#_Toc480398268)

[静态变量和成员变量的区别（小虎版） 68](#_Toc480398269)

[代码块 69](#_Toc480398270)

[局部代码块 69](#_Toc480398271)

[构造代码块 69](#_Toc480398272)

[静态代码块 69](#_Toc480398273)

[同步代码块 69](#_Toc480398274)

# Day 1 2017/3/23、2017/3/29

## 计算机存储单元

计算机存储设备的最小信息单元为bit（比特位）

最小的存储单元为byte，由连续的8个比特位组成

## 注意事项：

文件后缀名-查看

编译：全名称，需要后缀名

运行：不需要

class类名：首字母要大写。如：

class Hello

符号要用英文

在文件所在文件夹按住shift+右键-在此处打开命令窗口

括号成对出现

左大括号要有空格，遇到左大括号要有tab

方法与程序块间要空行；运算符两侧加空格

## Editplus设置

工具-

捕获控制台输出

单行注释可以嵌套，多行注释不能嵌套

## 常用的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
  + cd (change directory)改变指定目录(进入指定目录)
    - 进入 cd 目录；cd 多级目录
    - 回退 cd.. ；cd\
  + cls : (clear screen)清屏
  + exit : 退出dos命令行

## tips：

#### 1. complier/interpreter

javac 编译器 complier

java 解释器 interpreter

#### 2.报错类型之非法字符

通常是原文件中含有中文

#### 3.配置环境变量的原因

任何目录下都能运行java文件

# Day 2 2017/3/24、2017/3/29

## 注释(解读和排错)

单行注释 //（可嵌套）

多行注释 /\*\*/(不可嵌套)

文档注释

## 关键字

定义：被java赋予特殊含义的单词（全部小写）

注意事项：“goto”“const”是关键字，但是在java中暂时没有被赋予含义

## 标识符

定义：给类、接口、方法、变量（包、常量）起名字时使用的字符序列。

注：类是java程序中最基本的组成单位。

规则：$\_；数字0-9；英文大小写字母

组成规则：

1.unicode字符：数字字符、英文大小写字母、汉字（不建议使用汉字）

2.下划线

3.美元符

注意事项：1.不能与关键字重合；

2.不能以数字开头

3.区分大小写

识别方法：1.是否是关键字；

2.是否符合命名规则。

命名的一般规则：

* 包：

全小写；一般为公司域名倒着写；

eg：www.heima.com>com.heima.(用途)

* 类/接口

1个单词/多个单词：首字母大写，单词之间没有任何符号（驼峰命名）

* 方法/变量

1个单词：全部小写

多个单词：从第二个单词开始首字母大写

* 常量

1个单词：全部大写

多个单词：全部大写，单词间用\_相隔

## 常量

定义：在程序执行过程中不可发生改变的量

类型：

* 字符串常量：用双引号括起来
* 字符常量：用单引号括起来，只能放单个字符
* 整数常量：所有整数
* 小数常量：所有小数
* 布尔常量：只有true、false（两者为关键字）
* 空常量：null（讲数组时讲）

## 变量

定义：变量是内存中的一小块区域；在程序执行过程中，于一定范围内能发生改变的量（Q: 变量具体指哪一部分？变量名+变量值吗？还是包括数据类型的总体？）

组成：

1.对区域有一定限定：用数据类型限定；

2.给区域一个名称：变量名；

3.区域内要有内容：变量值。

格式：数据类型 变量名 = 变量值

注意事项：

1. 作用域：同一个区域不可使用相同变量名
2. 初始化值：使用前必须赋值（被命名的变量暂时不用参与运算，不赋值也不会报错）
3. 可以多次赋值

## 数据类型(本质是关键字)

Java是强类型语言，针对每种数据都给出了明确的数据类型

引用数据类型（类、接口、数组）（面向对象时再讲）

### 基本数据类型

* 整数型：

byte: 1个字节（8个开关） -128-127>> -2^7至2^7-1

short：2个字节

int：4个字节

long：8个字节

注意事项：

* 整数默认数据类型是int；
* 超过int范围的赋值给long的数字，需要在末尾加L；eg：long a=8888888L

因为8888888超过了int的取值范围（默认为int），因此需要在末尾标注L转换类型

* 在末尾加L时最好大写，因为小写L容易与1混淆
* 浮点型：

float：4个字节

double：8个字节

注意事项：

浮点型默认数据类型是double；

给float赋值时需要在数字后加“f”or“F”（原因待查）

解答：我们无法直接定义一个byte或short类型的直接量（比如附加一个b或s作为后缀是非法的！）。我们需要用int类型的直接量来给byte和short类型的变量赋值（[链接](http://blog.sina.com.cn/s/blog_6465b05f0100pcv6.html)）

* 字符型(字符型而非字符串型)

char：2个字节（可存储1个中文字）

* 布尔型（1个字节）

理论上1/8个字符。

### 引用数据类型

类类型：

1. 系统提供的类
2. 自定义的类

接口

数组

枚举

## 数据类型的转换

隐式转换：当小的数据类型与大的数据类型进行运算时，会自动提升为大的数据类型

(大小关系：double>float>long>int>short>byte)；

强制转换：b=(byte)(b+x) 想强制转换谁，就在谁前面加数据类型

## 字符与字符串参与运算（ASCII码表）

任何数据类型用“+”与字符串连接都会产生新字符串

当“+”两边至少有一个是字符串>>字符串连接符（而不是“字符”，如果只有字符的话，“+”扮演的角色则为算数运算符）

遵循顺序流程：从上到下；从左到右

记住：0 48

a 97

A 97-32=65

## 运算符

定义：对常量和变量进行操作的符号；

* 算术运算符 + - \* / % ++ --

特殊说明：

1./ **整数相除的结果只能是整数**（小数点后面自动被省略）；**如果想得到小数，则需将其中之一变成小数**。

2. % 取余。结果与符号右边的正负号无关。

3.正整数%2的结果不是0就是1>可在以后的编程中作为条件使用

4.++（--）

1. 单独使用——放在变量前后无差别

a++ a=a+1 不完全相等，涉及数据类型转换，但是值的结果一样

++a a=a+1

1. 参与运算使用

b=a++ 当++在后，先参与运算，再自增自减；

b=++a 当++在前，先自增自减，再参与运算。

* 赋值运算符

1.基本赋值运算符

=: int a=3； 左边须为变量

2.扩展赋值运算符

+=；-=；\*=；/=；%=

均表示左右两边进行相应运算后赋值左边

* 关系运算符(比较)

结果为布尔数据类型

**==** ！= > >= < <=

# Day 3 2017/3/25

## 自习

逻辑运算符、进制（scale）

## 课堂练习——冒泡排序法

拿钻石，上楼层，依次比较。比手里大的，换成那个大的，比手里小的，换成那个小的。

# Day 4 2017/3/26

## 逻辑运算符

结果为布尔类型数据

& 与 并且 and：遇false则false，否则true；

| 或 or：遇true则true，否则false；

! 非：非真即假，非假即真；偶数个！对自身结果无影响；

^ 或非：一真一假则为true；否则为false。

&& 双与 短路与：结果与单与相同；区别在于，左边为false则右边不执行；

|| 双或 短路或；结果与单或相同；区别在于，左边为true则右边不执行；

双与、双或的优势在于提高程序的效率（不必要的操作可以略过不执行）。

## 三元（三目）运算符

格式： （关系表达式）？ 表达式1：表达式2；

解释：关系表达式的结果为true，则执行表达式1，为false，则执行表达式2.。

### 三元运算符和if条件语句的区别

三元的表达式1、2不可以是输出语句。if可以

## 键盘录入（Java语言的独特之处）

目的：为了让程序更加符合开发的数据

格式：

import java.util.Scanner; //导包

class Demo1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in); //创建键盘录入对象（输入一次即可，如果需要键盘录入多次，只需要输入多次int i = sc.nextInt()）

int i = sc.nextInt(); //将键盘录入的数据存储在i中（调用方法）

System.out.println(“xxxx”); //将结果输出

}

}

## 流程控制语句

分为三种：顺序结构语句、选择结构语句、循环结构语句。

### 顺序结构语句

从上倒下、从左到右

### 选择结构语句

有两种：if和switch

##### if

3种格式

1.if (比较表达式) { //执行流程为先执行比较表达式，然后进行后面的

语句体;

}

2.if (比较表达式) {

语句体1;

} else {

语句体2;

}

3.if (比较表达式) { //最后一个else可以省略，但是不建议省略。因为可以借助其来判断错误的类型

语句体1;

} else if {

语句体 2;

} else if {

语句体 3;

} ……

} else if {

语句体n;

} else {

语句体n+1;

}

###### 注意事项：

当if的语句体只有一句话的时候，{}可以省略

##### switch

格式：

switch (表达式) { //**表达式可以为byte、short、chart、int、String（1.7版本）（没有long），枚举**

case 值1:

语句体1;

break;

case 值2:

语句体2;

break;

case 值3:

语句体3;

break;

default:

语句体;

break;

}

注意事项：

结束一个switch只有两种方法：break和switch的右大括号。

不管default在switch的任何位置，它都最后匹配执行。

## Special points：

**1.布尔类型数据可以作为程序的“开关”。**

**具体做法：设置一个boolean类型的数据，boolean n =true; 在需要关掉开关的条件下的语句体中加入n=false；在其他条件需要输出的语句体上再加一个条件：if（n）{输出的语句体}。**

eg：（boolean开关+精简重复语句）

import java.util.Scanner;

class Test7 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入您的工龄");

int x = sc.nextInt();

Scanner sc1 = new Scanner(System.in);

System.out.println("请输入您的基本工资");

int y = sc1.nextInt();

boolean n = true;

int m=0;

if (x<15&&x>=10&&y>0) {

m=5000;

} else if (x<10&&x>=5) {

m=2500;

} else if (x<5&&x>=3) {

m=1000;

} else if (x<3&&x>=1) {

m=500;

} else if (x<1&&x>=0) {

m=200;

} else if (x<0||x>15) {

n=false;

System.out.println("您输入的工龄有误");

} else {

n=false;

System.out.println("您输入的基本工资有误");

}

if (n) {

System.out.println("您目前工作了"+x+"年，基本工资为"+y+"，应涨工资"+m+"，涨后工资为"+(m+y)+"元。");

}

}

}

**2.在{}中定义的变量（eg: int i = 12）在{}外不起作用（就消失了），但是赋值不会。{}表示的是作用域**

eg(一段错误代码):

import java.util.Scanner;

class Test7 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入您的工龄");

int x = sc.nextInt();

Scanner sc1 = new Scanner(System.in);

System.out.println("请输入您的基本工资");

int y = sc1.nextInt();

boolean n = true;

if (x<15&&x>=10&&y>0) { **//错误原因为在if的{}内定义了一个变量，而该变量出了括号，作用就消失了。因此在对最后需要输出的含有该变量的语句进行编译时，报出了“找不到变量”**

int m=5000;

} else if (x<10&&x>=5) {

int m=2500;

} else if (x<5&&x>=3) {

int m=1000;

} else if (x<3&&x>=1) {

int m=500;

} else if (x<1&&x>=0) {

int m=200;

} else if (x<0||x>15) {

n=false;

System.out.println("您输入的工龄有误");

} else {

n=false;

System.out.println("您输入的基本工资有误");

}

if (n) {

System.out.println("您目前工作了"+x+"年，基本工资为"+y+"，应涨工资"+m+"，涨后工资为"+(m+y)+"元。");

}

}

}

# Day 5 2017/3/27

## 要记住的tips

1.取出一个多位数的个、十、百、千的方法：

int x = 123;

int ge = x%10;

int shi = x/10%10;

int bai = x/100;

2.分号与左大括号不能共存

eg：while （）**;** { //如果这里有；则{}内的循环体无法执行

}

## 流程控制语句

### 循环结构

#### for（要求精通）

作用：提高复用性

格式：for(初始化表达式；关系表达式；循环后操作表达式) {

循环体；

}

注意事项：for( ; ; ) { //for的括号内可以像这样不写任何内容，但是要写的话一定要写对位置。

}

**求和思想：**

**int sum = 0;**

**for(int i = 1; i<=10;i++) {**

**sum=sum+i; //sum += i；**

**}**

#### while循环

初始化语句；

while (条件表达式) {

语句体；

循环后操作表达式 *即控制条件语句*；

}

#### if和while的死循环

while(true) {

语句体；

}

if(;;) {

语句体；

}

## 控制跳转语句

### break

使用场景：只能在switch和循环中使用

作用：结束循环。

### continue

使用场景：只能在循环中使用

作用：结束本次循环，继续下次循环

### return

作用：结束方法

## 函数（方法）

### 方法概述

方法的作用：提高代码的复用性

定义：完成特定功能的代码块

格式： 修饰符 返回值类型 方法名（参数类型 参数名1,参数类型 参数名2,参数类型 参数名3,……） {

public static void main String

须符合标识符命名规则

return 返回值；

返回值须与返回值类型相符

（返回值相當於，這個方法運行后，需要得到的結果是誰）

}

**方法的格式官方**

修饰符 返回值类型 方法名(参数类型 参数名1,参数类型 参数名2...) {

方法体语句;

return 返回值;

方法的格式说明

\* 修饰符：目前就用 public static。后面我们再详细的讲解其他的修饰符。

\* 返回值类型：就是功能结果的数据类型。

\* 方法名：符合命名规则即可。方便我们的调用。

\* 参数：

\* 实际参数：就是实际参与运算的。

\* 形式参数：就是方法定义上的，用于接收实际参数的。

\* 参数类型：就是参数的数据类型

\* 参数名：就是变量名

\* 方法体语句：就是完成功能的代码。

\* return：结束方法的语句。

\* 返回值：就是功能的结果，由return带给调用者。

### 返回值为空（面试题）

返回值为空，方法中不需要返回结果。也可以加return，但是不允许带回任何结果。它存在的意义就是结束方法

### main方法

main方法是程序的入口方法。所有程序的执行都是从main方法开始的。

### 如何写方法

1.明确返回值类型

2.明确参数列表

3.位置：类中，方法外

### 注意事项

1.单独调用,一般来说没有意义，所以不推荐。//eg：直接add(10,20).这样就没意义了，要用int之类的容器接收

2.输出调用,但是不够好。因为我们可能需要针对结果进行进一步的操作。//eg：System.out.println(add(10,20));

3.赋值调用,推荐方案。//eg: int c = add(10.20); System.out.println(c);

### 方法的重载overload

两个名称相同的方法，参数列表不同，**与返回值的类型无关**

参数列表不同体现在两个方面：

1. 参数个数不同
2. 参数类型不同（包含顺序不同，但是开发时用不到，因为无意义）

类型不同eg：public static int max(int a,int b);

public static int max(double a, double b);

**注意：** 1.public static int max(int a,int b);

public static double max(int a,int b);

此种情况不是方法的重载

public static int max(int a,int b);

public static int max(int x,int y);

此种情况也不是方法的重载

2.不能出现方法名称相同，参数列表也相同的情况

3.重载与返回值类型无关

4.实际参数的数据类型如果小于定义的形式参数，也可以调用方法。而且数据类型的大小离实际参数的数据类型大小越接近，越先调用哪个方法。**（面试题）**

如，实际参数是byte类型的，如果重载的方法有int型，也有short型。此时，会先调用short

### 方法的数据参数传递问题（面试题）

1.如果方法的形参是基本数据类型，不会改变实参

因为参数为基本数据类型时，是把主方法中的对应变量的值传递到了被调用的方法中，并根据被调用方法返回对应的值。但是，主方法中的对应变量的值并没有发生改变。

2.如果方法的形参是引用数据类型，会改变实参

因为传递的是地址值，就算所属的方法执行完毕，被释放的也是地址值，堆中的实体仍然存在，而且存放的数据就是改变后的数据。

注意：暂时这样记忆，String也是引用数据类型，但是其结论正好相反

## 课上闪闪发光的练习题们：

#### 1.

import java.util.Scanner;

class Demo8 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入一个华氏温度");

int F = sc.nextInt();

int C = (int)(F-32)\*5/9; **//实际上不需要int强转，因为”/“自动把小数给抹去了**

System.out.println("对应的摄氏度为"+C);

}

}

#### 2.

**/\*UTC是世界协调时，BJT是北京时间，UTC时间相当于BJT减去8。现在，**

**你的程序要读入一个整数，表示BJT的时和分。整数的个位和十位表示分，**

**百位和千位表示小时。如果小时小于10，则没有千位部分；如果小时是0，**

**则没有百位部分；如果分小于10分，需要保留十位上的0。如1124表示11点24分，**

**而905表示9点5分，36表示0点36分，7表示0点7分。**

**有效的输入范围是0到2359，即你的程序不可能从测试服务器读到0到2359以外的输入数据。**

**你的程序要输出这个时间对应的UTC时间，输出的格式和输入的相同，即输出一个整数，**

**表示UTC的时和分。整数的个位和十位表示分，百位和千位表示小时。如果小时小于10，**

**则没有千位部分；如果小时是0，则没有百位部分；如果分小于10分，需要保留十位上的0。**

**提醒：要小心跨日的换算。**

**输入样例：903**

**输出样例：103**

**注意:当你输入803的时候,你返回的结果应该是3**

**a=x/100 在0-23之间**

**b=x%100 在0-59之间**

**小时部分大于等于8 直接减8.**

**小时部分小于8 加上16.**

**\*/**

**我的答案**：

import java.util.Scanner;

class Demo9 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int BJT = sc.nextInt();

int a = BJT/100; **//熟练运用这种方法取出一个整数各个位置上的数字**

int b = BJT%100;

int UTC = 0;

boolean flag = true;

if (a<=23&&a>=0&&b<=59&&b>=0) {

if (a>8) {

UTC = (a-8)\*100+b;

} else {

UTC = (a+16)\*100+b;

}

} else {

System.out.println("您输入的数字有误");

flag = false;

}

if (flag = true) { **//相当于再次把true赋值给flag，这时无论flag在之前有没有变成false，这里都会变成true。因此，应该把“=true”去掉。错误的深层原因是，模糊了赋值运算符“=”和关系运算符“==”**

}

System.out.println(UTC);

}

}

**teacher的答案**

import java.util.Scanner;

public class Test01 {

public static void main(String[] args) {

//String a ;

//创建键盘录入对象,给bjt赋值

Scanner sc = new Scanner(System.in);

int bjt = sc.nextInt();

//首先对数据进行判断,看数据是否合法

if(bjt<=2359&&bjt>=0&&bjt%100<60){

//进入判断

int h = bjt/100;

if(h>=8) {

h-=8;

}else {

h+=16;

}

//h代表的是小时数+原始的分数

int utc = h\*100+bjt%100;

System.out.println(utc);

}else{

System.out.printf("您输入的数据不合法,请重新输入");

}

}

}

#### 3.

对于以下代码：

Scanner in = new Scanner(System.in);

int i = in.nextInt() % 4;

执行后，i会有几种可能性？(B)

A.1

B.7

C.3

D.4

**注意：i=0/1/2/3/-1/-2/-3**

**因为包含负数的情况，所以有7种。**

#### 4 循环语句{前加了；

对于以下代码：

for (int i=0; i<10; i++);  **//for语句的括号后跟了一个分号“；”，而不是{}。所以这个for语句的语句体就局限在了（）内，让i一直自己自增，直到不再符合条件。**

System.out.println(i); **//for里面int定义的i在出了（）后，作用就消失了**

for循环后，变量i的值是多少？(D)

A.11

B.9

C.10

D.循环结束后i不存在

#### 5 while ( !e )

在代码：

while ( !e )

中，!e等价于：(D)

A.e!=0;

B.e==0

C.e

D.!(e==true)

**解答：假设e=true, !e为false；！（e==true）为false**

**假设e=false，（e==true）为false；！（e==true）为true**

**可能忽略的知识点：比较表达式输出的结果为true或者false。（e==true）的结果为true/false。**

#### 6.一行输出5个数后，换行

/\*1.分析以下需求，并用代码实现：

(1)打印1到100之内的整数，但数字中包含9的要跳过

(2)每行输出5个满足条件的数，之间用空格分隔

(3)如：1 2 3 4 5

\*/

class ForDemo {

public static void main(String[] args) { **//主方法**

int j = 0; **//输出格式为5个一行的关键所在。j相当于记录一行中已经输出了多少个数字，当j=5时，就需要插入一个回车**

for(int i = 1;i <= 100;i++) {

if(i%10 == 9 || i/10 == 9) {//跳过个位和十位为9的整数

continue;

}else {

if(j == 5){

System.out.println();//相当于回车

j = 0;

}

j ++; //j ++这个语句不可以放在if（j==5）这个条件之前，因为这样一来，println()在i=5之前就输出了。结果第五个数字就出现在了第二行。

}

System.out.print(i + " ");

}

}

}

**需要注意的点：**

对于每行固定输出5个值这样的格式要求，可以采用引入新变量j的方法，让j来记录截至目前该行已经录入多少个数字。当j=5时，输入一个换行的指令*System.out.println()*

# (基础班)自习 2017/3/30

## 循环语句经典思想——统计思想

引入一个变量，每执行一次循环该变量自增1

## 第三种循环语句——do while

循环结构do...while语句的格式：

\*

do {

循环体语句;

}while(判断条件语句);

完整格式；

初始化语句;

do {

循环体语句;

控制条件语句;

}while(判断条件语句);

## 三种循环语句的区别

do while的特别之处：不管条件是否满足，都至少执行一次；

for的特别之处：其定义的变量在执行后会被释放；

while的特别之处：初始化变量在执行后还能继续使用

建议在使用for和while都可以的时候，使用for。因为节省内存空间，可以提高效率。

## 代码优化

'\x' x表示任意，\是转义符号,这种做法叫转移字符。

'\t' tab键的位置

'\r' 回车

'\n' 换行

'\"'

'\''

另外，其实要打印“或者‘，除了用转义符，也可以用”’“和‘”’，这样的办法。

## 控制跳转语句的标号

一般用于多层循环嵌套时，想跳出外循环时使用

class Demo3\_Mark { //mark 标记

public static void main(String[] args) {

a: for (int i = 1;i <= 10 ;i++ ) { //a就是标号,只要是合法的标识符即可

System.out.println("i = " + i);

b: for (int j = 1;j <= 10 ;j++ ) {

System.out.println("j = " + j);

break b;

}

}

}

}

## 方法之输出星形及其调用

写方法之前先明确两点：

1.明确返回值类型；//输出星形没有明确返回值类型

2.明确参数列表；//

方法三种调用方式：

1.单独调用(一般不使用，因为如果要输出一个特定值，如只写int a就没有意义了。但是这里是要打印“\*”，所以单独调用就好)

2.输出调用 System.out.println(getMax(i,j,k));(在这里，输出调用和赋值调用都不适用)

3.赋值调用 int a = getMax(i,j,k);//getMax为自己定义的方法

## 数组

### **存在的原因：**

相当于一个容器，为了存储同一个数据类型的多个值

### 数组概念

\* 数组是存储同一种数据类型多个元素的集合。也可以看成是一个容器。

\* 数组既可以存储基本数据类型，也可以存储引用数据类型。

### 格式

数据类型[] 数组名 = new 数据类型[数组长度];

**数组 标识符 创建新的对象/实体**

**几个括号就代表几维数组**

### 数组初始化

为数组开辟连续的内存空间，并为数组中的每个元素赋予初始化值

可以存储同一种数据类型的多个元素。一旦初始化，长度就不可改变

#### 动态初始化

只指定长度，由系统给出初始化值

格式：int[] arr = new int[5]

系统默认给出的整数的初始值为0

系统默认给出的浮点数的初始值为0.0

系统默认给出的布尔型的初始值为false

系统默认给出的字符型的初始值为’\u0000’

引用数据类型的默认初始值为null

#### 静态初始化

给出初始化值，由系统定长度

格式：int[] arr = new int[] {1,2,3,4,5}

**注意：不允许手动给出数组的长度**

### 越界和空指针（重要）

\* a:ArrayIndexOutOfBoundsException:

\* b:数组索引越界异常

\* c:原因：你访问了不存在的索引。

\*

NullPointerException:空指针异常

\* 原因：数组的地址值被赋值给null，**又访问了数组中的元素。**

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

\* arr = null;

\* System.out.println(arr[0]);

### 数组的遍历

就是依次输出数组中的每一个元素。

\* 数组的属性:arr.length数组的长度

\* 数组的最大索引:arr.length - 1;

public static void print(int[] arr) {

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

System.out.print(arr[i] + " ");

}

}

### 数组获取最值(获取数组中的最大值最小值)

public static int getMax(int[] arr) {

int max = arr[0];

for (int i = 1;i < arr.length ;i++ ) { //从数组的第二个元素开始遍历

if (**max < arr[i]**) { //如果max记录的值小于的数组中的元素（**这一步十分genius**）

max = arr[i]; //max记录住较大的

}

}

return max;

}

### 数组元素反转(就是把元素对调)

public static void reverseArray(int[] arr) {

for (int i = 0;i < arr.length / 2 ; i++) {

//arr[0]和arr[arr.length-1-0]交换

//arr[1]和arr[arr.length-1-1]交换

//arr[2]和arr[arr.lentth-1-2]

//...

int temp = arr[i];

arr[i] = arr[arr.length-1-i];

arr[arr.length-1-i] = temp;

}

}

JVM的内存是如何划分的

栈：存储局部变量，方法在执行时都会进栈，完毕后立刻释放

堆：存储new出来的对象

# Day 3 2017/4/1

## 课上闪闪发光的练习题们：

### 1.if

/\* 键盘录入一个整数，使用if语句判断x取值范围

x和y的关系满足如下：

x>=3 y = 2x + 1;

-1<=x<3 y = 2x;

x<=-1 y = 2x – 1;

根据给定的x的值，计算出y的值并输出。\*/

**package** com.itheima;

**import** java.util.Scanner;

**public** **class** IfDemo {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.*in*);

System.*out*.println("請輸入一個整數");

**int** x = sc.nextInt();

**int** y ;

**if** (x >= 3) {

y = 2\*x + 1;

}**else** **if**(x < 3&&x >= -1) {

y = 2\*x;

}**else** **if**(x < -1) {

y = 2\*x - 1;

}**else** {

System.*out*.println("您輸入的x有誤");

**y = 0;** //因为上面声明变量y时没有给它赋值，而计算机会认为，if的每一种情况都需要能输出y的值们因为最后要输出y的值。所以在最后一种情况中，需要特别赋值给y。当然，在一开始声明变量y时直接赋值给它也是可以的（更好）。

}

System.*out*.println("y="+y);

}

}

### 2.

/\*第四题：分析以下需求，并用代码实现：

1.键盘录入三个整数，按照从小到大的顺序输出

2.如果用户输入的是3 2 1，程序运行后打印格式"按照从小到大排序后的顺序为：1 2 3"\*/

**package** day03;

**import** java.util.Scanner;

**public** **class** Test04 {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.*in*);

**int** a = sc.nextInt();

**int** b = sc.nextInt();

**int** c = sc.nextInt();

**int** temp;

**if**(a<b) {

temp = a;

a = b;

b = temp;

}

**if**(a<c) {

temp = a;

a = c;

c = temp;

}

**if**(b<c) {

temp = b;

c = b;

c = temp;

}

System.*out*.println("按照从小到大排序后的顺序为："+c+" "+b+" "+a+" ");

}

}

3.

/\*第八题:分析以下需求，并用代码实现：

1.按照从大到小的顺序输出四位数中的个位+百位=十位+千位(3553,2332,1166,8228,3773)的数字及个数

2.每行输出5个满足条件的数，之间用空格分隔

3.如：9999 9988 9977 9966 9955 \*/

**package** day03;

**public** **class** Test08 {

**public** **static** **void** main(String[] args) {

**int** count = 0;

**for**(**int** i = 1000;i<=9999;i++) {

**int** a = i%10;

**int** b = i/10%10;

**int** c = i/100%10;

**int** d = i/1000;

**if**(a+c==c+d) {

System.*out*.print(i+"\t");

count++;

**if**(**count%5==0**) { **//这一个变量兼具记总数和记列数的功能**

System.*out*.println();

}

}

}

System.*out*.println(count);

}

}

# Day 4 2017/4/3

## 随机数

产生随机数的方法

1. 导包 import java.util.Random
2. 创建对象 Random r = new Random();
3. 获取随机数 int x = r.nextInt(y);

y为自由填入的整数，限定随机数产生的范围：[0,y）

如果想改变这个范围，可以在获取随机数时，视具体情况而定，在末尾做算术运算

eg: int x = r.nextInt(5)+1

# Day5 2017/4/5

## 注意：

int[] arr = new int[] {1,2,3,4,’a’};//成立，遍歷出來的a是97

int[] arr = new int[] {1,2,3,4,“a”};//不成立，因爲是字符串而不是字符

**byte arr = new int[] {1,2,3,4,’a’};//成立，因爲a為97，在byte的範圍内**

## 課上闪闪发光的练习题们：

### 1.打印水仙花数，同时数出水仙花数的个数

**public** **class** MethodFlower {

**public** **static** **void** main(String[] args) {

*countFlower*(); //没有参数列表，所以括号要空着

}

**public** **static** **void** countFlower() {

**int** count = 0;

**for**(**int** i = 100;i <=999;i++) {

**int** ge = i%10;

**int** shi = i/10%10;

**int** bai = i/100;

**if**(ge\*ge\*ge+shi\*shi\*shi+bai\*bai\*bai == i) {

System.*out*.println(i);

count++;

}

}

System.*out*.println("水仙花數有"+count+"個");

}

}

# Day 6 2017/4/6

## Eclipse断点调试

### 作用：

1.查看程序的执行流程；解决bug

2.public class Test2 {

### 常用操作：

A:什么是断点：

就是一个标记，从哪里开始。

B:如何设置断点：

你想看哪里的程序，你就在那个有效程序的左边双击即可。

C:在哪里设置断点：

哪里不会点哪里。

目前：我们就在每个方法的第一条有效语句上都加。

D:如何运行设置断点后的程序：

右键 -- Debug as -- Java Application

E:看哪些地方：

Debug：断点测试的地方

在这个地方，记住F6，或者点击也可以。一次看一行的执行过程。

Variables：查看程序的变量变化

ForDemo：被查看的源文件

Console：控制台

F:如何去断点：

再次双击即可

找到Debug视图，Variables界面，找到Breakpoints，并点击，然后看到所有的断点，最后点击那个双叉。

### 注意

断点必须加在有效的语句上

要想看被调用方法的执行流程，要在方法中也加断点

## 综合练习

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("请输入月份(1-12)：");

int month = sc.nextInt();

switch(month) { **//利用case的穿透性**

case 1:

case 2:

case 12:

System.out.println("冬季");

break;

case 3:

case 4:

case 5:

System.out.println("春季");

break;

case 6:

case 7:

case 8:

System.out.println("夏季");

break;

case 9:

case 10:

case 11:

System.out.println("秋季");

break;

default:

System.out.println("你输入的月份有误");

break;

}

}

}

# Day 7 2017/4/8

## 面向对象

### 思想概述

面向过程：如C语言。强调实现每一个功能的步骤。

面向对象：基于面向过程。是把实现功能的过程封装在了对象里。

特点：A.符合人们的思考习惯；

B.把复杂的问题简单化；

C.从执行者变成指挥者。

### 类与对象

#### 类

Java语言最基本的单位是类。类是对对象的描述（蓝图）。是一组相关属性与行为 的集合

##### 属性

事物的描述信息——成员变量

##### 行为

事物能做什么——成员方法

#### 对象

对该类事物的具体体现

### 类的使用

我们如何拥有一个类的对象呢?

\* 创建对象就可以了?

\* 我们如何创建对象呢?

\* 格式：类名 对象名 = new 类名();

\* 对象如何访问成员呢?

\* 成员变量：对象名.变量名

\* 成员方法：对象名.方法名(...)

### 注意事项：

#### 1.基本类与测试类

基本类：无主方法；原因是main是程序的入口，要运行程序必须有main，有main则必须运行。而基本类无需自己独立运行。所以不需要main

测试类：有主方法

#### 2.成员变量和局部变量的区别（面试题）

a.位置不同：

成员变量存储在类中方法外，局部变量存储在方法中和方法声明上；

成员变量存在于堆内存，局部变量存储在栈内存。

b.生命周期：

成员变量随对象的创建而存在，随对象的消失而消失；

局部变量随方法的调用而存在，随方法调用的结束而消失；

c.初始化值：

成员变量有默认值，无需初始化；局部变量无默认值，需要初始化才能使用。

#### 3.成员变量的其他注意点

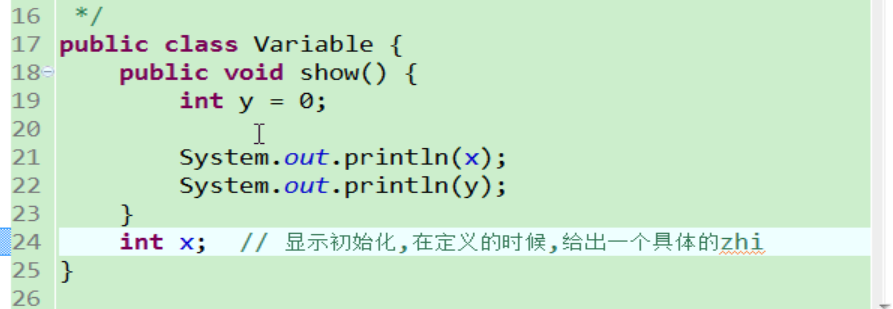
a.作用域：整个类中有效；

b.使用原则：就近原则——局部位置有，就直接使用，没有就到成员位置上找

c.赋值的方法：set方法；带参构造方法。

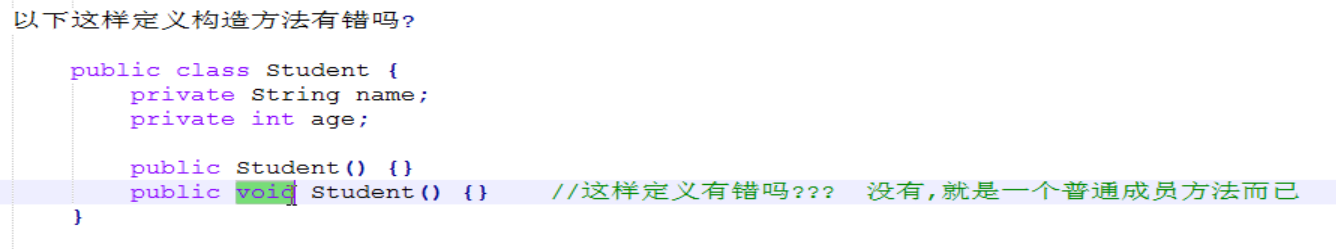
#### 4.成员变量放在方法后面也是可以的

如下图：x在方法后面也是可以的



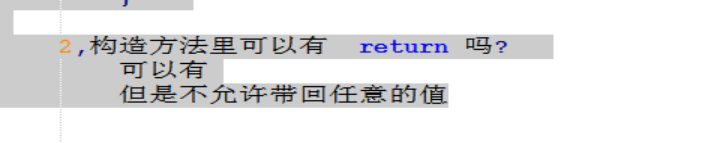
#### 5.面试题

构造方法是public加上类名，所以如果额外加上了void，是否会报错（如下图）



#### 6.面试题

如图



## Private关键字

### 特点

a:是一个权限修饰符。

b:可以修饰成员(成员变量和成员方法)

c:被private修饰的成员只在本类中才能访问。

### 存在的意义

通过对象直接访问成员变量，会存在数据的安全问题

### set/get方法

使用了private关键字后，在测试类中无法直接对成员变量赋值，则需要相应的提供getXxx()和setXxx()用于获取和设置成员变量的值,方法用public修饰

public class Student {

private String name;

private int age;

public void setName(String n) {

name = n;

}

public String getName() {

return name;

}

public void setAge(int a) {

age = a;

}

public int getAge() {

return age;

}

}

### 基本类的标准代码:

class 类名{

//成员变量

//对应的 setXxx() 和 getXxx()

//成员方法

}

### 封装的概述和好处:

A:封装概述

是面向对象三大特征之一

是面向对象编程语言对客观世界的模拟，客观世界里成员变量都是隐藏在对象内部的，外界无法直接操作和修改。就像刚才说的年龄。

B:封装原则：

将不需要对外提供的内容都隐藏起来。

把属性隐藏，提供公共方法对其访问。

成员变量private，提供对应的getXxx()/setXxx()方法

C:好处：

通过方法来控制成员变量的操作，提高了代码的安全性

把代码用方法进行封装，提高了代码的复用性

this关键字由来和使用:

## this关键字

定义：this就是对所在函数所属对象的引用

this是什么？this就是对象，当前对象

A:方法被哪个对象调用，this就代表哪个对象

B:什么时候使用this呢

局部变量和成员变量重名

构造方法压栈时，里面有this，代表被创建的那个对象的地址值。因此构造方法可以找到对应的对象给其赋值。

只要在本类中用到了本类的对象，通常都用this

### 注意事项

this也可以用于构造函数中调用其他构造函数（普通函数不可以调用构造函数），可以提高代码的复用性

注意只能定义在构造函数的第一行。因为初始化的动作要先执行。

代码：

**public** Teacher(String name) {

**this**.name = name;

}

**public** Teacher(String name,**int** age) {

**this**(name);//this指代当前对象，因为构造函数的重载区别在于参数列表的不同，因此可以用往括号里传的参数的不同来区分不同的构造函数

**this**.age = age;

}

## 构造方法

定义：构建创造对象时所用到的函数

### 作用

主要用来给对象的数据进行初始化

创建对象都必须通过构造函数初始化

### 格式

a:方法名与类名相同

b:没有返回值类型，连void都没有

c:没有具体的返回值

### 构造方法注意事项与重载

如果你不提供构造方法，系统会给出默认构造方法

如果你提供了构造方法，系统将不再提供

构造方法也是可以重载的,重载条件和普通方法相同

### 构造函数与setter方法共存的意义

构造函数是对象一存在就具备的内容，setter方法是对象存在后，对其进行的操作。

如果 改变对象的某个参数只能通过创造对象进行，那么这个对象已经不是一开始的那个对象了。有的时候我们只需要改变对象的某个参数，而不想重新创造对象，这时就可以使用setter方法

### 一般函数与构造函数的区别：

构造函数：对象创建后，就会调用与之对应的构造函数，对对象进行初始化

对象创建后，只调用一次

一般函数：对象创建后，当需要函数功能时，对函数进行调用后才会访问

对象创建后，根据需求，可被调用多次

### 注意事项

一般的成员方法也可以用类名命名

构造方法中可以有return，用来判断所附的值是否符合条件。但是一般很少用，因为实际工作中，传来的参数是已经校验好了的。

**Student（int age）{**

**if(age<0) {**

**return;**

**}else {**

**this.age = age;**

**}**

**}**

## 类名作为方法的形式参数

### 案例代码十四:

package com.itheima\_10;

public class Student {

public void study() {

System.out.println("好好学习,天天向上");

}

}

package com.itheima\_10;

public class Teacher {

public void test(Student s) {//接收传递过来的Student对象的地址值

s.study();

}

}

package com.itheima\_10;

//需求： 调用Teacher的test方法

//类名作为形式参数：其实这里需要的是该类对象。

public class Test {

public static void main(String[] args) {

Teacher t = new Teacher();

Student s = new Student();

t.test(s);

}

}

## 类名作为返回值案例

### 案例代码十五:

package com.itheima\_11;

public class Student {

public void study() {

System.out.println("好好学习,天天向上");

}

}

package com.itheima\_11;

public class Teacher {

public Student getStudent() {

Student s = new Student();

return s;//返回的是Student对象的地址值

}

}

package com.itheima\_11;

//需求： 通过Teacher得到Student对象，然后调用Student类的方法

//如果方法的返回值是类名：其实返回的是该类的对象

public class Test {

public static void main(String[] args) {

Teacher t = new Teacher();

Student s = t.getStudent();

s.study();

}

}

# Day 8 2017/4/9

## String

### 构造方法（3种）

String(String original):把字符串数据封装成字符串对象

String(char[] value):把字符数组的数据封装成字符串对象

String(char[] value, int index, int count):把字符数组中的一部分数据封装成字符串对象

代码：

**public** **class** ConstructMethod {

**public** **static** **void** main(String[] args) {

//String(String original):把字符串数据封装成字符串对象

String s0 = **new** String("hello");

System.***out***.println(s0);

//String(char[] value):把字符数组的数据封装成字符串对象

**char**[] chs = {'a','b','c','d'};

String s = **new** String(chs);

System.***out***.println(s);

//String(char[] value, int index, int count):把字符数组中的一部分数据封装成字符串对象

String s1 = **new** String(chs,2,2);

System.***out***.println(s1);

}

}

### 判断功能（4种）

boolean equals(Object obj):比较字符串的内容是否相同

boolean equalsIgnoreCase(String str):比较字符串的内容是否相同,忽略大小写

boolean startsWith(String str):判断字符串对象是否以指定的str开头

boolean endsWith(String str):判断字符串对象是否以指定的str结尾

代码示例：

**public** **class** Determination {

**public** **static** **void** main(String[] args) {

String s = "Hello";

String s1 = "hello";

String s2 = **new** String("Hello");

System.***out***.println("s equals? 1:"+s.equals(s1));

System.***out***.println("s equals? s2:"+s.equals(s2));

System.***out***.println("s==s2:"+(s==s2));

//直接赋值，把常量池的对应字符串的地址值赋过来

//创建对象，在堆中创建新的对象，常量池中对应的字符串地址赋给堆中的相应位置，

//堆中对象的地址值再赋给栈中。“equals”比较的是字符串内容，“==”比较的是地址值

System.***out***.println("s equalsIgnoreCase s1:"+s.equalsIgnoreCase(s1));

System.***out***.println("s startsWith He:"+s.startsWith("He"));

System.***out***.println("s endWith lo:"+s.endsWith("lo"));

}

}

### 获取功能（5种）

int length():获取字符串的长度，其实也就是字符个数

char charAt(int index):获取指定索引处的字符

int indexOf(String str):获取str在字符串对象中第一次出现的索引

String substring(int start):从start开始截取字符串

String substring(int start,int end):从start开始，到end结束截取字符串。包括start，不包括end

代码示例：

**public** **class** Getting {

**public** **static** **void** main(String[] args) {

String s = "Forest";

System.***out***.println("字符串长度："+s.length());

System.***out***.println("指定索引处的字符："+s.charAt(3));

System.***out***.println("r第一次出现时对应的索引："+s.indexOf("r"));

System.***out***.println("从索引1开始截取字符串："+s.substring(1));

System.***out***.println("从索引1开始截取字符串，到5截止："+s.substring(1, 5));//end填入索引5，结果不包含5.[0,5)

}

}

### 其他功能（2种）

去除字符串两端空格

\* String trim()

\* 按照指定符号分割字符串

\* String[] split(String str)

代码示例：

**public** **class** Others {

**public** **static** **void** main(String[] args) {

String s = " Fore st ";

//去除字符串两端空格 String trim()

System.***out***.print(s.trim());

//按照指定符号分割字符串 String[] split(String str)

String[] stArr = s.split(" ");

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

System.***out***.println(stArr[i]);

}

}

}

### String面试题

String s = new String(“Hello”)

以上代码会创建几个String对象？2个（[详解](http://blog.csdn.net/u012110719/article/details/45671557)）

以上代码会创建几个对象？3个（还有字符数组）

### java编译器的优化机制

常量优化机制。String s = “abc”; “abc”第一次出现的时候，在常量池中创建一个“abc”，但是第二次如果以赋值的形式出现，即String s1 = “abc”，那么就不会再次创建对象，而是拿着已经存在的对象，直接使用。

## StringBuilder

### 构造方法

StringBuilder()

### 成员方法

public int capacity():返回当前容量 (理论值)

public int length():返回长度(已经存储的字符个数)

public StringBuilder append(任意类型):添加数据，并返回自身对象

public StringBuilder reverse():反转功能

## String和StringBuilder的互相转换

StringBuilder -- String

public String toString():通过toString()就可以实现把StringBuilder转成String

String 类的构造(StringBuilder sb);

String -- StringBuilder

StringBuilder(String str):通过构造方法就可以实现把String转成StringBuilder

利用 StringBuilder 里的append(String str);

代碼示例

**public** **class** StringBuilderString {

**public** **static** **void** main(String[] args) {

StringBuilder sb = **new** StringBuilder("hello");

//StringBuilder轉String

//StringBuilder的成員方法toString()

String s = sb.toString();

System.***out***.println(s);

//String的構造方法

String s1 = **new** String(sb);

System.***out***.println(s1);

//String轉StringBuilder

//StringBuilder的構造方法

StringBuilder sb1 = **new** StringBuilder(s);

System.***out***.println(sb1);

//StringBuilder的成員方法append(String str);

sb.append(s);

System.***out***.println(sb);

}

}

# Day 9 2017/4/10

## 集合概述

A:我们学习的是面向对象编程语言，而面向对象编程语言对事物的描述都是通过对象来体现的。

为了方便对多个对象进行操作，我们就必须对这多个对象进行存储，而要想对多个对象进行存储， 就不能是一个基本的变量，而应该是一个容器类型的变量。

B:到目前为止，我们学习过了哪些容器类型的数据呢？

StringBuilder,数组。StringBuilder的结果只能是一个字符串类型，不一定满足我们的需求。

所以，我们目前只能选择数组了，也就是我们前面学习过的对象数组。但是，数组的长度是固定的， 如果有时候元素的个数不确定的,我们无法定义出数组的长度,这个时候，java就提供了集合类供我们使用。

## 集合的功能

### 增加元素

在出现E的地方我们使用引用数据类型替换即可

举例：ArrayList<String>,ArrayList<Student>

构造方法：

ArrayList()

添加元素：

\* public boolean add(E e):添加元素

\* public void add(int index,E element):在指定的索引处添加一个元素

代码示例

**public** **class** ArrayListDemo {

**public** **static** **void** main(String[] args) {

//创建集合对象

ArrayList<String> array = **new** ArrayList<String>();

//add(E e):添加元素

array.add("hello");

array.add("world");

array.add("java");

//add(int index,E element):在指定的索引处添加一个元素

//array.add(1, "android");

System.*out*.println("array:"+array);

}

}

### 删改查元素

A:获取元素

public E get(int index):返回指定索引处的元素

B:集合长度

public int size():返回集合中的元素的个数

C:删除元素

public boolean remove(Object o):删除指定的元素，返回删除是否成功

public E remove(int index):删除指定索引处的元素，返回被删除的元素

D:修改元素

public E set(int index,E element):修改指定索引处的元素，返回被修改的元素

## 学生管理系统案例

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** StudentManagementSystem {

**public** **static** **void** main(String[] args) {

ArrayList<Student> list = **new** ArrayList<Student>();

**while**(**true**) {

System.***out***.println("-------------欢迎来到学生管理系统-------------");

System.***out***.println("1 查看所有学生");

System.***out***.println("2 添加学生");

System.***out***.println("3 删除学生");

System.***out***.println("4 修改学生");

System.***out***.println("5 退出");

System.***out***.println("请输入您的选择：");

Scanner sc = **new** Scanner(System.***in***);

String choice = sc.nextLine();

**switch** (choice) {

**case** "1":

//find all students

*findAll*(list);

**break**;

**case** "2":

//add students

*addStudent*(list);

**break**;

**case** "3":

//delete students

*deleteStudent*(list);

**break**;

**case** "4":

//reset students

*resetStudent*(list);

**break**;

**case** "5":

//exit

**default**:

System.***out***.println("感谢您的使用！");

**return**;

}

}

}

**public** **static** **void** findAll(ArrayList<Student> list) {

**if**(list.size()==0) {

System.***out***.println("很抱歉，暂无任何学生信息，请重新进行选择");

}**else** {

System.***out***.println("学号\t姓名\t年龄\t地址");

**for**(**int** i = 0;i < list.size();i++) {

Student s = list.get(i);

System.***out***.println(s.getId()+"\t\t"+s.getName()+"\t"+s.getAge()+"\t\t"+s.getAddress());

}

}

}

**public** **static** **void** addStudent(ArrayList<Student> list) {

Scanner sc = **new** Scanner(System.***in***);

String id;

**while**(**true**) {

System.***out***.println("请输入学号:");

id = sc.nextLine();

**boolean** flag = **true**;

**for**(**int** i = 0;i < list.size();i++) {

Student s = list.get(i);

**if**(s.getId().equals(id)) {

System.***out***.println("您输入的学号已经存在");

flag = **false**;

}

}

**if**(flag) {

System.***out***.println("请输入姓名");

String name = sc.nextLine();

System.***out***.println("请输入年龄");

String age = sc.nextLine();

System.***out***.println("请输入地址");

String address = sc.nextLine();

Student s1 = **new** Student(id,name,age,address);

list.add(s1);

System.***out***.println("注册成功");

**return**;

}

}

}

**public** **static** **void** deleteStudent(ArrayList<Student> list) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("请输入要删除的学生学号");

String id = sc.nextLine();

**int** index = -1;

**for**(**int** i = 0;i < list.size();i++) {

Student s = list.get(i);

**if**(s.getId().equals(id)) {

index = i;

**break**;

}

}

**if**(index == -1) {

System.***out***.println("抱歉，不存在您输入的学号");

}**else** {

list.remove(index);

System.***out***.println("删除成功");

}

}

**public** **static** **void** resetStudent(ArrayList<Student> list) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("请输入要修改的学生学号");

String id = sc.nextLine();

**int** index = -1;

**for**(**int** i = 0;i < list.size();i++) {

Student s = list.get(i);

**if**(s.getId().equals(id)) {

index = i;

}

}

**if**(index == -1) {

System.***out***.println("您输入的学号不存在，请重新输入");

}**else** {

System.***out***.println("请输入姓名");

String name = sc.nextLine();

System.***out***.println("请输入年龄");

String age = sc.nextLine();

System.***out***.println("请输入地址");

String address = sc.nextLine();

Student s1 = **new** Student(id,name,age,address);

list.set(index,s1);

System.***out***.println("修改成功");

}

}

}

# day 10 2017/4/13

## IO流

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

/\*第一题：分析以下需求，并用代码实现

1.将a.txt内容复制到b.txt文件中

2.用多种方式实现复制

(1)使用字符流单字符循环复制

(2)使用字符流利用字符数组实现复制

(3)使用高效字符缓冲流实现复制\*/

**public** **class** Test1 {

**public** **static** **void** main(String[] args) **throws** Exception {

String desStr = "b.txt";

String orgStr = "a.txt";

// FileChar(orgStr,desStr);

// FileArray(orgStr,desStr);

// BufferedChar(orgStr,desStr);

// BufferedArray(orgStr,desStr);

*BufferedSpecial*(orgStr,desStr);

// fr.close();

// fw.close();

}

**public** **static** **void** FileChar(String orgStr,String desStr) **throws** Exception {

FileReader fr = **new** FileReader(orgStr);

FileWriter fw = **new** FileWriter(desStr);

**int** ch;

**while**((ch=fr.read())!=-1) {

fw.write(ch);

}

fr.close();

fw.close();

}

**public** **static** **void** FileArray(String orgStr,String desStr) **throws** Exception {

FileReader fr = **new** FileReader(orgStr);

FileWriter fw = **new** FileWriter(desStr);

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len = fr.read(chs))!=-1) {

fw.write(chs,0,len);

}

fr.close();

fw.close();

}

**public** **static** **void** BufferedChar(String orgStr,String desStr) **throws** Exception {

BufferedReader br = **new** BufferedReader(**new** FileReader(orgStr));

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(desStr));

**int** ch;

**while**((ch = br.read())!=-1) {

bw.write(ch);

}

br.close();

bw.close();

}

**public** **static** **void** BufferedArray(String orgStr, String desStr) **throws** Exception{

BufferedReader br = **new** BufferedReader(**new** FileReader(orgStr));

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(desStr));

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len = br.read(chs))!=-1) {

bw.write(chs,0,len);

}

bw.close();

br.close();

}

**public** **static** **void** BufferedSpecial(String orgStr, String desStr) **throws** Exception{

BufferedReader br = **new** BufferedReader(**new** FileReader(orgStr));

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(desStr));

String s;

**while**((s=br.readLine())!=**null**) {

bw.write(s);

bw.newLine();

}

br.close();

bw.close();

}

}

# 就业班

# Day 1 2017/4/19

## Static

### 概述

静态修饰符，用于修饰类中的成员。用在不想创建对象但想直接调用方法的时候。

为什么main是静态的？

### 特点

1. 被多个对象共享
2. **被static修饰的成员可以并且建议通过类名直接访问**

访问静态成员的格式:

类名.静态成员变量名

类名.静态成员方法名(参数)

1. 静态的加载优先于对象，随着类的加载而加载（加载到方法区，方法区内部有共享区）
2. static用于修饰成员
3. static修饰的数据是共享数据，对象存储的是特有数据

### 注意事项

1. 静态成员只能访问静态成员
2. 非静态成员都可以访问
3. 静态方法中无this，无super。

静态是随着类的加载而加载，this是随着对象的创建而存在。而静态比对象先存在

d．主函数是静态的（待添加主函数的解释）

### 静态的优缺点

优点：

对对象的共享数据提供单独空间的存储，节省空间，没有必要每一个对象都存储一份（**节省空间**）

可以直接被类名调用,不用在堆内存创建对象（**不用创建对象**）

静态成员可以通过类名直接访问,相对创建对象访问成员方便（**通过类名直接访问，方便**）

缺点：

访问出现局限性。（**静态虽好，但只能访问静态**）

### 静态应用

#### Math类使用

A:Math 类包含用于执行基本数学运算的方法。数学操作常用的类。

B:Math类的构造方法被private,无法创建对象,也就无法通过对象来访问Math类中的成员

C:Math类中所有的成员都被静态修饰,因此我们可以直接通过类名访问

代码示例**：**

**public** **class** MathDemo {

**public** **static** **void** main(String[] args) {

//Math:包含了一些基本的数学运算方法

//static double PI

//System.out.println(Math.PI);

//static double abs(double a) :返回绝对值

//System.out.println(Math.abs(15));

//System.out.println(Math.abs(-10));

//static double ceil(double a) 天花板 向上取整

//System.out.println(Math.ceil(1.2));

//System.out.println(Math.ceil(1.6));

//static double floor(double a) 地板 向下取整

//System.out.println(Math.floor(1.2));

//System.out.println(Math.floor(1.6));

//static long round(double a) ：四舍五入

//System.out.println(Math.round(1.2));

//System.out.println(Math.round(1.6));

//static double max(double a, double b)

//System.out.println(Math.max(3, 4));

//static double pow(double a, double b) :返回第一个参数的第二个参数次幂

//System.out.println(Math.pow(3, 2));

//static double random() :返回一个随机数，大于零且小于一

System.***out***.println(Math.*random*());

}

}

### 类变量与实例变量辨析（官方版）

A:类变量:其实就是静态变量（类类型变量不是类变量。类类型变量是类类型的变量，Student s, s, s就是类类型变量）

定义位置:定义在类中方法外

所在内存区域:方法区

生命周期:随着类的加载而加载

特点:无论创建多少对象,类变量仅在方法区中,并且只有一份

B:实例变量:其实就是非静态变量

定义位置:定义在类中方法外

所在内存区域:堆

生命周期:随着对象的创建而加载

特点:每创建一个对象,堆中的对象中就有一份实例变量

### 静态变量（类变量）和成员变量（实例变量）的区别（小虎版）

1.所属不同（名称不同）

静态变量属于类，所以称为类变量

成员变量属于对象，所以称为实例变量（对象变量）

2.内存中位置不同

静态变量存储于方法区的静态区

成员变量存储于堆内存

3.内存出现的时间不同（生命周期不同）

静态变量随着类的加载而加载，随着类的消失而消失

成员变量随着对象的创建而存在，随对象消失而消失

4.调用方式不同

静态变量可以通过类名调用，也可以通过对象调用

成员变量只能通过创建对象调用

## 代码块

### 局部代码块

局部代码块是定义在方法或语句中

*以”{}”划定的代码区域，此时只需要关注作用域的不同即可*

*方法和类都是以代码块的方式划定边界的*

***作用：****节省内存，控制变量的生命周期*

### 构造代码块

构造代码块是定义在类中成员位置的代码块**（类中的独立代码块）**

**作用：**抽取构造方法中重复的部分*，提高代码的复用性*

*优先于构造方法执行，构造代码块用于执行所有对象均需要的初始化动作*

*每创建一个对象均会执行一次构造代码块。*

*类中方法外*

#### 构造函数与构造代码块的区别

构造函数：给对应的对象进行针对性的初始化（针对性）

构造代码块：给所有对象进行初始化（通用性）

### 静态代码块

A:静态代码块是定义在成员位置，使用static修饰的代码块

*它优先于主方法执行、优先于构造代码块执行，当以任意形式第一次使用到该类时执行。*

*该类不管创建多少对象，静态代码块只执行一次。*

***作用：****可用于给静态变量赋值，用来给****类****进行初始化，如驱动。（构造函数给****对象****初始化）*

*先于主方法，先于构造代码块，先于构造方法*

使用场景**：一般使用静态代码块的情况为，类中全是静态成员。**

### 同步代码块

# Day 2 2017/4/21

## 继承

概述：子类继承父类的非私有的成员变量和成员方法

注意：构造方法不能继承

### 格式

public class 子类extends 父类 {

}

*举例：*

class CatFamily{ // 猫科

int age;

String color;

String leg;

}

class Cat extends CatFamily{} // 猫

class Tiger extends CatFamily{} // 老虎

class Chicken extends CatFamily{}// **错误的思想**

注：Java中的继承描述的是类与类之间的所属关系，只有当一个类是属于另一个类的一种时，

满足is a关系时，才能使用继承，**不能为了继承而继承**

### 特点

1. 一个父类可以有多个子类
2. Java语言的类只支持单继承，一个子类只能有一个父类
3. 可以多层继承。一个父类可以是另一个类的子类

### 继承成员变量的特点

1. 子类只能继承父类的非私有成员变量
2. 就近原则（谁离得近就用谁）

如果有局部变量就使用局部变量

如果没有局部变量，有子类的成员变量就使用子类的成员变量

如果没有局部变量和子类的成员变量，有父类的成员变量就使用父类的成员变量

3. super:可以获取父类的成员变量和成员方法,用法和this是相似的

### 继承成员方法的特点

1. 子类只能继承父类的非私有成员方法
2. 子类中没有方法，就调用父类的
3. 方法的重写：在子父类当中，子类的方法和父类的完全一样（**方法名和参数列表**），子类重写了父类的方法（覆盖），当子类重写了父类的方法之后，使用子类对象调用的就是子类的方法

### 方法重写的应用场景和注意事项

方法重写的应用场景：当父类的方法不能完全满足子类使用的时候，既可以保留父类的功能（沿袭、传承），还可以有自己特有的功能

**方法重写的注意事项：**

\* 两同两小一大

两同：

方法名一定相同

参数列表一定要相同

两小：（**不是很懂**）

子类方法的返回类型小于等于父类方法的返回值类型

子类方法抛出的异常小于等于父类方法抛出的异常

一大

子类的方法的权限修饰符大于等于父类方法的权限修饰符

权限修饰包括四种：public>protected>默认>private

总结：方法重写时，一般方法签名（声明）保持和父类的方法签名（声明）一致，修改方法体中的内容即可

注解

@Override：方法重写，说明下面的方法是重写父类的方法

### 方法重写和重载辨析

方法重载：在同一个类中，方法名相同，参数列表不同，和返回值类型和权限修饰符都无关

方法重写：在两个类（子父类）中，方法名相同，参数列表相同，

子类权限修饰符大于等于父类方法的权限修饰符，子类返回值类型小于等于父类方法的返回值类型

### 继承中构造方法的执行顺序

\* 在子父类中，创建子类对象，调用子类的构造方法，

\* 在子类的构造方法的第一行代码如果没有调用父类的构造或者没有调用子类的其他构造（没有super或this），则默认调用父类无参构造

\* 为什么要调用父类构造？

\* 因为需要给父类的成员变量初始化

\* 肯定会先把父类的构造执行完毕，在去执行子类构造中的其他代码

代码： 结果为**“我是父类有参”-“我是子类有参”-“我是子类无参”**

**public** **class** ExtendsSequence {

**public** **static** **void** main(String[] args) {

Zi6 z = **new** Zi6();

}

}

**class** Die6 {

**public** Die6(){

System.***out***.println("我是父类无参构造");

}

**public** Die6(**int** i) {

System.***out***.println("我是父类有参构造");

}

}

**class** Zi6 **extends** Die6 {

**public** Zi6() {

**this**(1);

System.***out***.println("我是子类无参构造");

}

**public** Zi6(**int** i ) {

**super**(1);

System.***out***.println("我是子类有参构造");

}

}

### this和super的区别

this:当前对象的引用

调用子类的成员变量

调用子类的成员方法

在子类的构造方法第一行调用子类其他构造方法

super:子类对象的父类引用

调用父类的成员变量

调用父类的成员方法

在子类的构造方法第一行调用父类的构造方法

### 继承优缺点

A:优点

提高了代码的复用性

提高了代码的可维护性（修改某一共享功能，一改则全改）

使类与类之间产生了关联关系，形成继承体系，方便对类的研究

是多态的前提之一

B:缺点：

类的耦合性增强了

开发的原则：**高内聚低耦合**

内聚：就是自己完成某件事情的能力

耦合：类与类的关系

## 匿名对象

匿名对象即无名对象,直接使用new关键字来创建对象

**应用场景：**

当方法只调用一次的时候使用

当做方法的实参传递

代码：

**public** **class** AnonymousObject {

**public** **static** **void** main(String[] args) {

*method*(**new** Student());

*method*();

}

**public** **static** **void** method(Student s) {

System.***out***.println("method方法");

}

}

**class** Student {

String name;

**int** age;

**public** Student() {

}

}

## final关键字

final： 修饰符，可以用于修饰**类**、**成员方法**和**成员变量**

final所修饰的类：不能被继承，不能有子类

final所修饰的方法:不能被重写

final所修饰的变量：是不可以修改的，是常量

## 抽象类

### 存在原因

分析事物时，发现了共性内容，就出现向上抽取。会有这样一种特殊情况，就是方法功能声明相同，但方法功能主体不同。那么这时也可以抽取，但只抽取方法声明，不抽取方法主体。那么此方法就是一个抽象方法。

**规定一个格式，来规范一系列类的书写。**

### 抽象类的特点

抽象方法只能在抽象类里面（**有抽象方法的类一定是抽象类**）

抽象类和抽象方法必须被abstract修饰

抽象类不能创建对象（不能实例化）

抽象类中可以有非抽象的方法

抽象类和类的关系也是继承（**没有子类的抽象类没有意义**）

一个类继承了抽象类——要么重写所有的抽象方法，要么他自己是抽象类

### 抽象类的成员的特点

成员变量

可以有成员变量

可以有常量（final修饰的变量）

成员方法

可以有抽象方法

可以有非抽象方法

构造方法

可以有构造方法的，需要对抽象类的成员变量进行初始化

代码：

**public** **class** AbstractCompanyDemo {

**public** **static** **void** main(String[] args) {

Programmer p = **new** Programmer();

p.name = "吳彥祖";

p.work();

Manager m = **new** Manager();

m.name = "周傑倫";

m.work();

}

}

**abstract** **class** Employee {

String name;

String id;

String salary;

**public** **abstract** **void** work();

}

**class** Programmer **extends** Employee {

**public** **void** work() {

System.***out***.println(name+"寫代碼工作");

}

}

**class** Manager **extends** Employee {

**public** **void** work() {

System.***out***.println(name+"盯着程序员写代码工作");

}

}

### 抽象类的细节

*A:抽象类关键字abstract不可以和哪些关键字共存?（private, static, final）*（**关键在于抽象方法被重写才有意义**）

1.private：

私有的方法子类是无法继承到的，也不存在重写，而abstract和private一起使用修饰方法，abstract既要子类去实现这个方法，而private修饰子类根本无法得到父类这个方法。互相矛盾。

另外，用private修饰抽象类的构造方法，也会导致子类无法继承该抽象类

2.final:

抽象类不能和final共存,因为抽象类自身无法创建对象,我们需要通过子类创建对象,一旦抽象类使用final关键字,那么抽象类就没有子类

抽象方法不能和final共存,因为抽象方法后期需要被子类重写,一旦加final无法重写

3.static:

抽象方法不能和static关键字共存,因为一旦加static我们就可以通过类名直接访问抽象方法,由于**抽象方法没有方法体**,没有任何意义,也不允许这样做

*B:抽象类中是否可以不定义抽象方法?*

是可以的，那这个抽象类的存在到底有什么意义呢？不让该类创建对象,方法可以直接让子类去使用

*C:抽象类是否有构造函数?*

有,抽象类的构造函数,是由子类的super语句来调用,用于给抽象类中的成员初始化