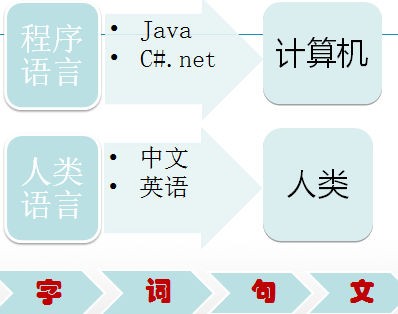
引言

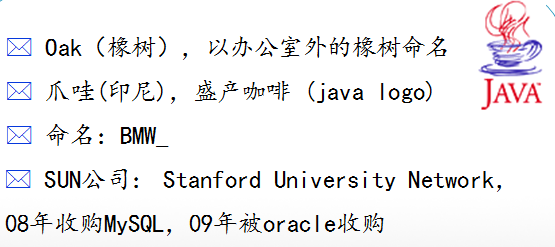




Java面向对象程序设计

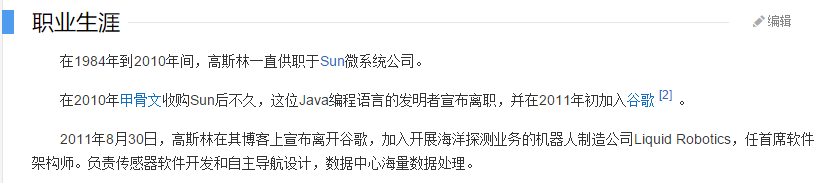
# Java开发入门

## 概述



### Java之父

 1955年



电脑前坐久了，要多喝水多运动

### 3、Java语言平台

**Java SE 标准版本**

Java ME移动平台版本——移动设备电子设备，如数字机顶盒等——Android

Java EE企业级版本——组件等——更复杂

叫法：

JDK1.8——JDK8

J2SE——Java SE

### 4、Java语言特点

由很多其他小特点，但最重要的2个特点是：

**1）开源**

软件的源代码发布到网上， 让大家都可以学习改进

Java开源 J2EE框架（Spring Struts Hibernate…）

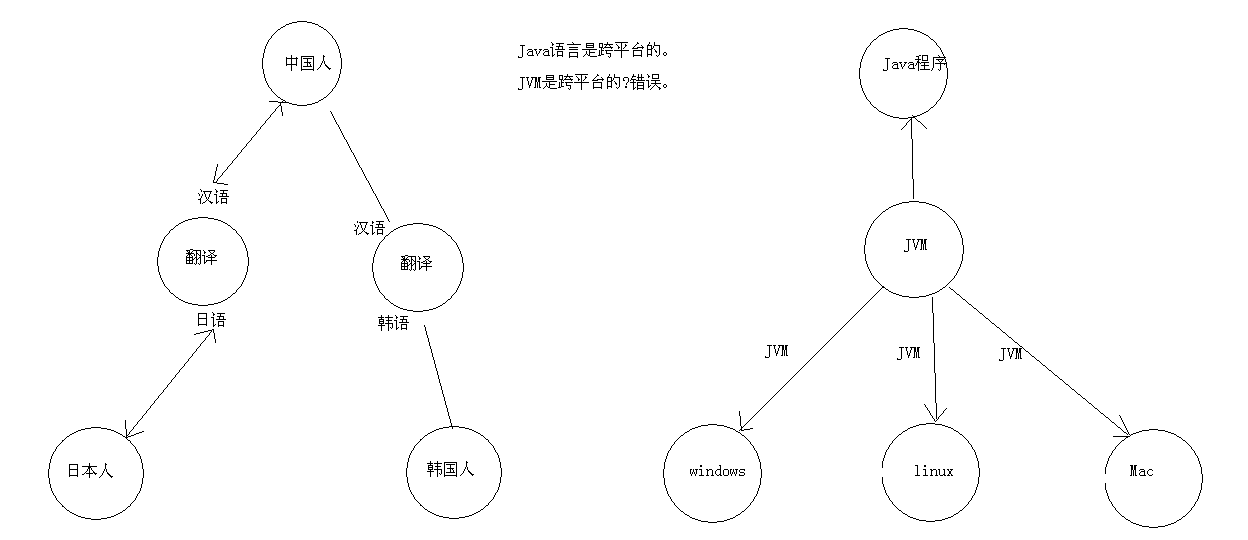
工具（开发环境）：Eclipse、Intellij idea

可扩展 插件（阿里巴巴代码规范、代码管理gitee码云…）

**2）跨平台**

**一次编译、处处运行**

因为有了JVM，所以Java是跨平台的；那么请问JVM跨平台吗，也就是不同平台的JVM相同吗？



### 5、JDK&JRE&JVM

作用：

JDK：Java程序的开发环境/工具（Java Development kit）

JRE：Java程序的运行环境（Java Runtime Environment）

JVM：保证Java语言跨平台

关系：

JDK：JRE+工具

JRE：JVM+类库

## 搭建开发环境

### 1、JDK

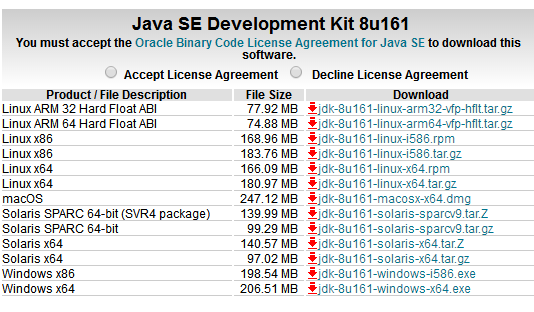
#### （1）版本



你会选用哪个版本？

试验品 NO 淘宝MySQL

#### （2）下载



官网下载，Windows 注意 32位还是64位

建议：所有跟开发相关的软件都不要安装在有中文或者空格的目录下。

查看是否安装成功Java： java –version

#### （3）配置环境变量

1）新建系统变量

JAVA\_HOME C:\Program Files\Java\jdk1.8.0\_60

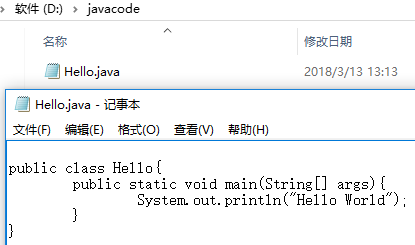
CLASSPATH .;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar;

2）修改系统变量

Path %JAVA\_HOME%\bin

#### 练习1-1 Hello World（cmd运行java程序）

1）编写java源程序

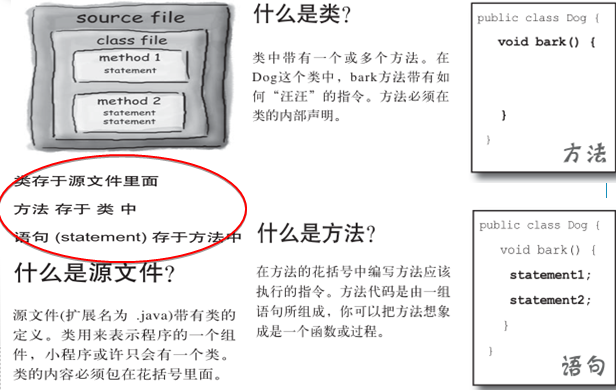


要求文件名称和类名一致。实际上不这样做也是可以的。但是，注意：

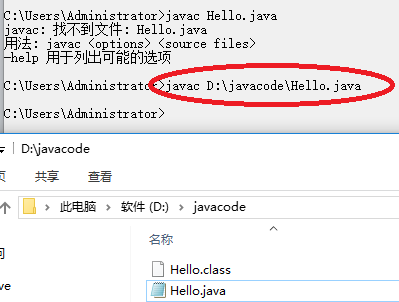
javac后面跟的是文件名+扩展名

java后面跟的类名不带扩展名

程序结构：



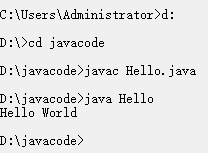
2）将源程序编译成字节码文件.class



**注意：**因为设置了环境变量，所以能自动找到javac.exe的位置，否则需要先到javac.exe所在目录才能执行成功。

**javac.exe是Java编译器工具**，可以将编写好的Java文件编译成Java字节码文件（可执行的Java程序）。

3）运行查看结果



**java.exe是Java运行工具**，会启动一个JVM进程，JVM相当于一个虚拟的操作系统，专门负责运行由Java编译器生成的字节码文件（.class文件）。

4）可能出错：

javac编译成功，但是用Java运行class文件出现“找不到或无法加载主类” 的问题。

javac后文件加后缀.java例如 javac main.java

java后不加后缀.class    例如 Java main

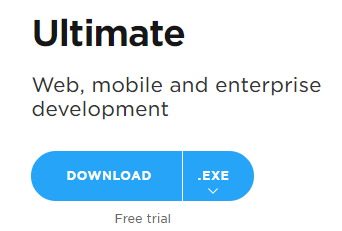
在classpath的最前面要加一个本目录路径，用.表示，由于后面还有java的dt和tools两个路径，所以.后要加;进行分隔表示。

例如：**.;**D:\Java\jdk1.8.0\_11\lib\tools.jar;D:\Java\jdk1.8.0\_11\lib\dt.jar

### 2、intellij idea

#### （1）下载安装

http://www.jetbrains.com/idea/download/#section=windows



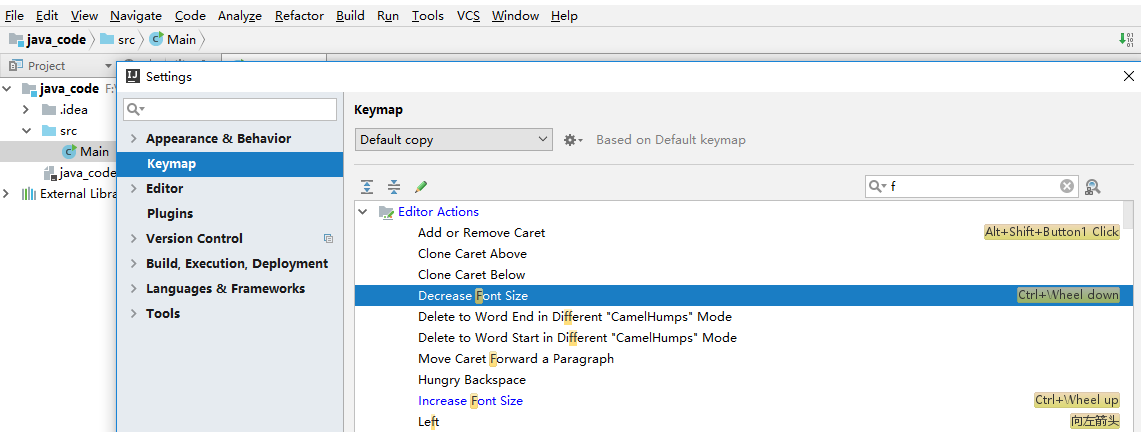
下一步安装，注册码。

打开时，关联java（C:\Program Files\Java\jdk1.8.0\_60）

#### （2）设置—字体大小

<https://www.cnblogs.com/zlslch/p/7770965.html>

改成用按住ctrl键鼠标滚动来更改字体大小



【Decrease Font Size】右键选择【Add Mouse Shortcut】，按住Ctrl键+鼠标向下滚动，【OK】即可。

【Increase Font Size】同理

#### （3）设置—背景色

默认是白色背景

File—Setting—Editor—Color Scheme—Scheme：Darcula 黑色背景

#### （4）设置—编码

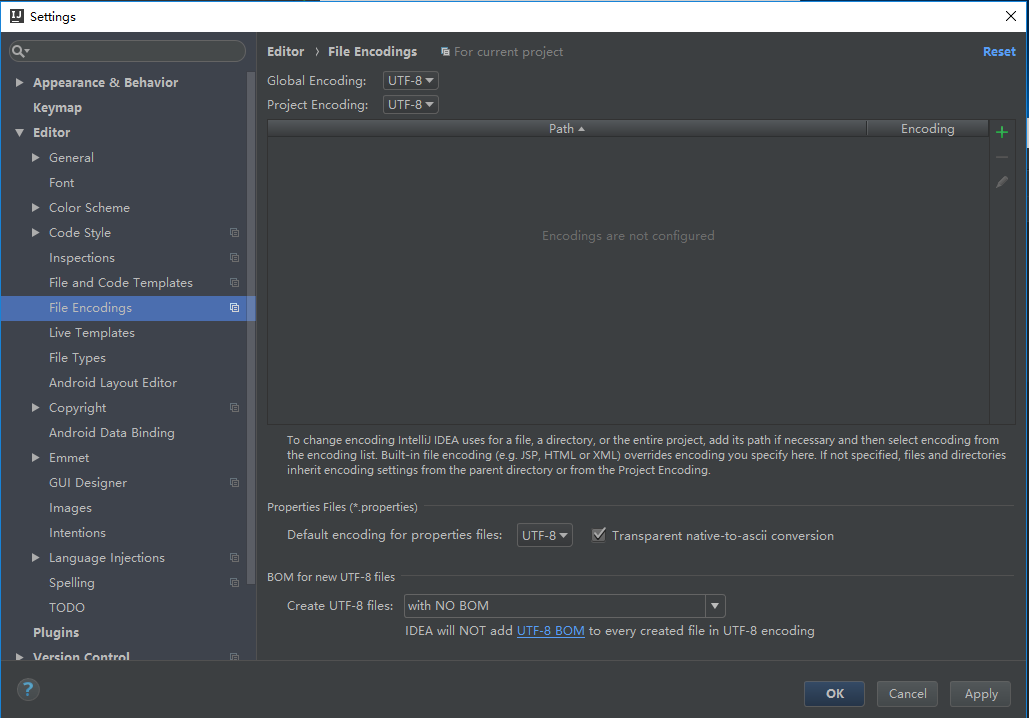
File->Settings->Editor->File Encodings：

Global Encoding:UTF-8

Projectt Encoding:UTF-8

Default encoding for properties files:UTF-8

勾选上Transparent native-to-ascii conversion



#### （5）快捷键

1）代码格式化

Ctrl+Alt+L

与QQ热键冲突

2）注释

单行注释： Ctrl+/

多行注释： Ctrl+Shift+/

文档注释： /\*\* 按回车键，后面自动会出来

3）代码提示

Alt+Enter

4）方法参数提示

Ctrl+P

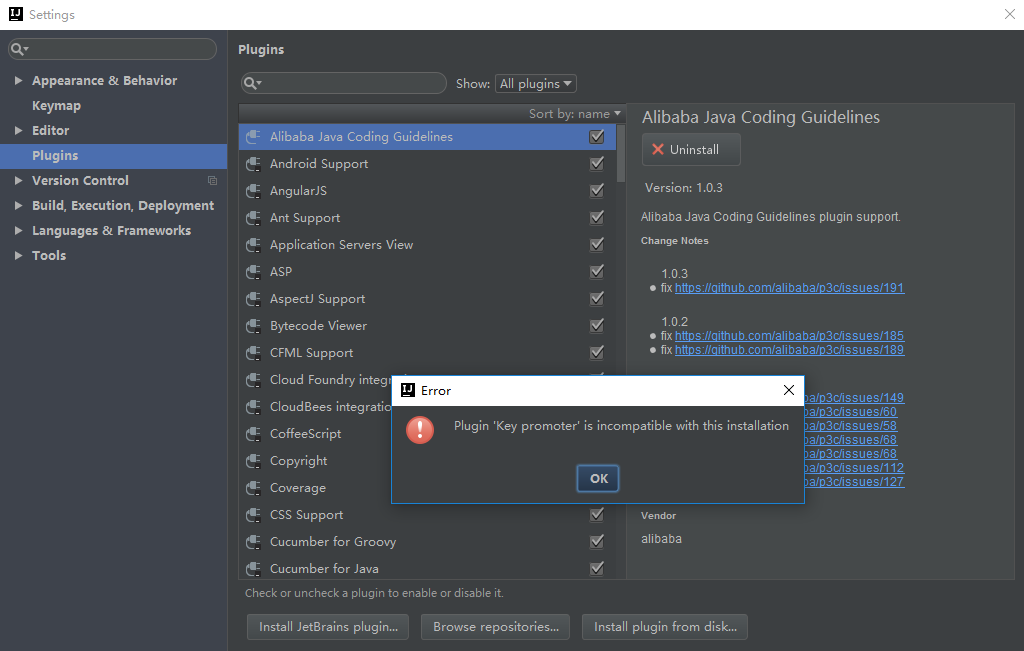
#### （6）插件

http://blog.csdn.net/RobertoHuang/article/details/75042116

1）快捷键提示插件

Key promoter是在你通过非快捷键方式使用某功能时 为你提供快捷键建议

Setting—Plugins—Browse Repositories—搜索 key promoter

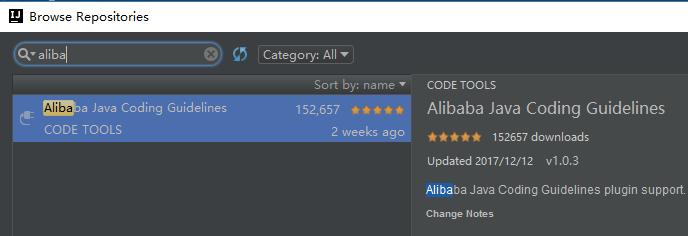


ideaIU-2017.3.2版本不兼容

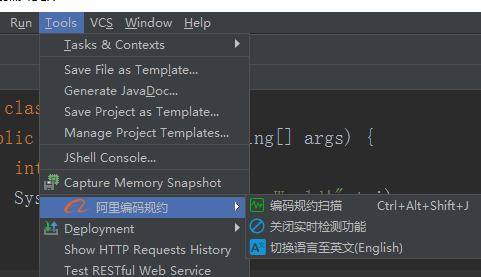
升级到2017.3.5版本可以

2）阿里代码规范

Setting—Plugins—Browse Repositories



安装完毕后：



3）代码管理gitee码云

**1、注册码云账号，创建项目**

<https://gitee.com/>

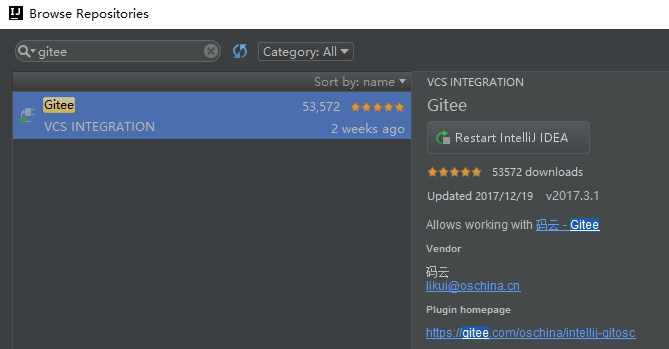
<https://www.cnblogs.com/xiaoxiaoccaiya/p/7125136.html>



**2、安装git**：Git-2.15.1.2-64-bit.exe

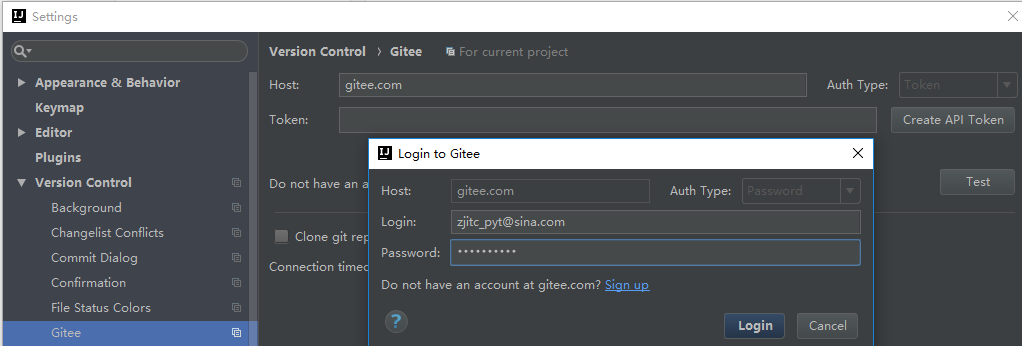
安装后的目录： C:\Program Files\Git\bin

**3、安装插件gitee**



**4、配置**

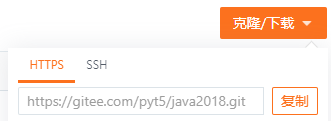
点击Setting—Version Control—Gitee，点击Create API Token，输入码云上注册的账号密码。Test测试通过



#### 练习1-2 Hello World（使用idea+Git）

**1、将远程git项目clone到本地，以后都在此项目基础上操作**

1）复制git地址

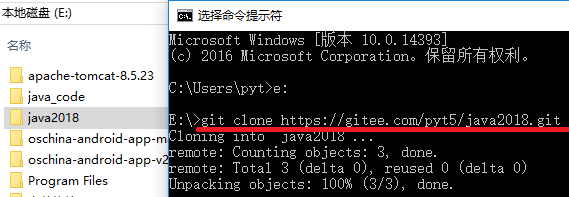


2）克隆到本地

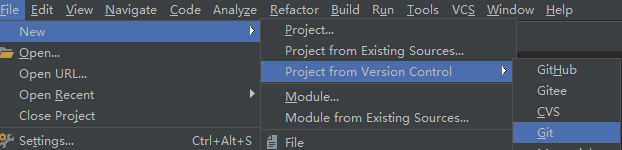
第一种方法：

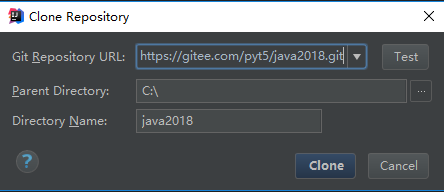
git clone <https://gitee.com/pyt5/java2018.git>

这样本地E盘就有了该项目



第二种方法：

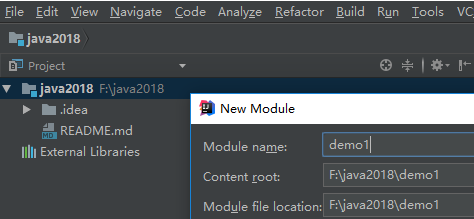




先Test拷贝过来的git项目URL，然后点击“Clone”

**2、新建HelloWorld**

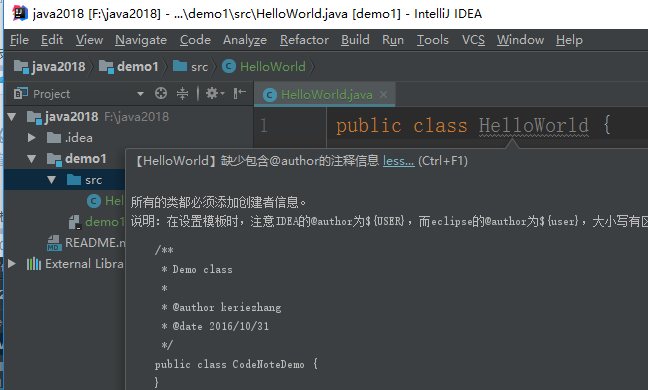
1）第一次从远程git 克隆下来的项目，可以新建module



2）新建类HelloWorld

New—Java Class：HelloWorld

加了阿里编码规约插件后，会自动提示：缺少注释信息



Alt+Enter代码提示：

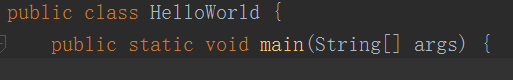


可以选择“添加/提取@author”

也可以选择“Add Javadoc”，添加文档注释

3）新建main方法

输入**psvm** 自动提示main方法



4）输出语句

输入**sout**可以快速生成 System.out.println();

System.*out*.println("HelloWorld");

5）运行查看结果

6）出错

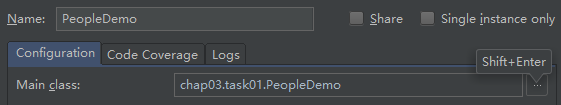
无效的源发行版：8

如果你安装的是JDK1.7，而在file->project structure中设置的是language level是8的话，就会出现这个错误提示

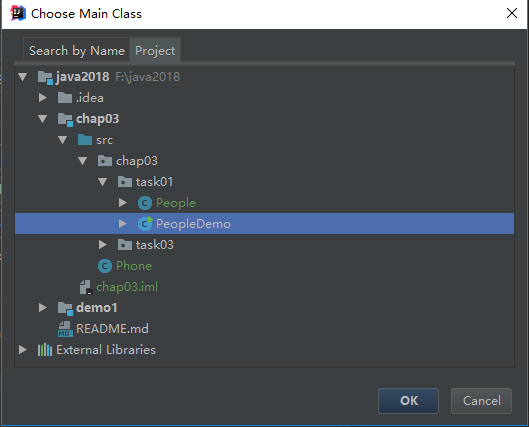
错误: 找不到或无法加载主类

会有红叉叉，点击右上角

Main class配置浏览选择正确的类就行：

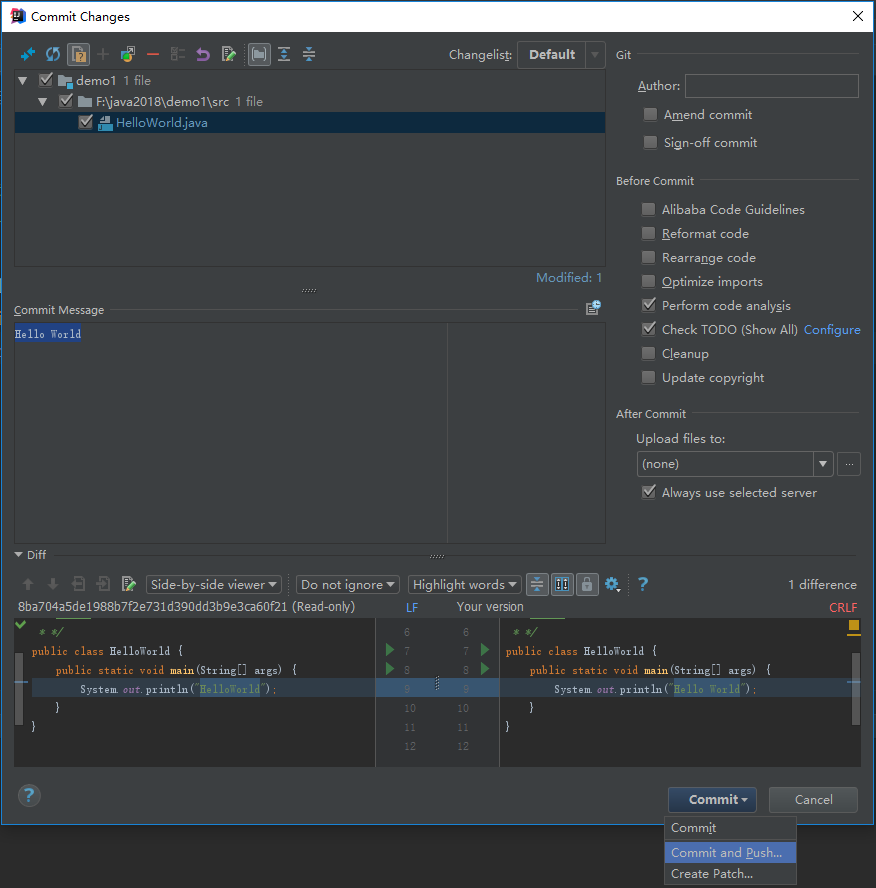


Project选项卡，选择正确的类：



**3、本地项目push到远程git**

项目右键—【Git】—先add 再commit。写上提交信息（修改的内容），然后点击commit and push，最后点击push即可。

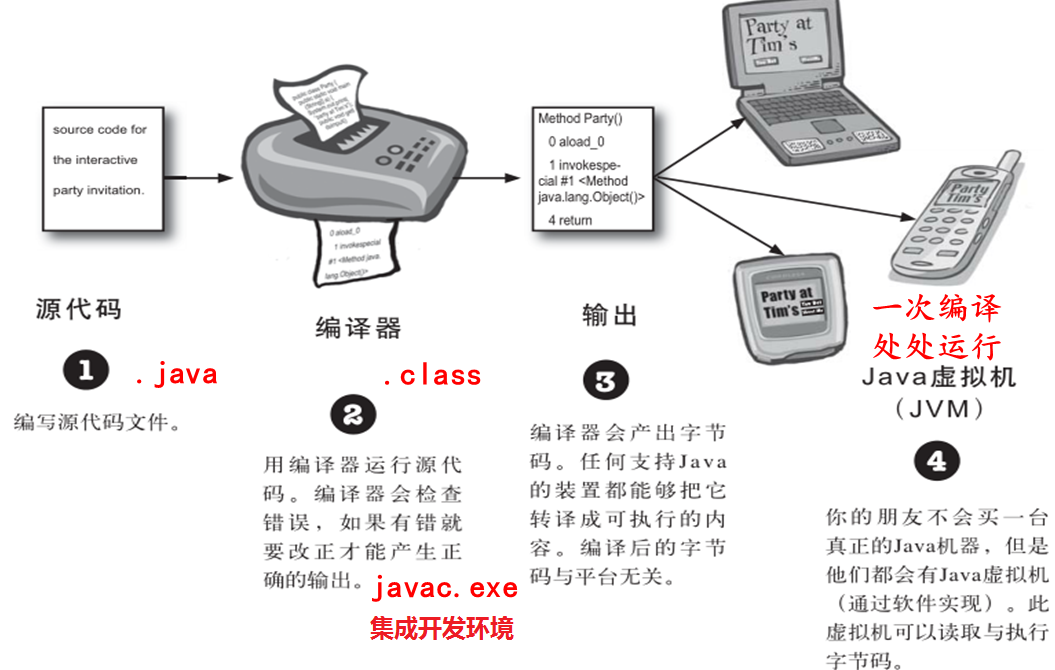


## 工作方式

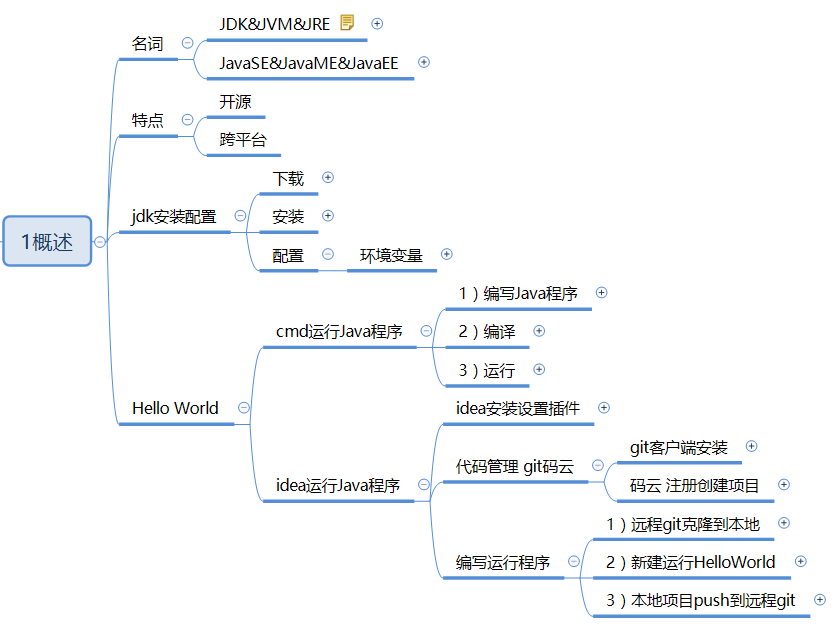
1、编写源文件 .java

2、将源文件编译成字节码文件 .class

3、有了JVM，.class字节码文件就可以运行在任何平台



## 小结



## 附加：类&方法注释模板

https://jingyan.baidu.com/article/6766299782849f54d51b848d.html

http://blog.csdn.net/u013412790/article/details/52807102

#### 1、类头注释

打开file->setting->Editor->File and Code Templates->Includes->File Header

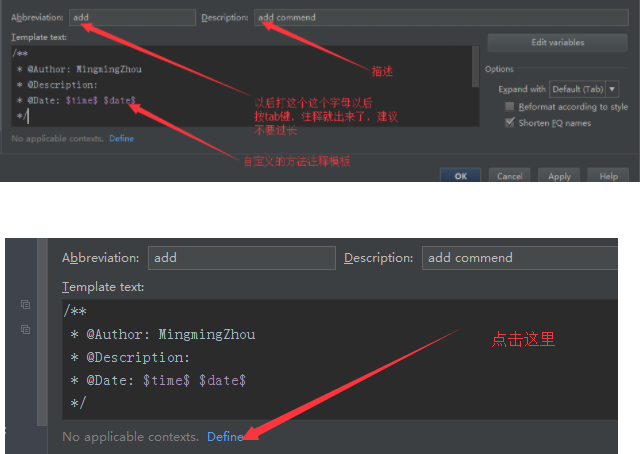
加上下面这个注释模板，以后新建类 都会在类头自动加上文档注释。

/\*\*  
 \* @author: pyt  
 \* @date: **${**DATE**} ${**TIME**}** \* @description：   
 \*/

#### 2、方法注释

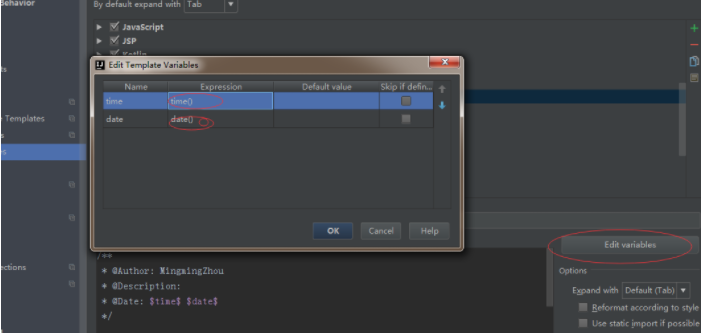
打开file->setting->Editor->LiveTemplates点击右边上面那个绿色的+号，选择Template Group双击，然后弹出一个窗口，随便添加一个名字，我这里添加的是MyGroup然后点击OK

点击一下刚刚你添加的那个名字（我这是MyGroup），然后点击右边上面那个绿色的+号，选择LiveTemplate双击，填写模板：



点击Define，选择Everywhere

最后点击右下角的Edit variables 按钮，然后弹出一个窗口



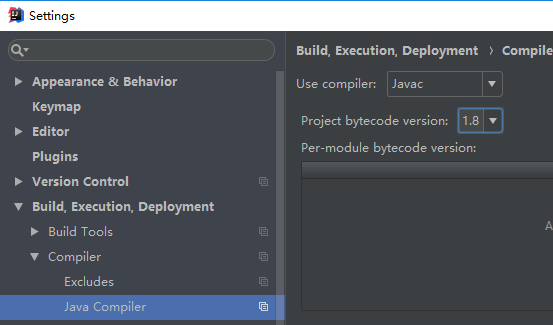
**写方法之前 写上add（刚刚填的）按tab，注释就出来了**

## 错误更改

### 1、无法运行helloworld

JDK版本保持一致

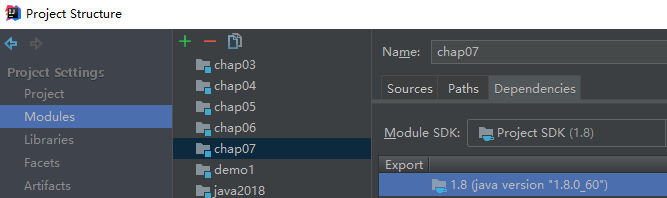
1）File——Setting——Java Compiler



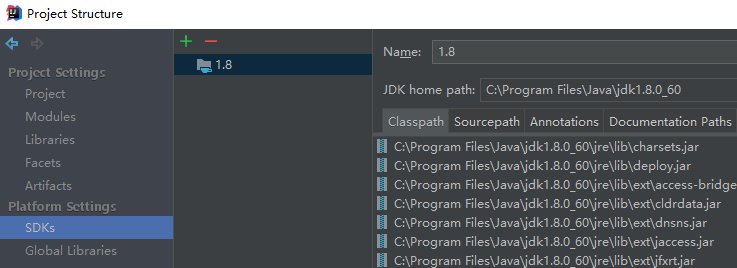
2）File——Project Structure——Project



3）Dependencies

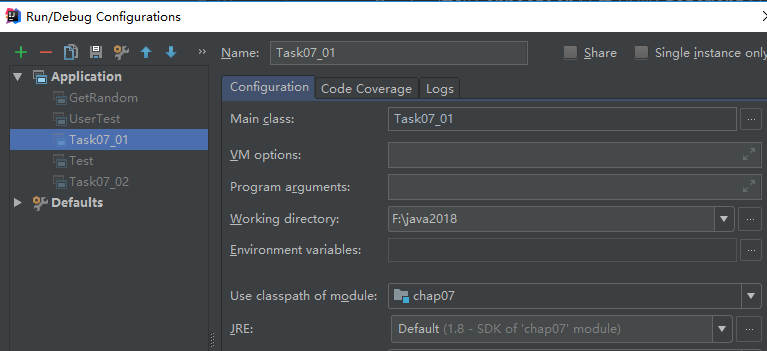


4）SDK



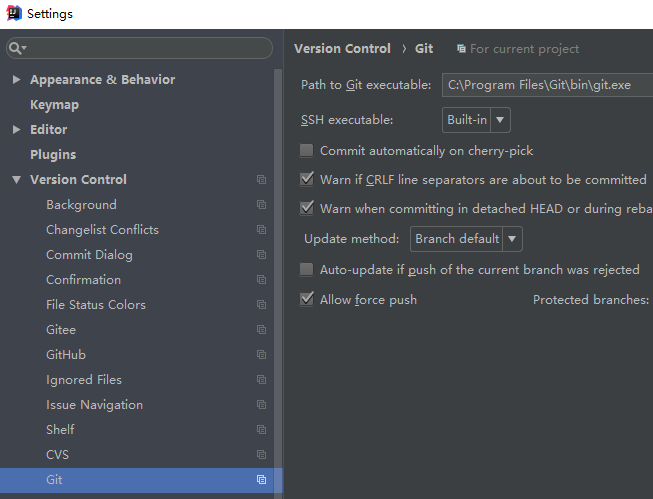
5）Edit Configuration



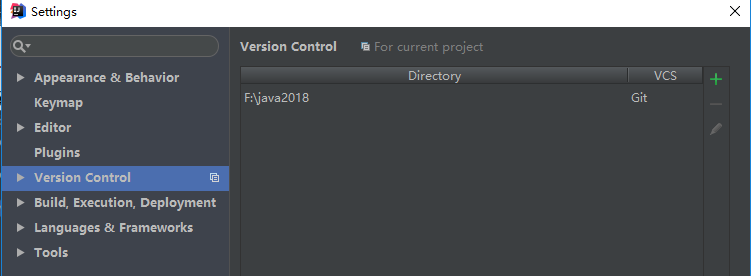


### 2、无法git

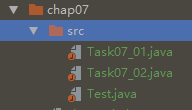
1）Git路径



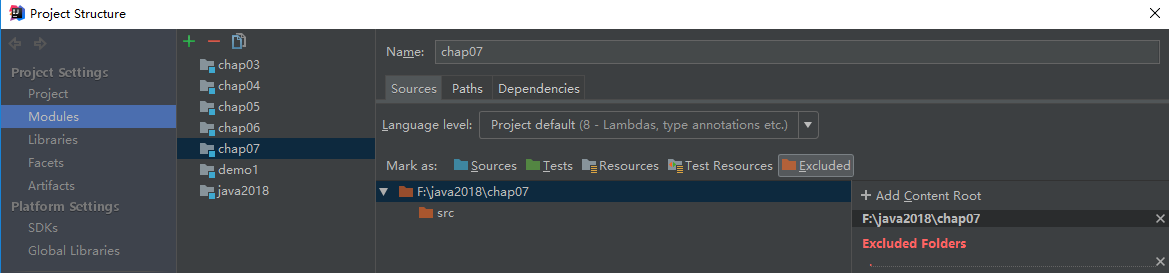
2）没有关联git仓库



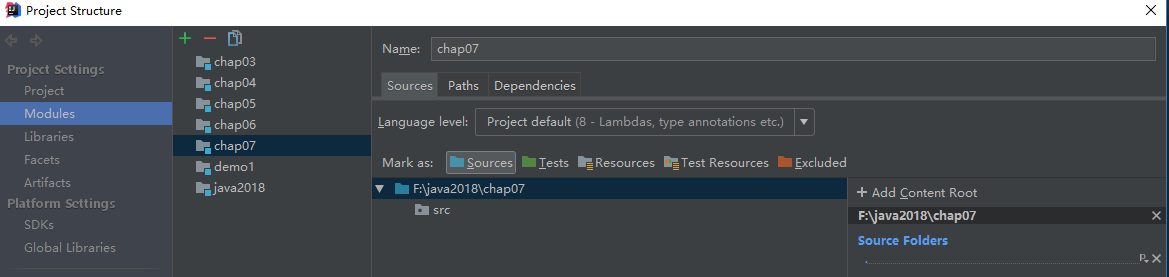
### 3、类C图标变J图标后无法运行



1）x掉Excluded…红色的



2）点击Sources 增加



# Java编程基础

## 常量&变量

### 1、关键字

被Java语言赋予特定含义的单词。

类似于Notepad++这样的高级记事本会对关键字有特殊颜色标记

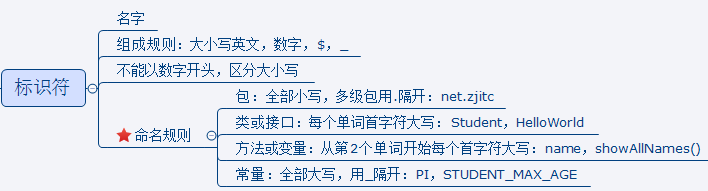
特点：

全部小写。(取变量名的时候不能使用关键字 不能int class)



### 2、标识符

就是给类，接口，方法，变量等起名字的字符序列



### 3、注释

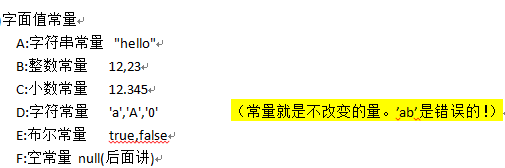


快捷键ctrl+/ shift+ctrl+/

将思想先用注释表达出来，再用代码去体现

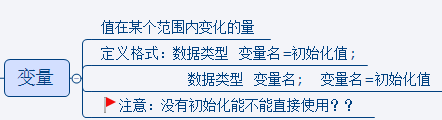
### 4、常量

在程序执行的过程中，其值不发生改变的量



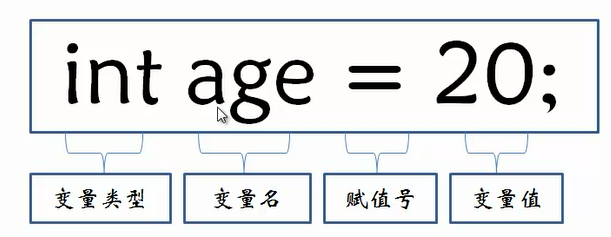
### 5、变量

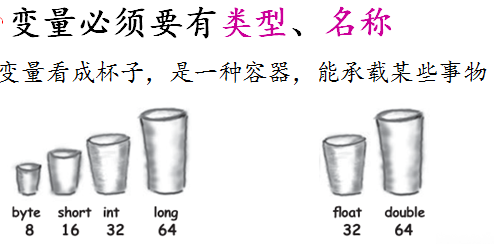
#### （1）定义



变量由3部分组成：数据类型、变量名、值

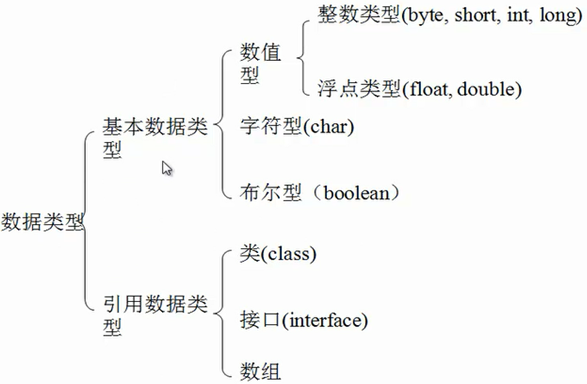
Eg： int age=20; String name=”pyt”;

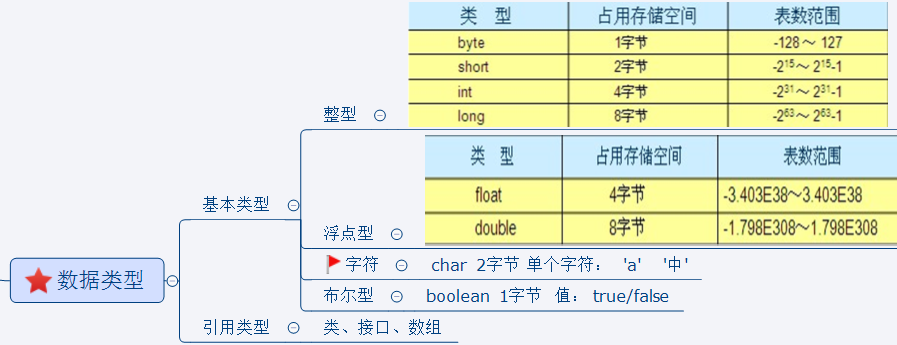




#### （2）数据类型



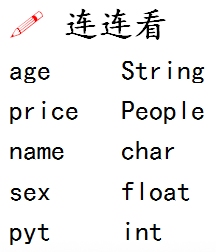




注意：

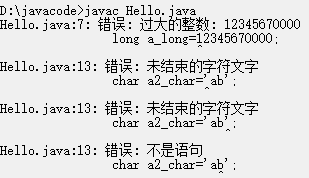
整数默认是int类型，浮点数默认是double。

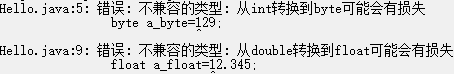
长整数要加L或者l；单精度的浮点数要加F或者f。



#### 练习2-1 基本数据类型







为什么错误？怎么修改？



#### （3）数据类型转换

byte x=3; int y=4; z=x+y; z应该定义成什么数据类型？

**1）默认转换**

byte,short,char 🡪 int 🡪 long 🡪 float 🡪 double

**“从小到大”转换**

so: 3/2 结果是1 3/2.0结果是1.5

注意：

boolean类型不参与转换。

byte,short,char之间不相互转换，直接转成int比较类型参与运算。

**2）强制转换**

格式：

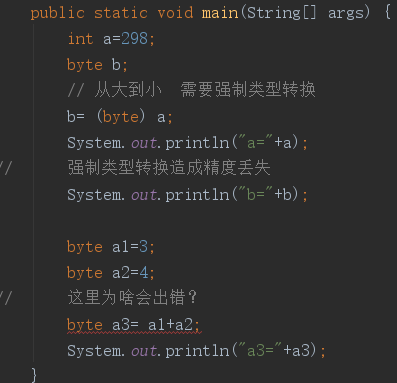
目标数据类型 变量名 = (目标数据类型) (被转换的数据);

int a=(int) b;

**“从大到小”转换**

强制类型转换可能会有精度损失

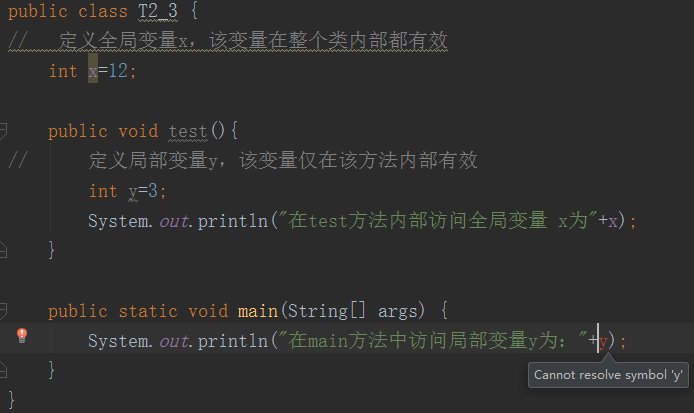
#### 练习2-2 类型转换



表达式类型自动提升：变量在表达式中进行运算时，也有可能发生自动类型转换。

#### （4）作用域

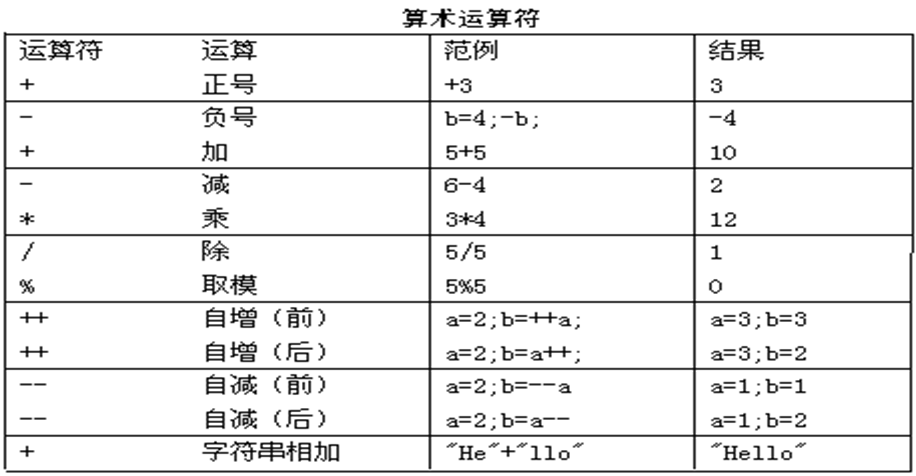
变量需要先定义后使用。并且变量需要在它的作用范围内才可以被使用，这个作用范围被称为变量的作用域。



## 运算符

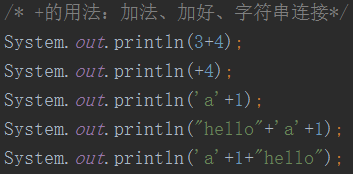
### 1、算术运算符

+ - \* / % ++ --



注意：

1）+ 的用法



注意：字符与字符串、ASCII码

2）++ -- 容易搞错，建议直接用x=x+1 不用x++。 但是别人写的代码要能读懂

3）/ %

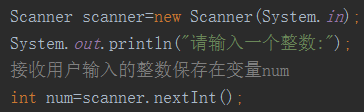
#### 练习2-3 算术运算符

如何判断一个数是不是偶数？

如何判断一个数能不能被7整除？

如何将秒转成时间，例如将156秒转成2分36秒

提示：



### 2、赋值运算符

基本赋值运算符：= （= 与 ==的区别）

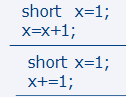
扩展的赋值运算符：+= -= \*= /= %=

x+=1 等价于 x=x+1

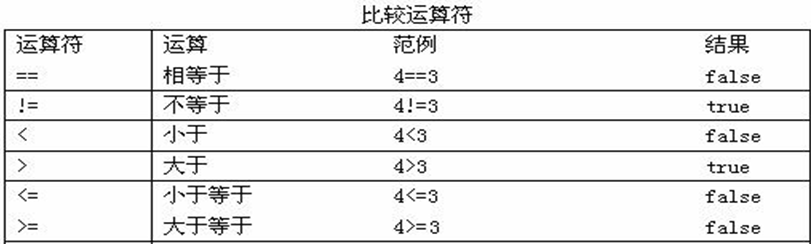
但也还是有区别：扩展的赋值运算符隐含了自动类型转换

#### 练习2-4 赋值运算符

以下2种方式都对吗？有区别吗？



### 3、比较运算符



最终结果都是Boolean

注意：=与==的区别

猜结果：

int x=3,y=4;  
System.*out*.println(x==y);  
System.*out*.println(x=y);

### 4、逻辑运算符

用于连接boolean类型的式子



结论：

&：有false则false

|：有true则true

^：相同则false，不同则true。情侣关系。

!：非true则false，非false则true

&&：结果和&是一样的，只不过有短路效果。左边是false，右边不执行。

||：结果和|是一样的，只不过有短路效果。左边是true，右边不执行。

#### 练习2-5 逻辑运算符

如何判断一个数是否比3大，比6小，类似小学数学的： 6>x>3

如何判断一个数既能被3整除又能被7整除

### 5、三元运算符

格式

比较表达式?表达式1:表达式2;

执行流程：

首先计算比较表达式的值，看是true还是false。

如果是true，表达式1就是结果。

如果是false，表达式2就是结果。

#### 练习2-6 键盘输入和三元运算符

请用三元运算符表示：



**没必要记运算符的优先级，用()来实现想要的运算顺序。**

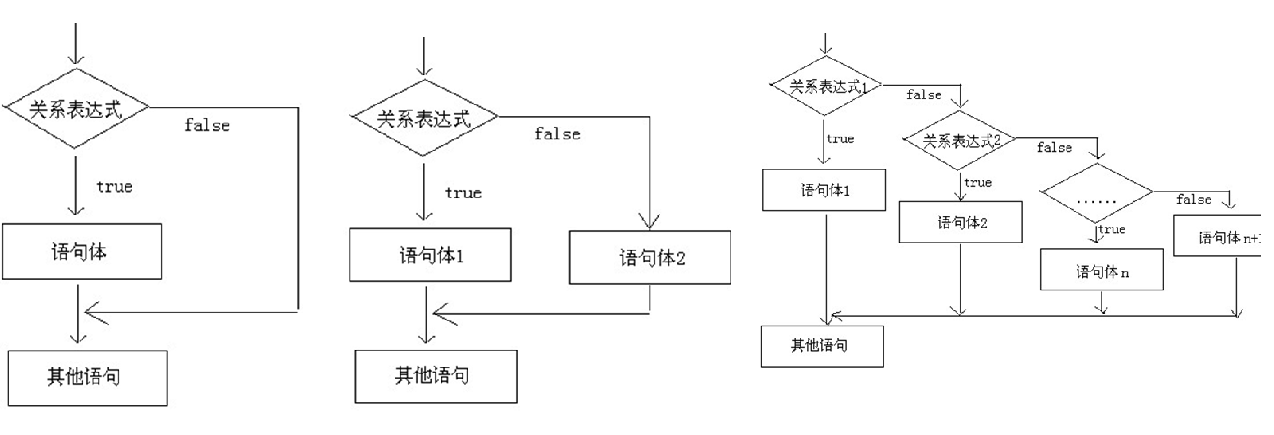
## 语句

顺序结构：从上往下，依次执行

选择结构：按照不同的选择，执行不同的代码

循环结构：做一些重复的代码

### 1、if语句



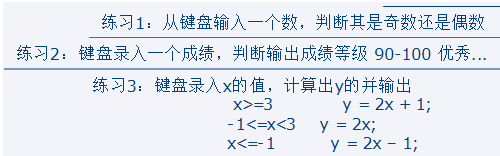
If语句：如果满足某种条件，就进行某种处理；

If..else语句：如果满足某种条件，就进行某种处理，否则就进行另一种处理。

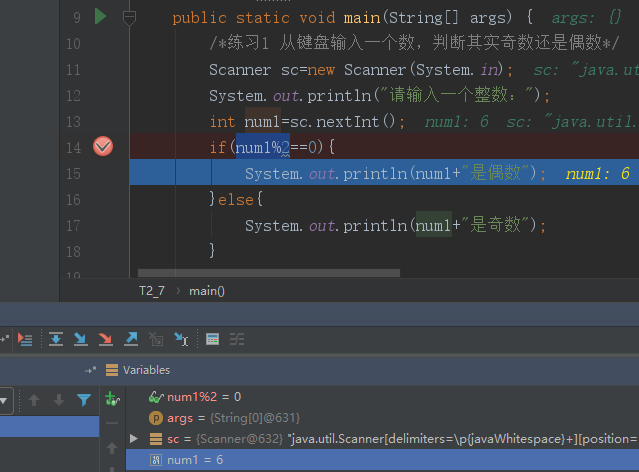
类似三元运算符

If..else if..else语句：用于对多个条件进行判断，进行多种不同的处理。

#### 练习2-7 if语句练习

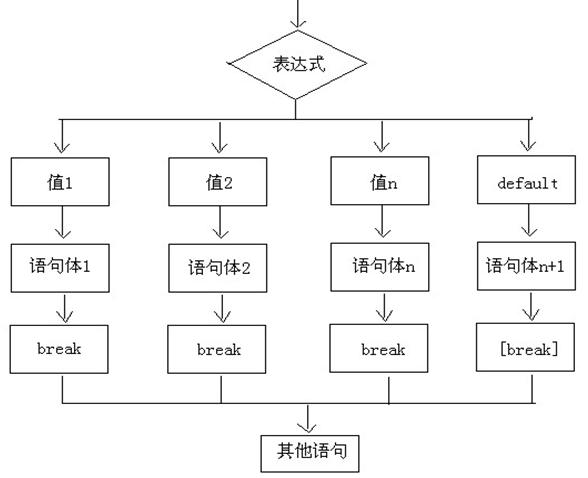


Debug 单步调试 查看过程



加断点，调试，单步执行，加观察量（watch）

### 2、switch语句



switch(表达式) {

case 值1:

语句体1;

break;

case 值2:

语句体2;

break;

...

default:

语句体n+1;

break;

}

**格式解释说明：**

switch:说明这是switch语句。

表达式:可以是byte,short,int,char

JDK5以后可以是枚举

JDK7以后可以是字符串

case:后面的值就是要和表达式进行比较的值

break:表示程序到这里中断，跳出switch语句

default:如果所有的情况都不匹配,就执行这里,相当于if语句中的else

**执行流程:**

首先计算表达式的值

和每一个case进行匹配，如果有就执行对应的语句体，看到break就结束。

如果没有匹配，就执行default的语句体n+1。

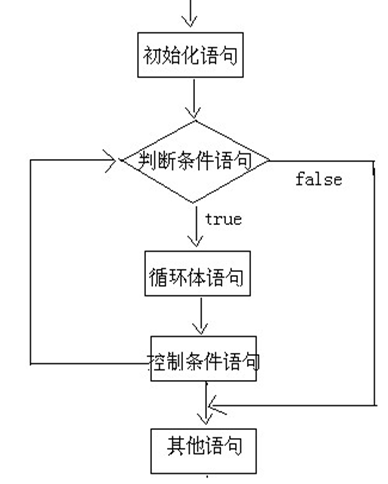


#### 练习2-8 switch语句练习

练习1：输出当前的月份值以及对应的季节

提示：/\* 使用Calendar类\*/  
 Calendar cal=Calendar.*getInstance*();  
 int month=cal.get(Calendar.*MONTH*);  
// 当前月份是month+1（月份从0开始）

### 3、while语句



基本格式：

while(判断条件语句) {

循环体语句;

}

扩展格式：

初始化语句;

while(判断条件语句){

循环体语句;

控制条件语句;

}

#### 练习2-9 while语句练习

练习1：输出1至10的偶数和

练习2：任意输入一整数，求它的位数

练习3：求水仙花数

求100至999之间的所有水仙花数。

例如：153=13+53+33

练习4：我国最高山峰是珠穆朗玛峰：8848m，我现在有一张足够大的纸张，厚度为：0.01m。请问，我折叠多少次，就可以保证厚度不低于珠穆朗玛峰的高度?

### 4、do while语句

基本格式

do {

循环体语句;

}while(判断条件语句);

扩展格式：

初始化语句;

do {

循环体语句;

控制条件语句;

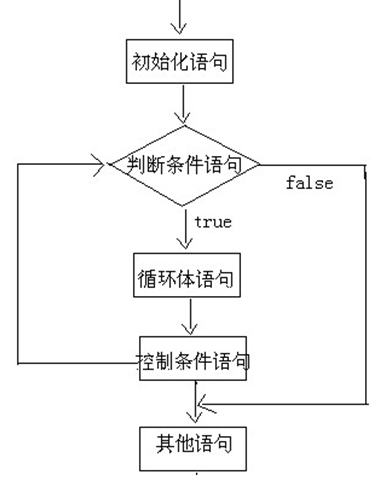
}while(判断条件语句);

区别：

do...while循环至少执行一次循环体

while必须先判断条件是否是true，然后才能决定是否执行循环体

### 5、for语句



格式：

for(初始化语句;判断条件语句;控制条件语句){

循环体语句;

}

执行流程：

1）执行初始化语句

2）执行判断条件语句

如果这里是true，就继续

如果这里是false，循环就结束

3）执行循环体语句

4）执行控制条件语句

5）回到2

**while与for的区别：**

for适合于一个范围的判断

while适合次数不明确的

#### 练习2-10 for语句练习

练习1：统计1-1000之间同时满足如下条件的数据个数：

对3整除余2

对5整除余3

对7整除余2

练习2：求阶乘

例如：5！=5\*4\*3\*2\*1

练习3：求水仙花数

求100至999之间的所有水仙花数。

例如：153=13+53+33

练习4：输入一整数，反转输出这个数的每一个数字。

Eg：输入1234，输出4321.

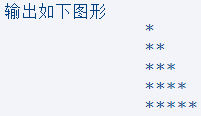
提示：如果该数是个4位数，个位数为4，则反转后最高位为：4\*10的3次方，Math.pow(10,3) 表示10的3次方。

### 6、循环嵌套

外循环控制行，内循环控制列

#### 练习2-11循环嵌套练习

练习1：



找规律：（显示的列 与 行数的规律）

第1行：1列

第2行：2

第3行：3

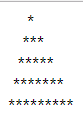
…

练习2：打印金字塔

1）接收用户输入的金字塔层数n

2）每层空格数= ？（与行的关系是什么？ i）

3）每层星星数=？ （与行的关系是什么？ i）



找规律：

行数 空格数 星星数

第1行 4 1

2 3 3

3 2 5

4 1 7

…

i n-i ?

行数i 通过循环实现每次递增1，每次循环要输出多少个空格，输出多少个星星。

比如第一次外循环（第1行）中：循环（内循环）输出n-i个空格

### 7、控制跳转语句

#### 1）break

break：中断的意思

用在循环和switch语句中，离开此应用场景无意义。

作用： 跳出单层循环（跳出当前循环）

#### 2）continue

continue：继续

用在循环中，离开此应用场景无意义。

作用：跳出单层循环的一次，可以继续下一次

#### 3）return

return：返回

用于结束方法。

一旦遇到return，程序就不会在继续往后执行。后面介绍。

#### 练习2-12 跳转语句练习

练习1：代码填空

for(int x=1; x<=10; x++) {

if(x%3 == 0) {

//补齐代码

}

System.out.println("Java基础班");

}

如何让控制台输出2次：Java基础班

如何让控制台输出7次：Java基础班

如何让控制台输出13次：Java基础班

练习2：

小芳的妈妈每天给她2.5元钱，她都会存起来，但是，每当这一天是存钱的第5天或者5的倍数的话，她都会花去6元钱，请问，经过多少天，小芳才可以存到100元钱。

### 练习2-13 BMI指数

1、接收用户输入身高体重（假设用户输入正确，身高单位cm,体重单位kg）

2、根据体质指数计算出结果，并保留2位小数：

体质指数（BMI）=体重（kg）÷身高^2（m）

EX：70kg÷（1.75×1.75）=22.86

3、根据结果给出用户提示信息：

成人的BMI数值：

过轻：低于18.5

正常：18.5-23.9

过重：24-27

肥胖：28-32

非常肥胖, 高于32

例如：当输入身高175，体重70时，提示：



提示：保留2位小数：

float d = 756.2345566f;

//方法一：调用DecimalFormat类

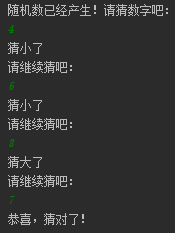
DecimalFormat df = new DecimalFormat(".00");

System.out.println(df.format(d));

//方法二：通过String类的format函数实现

System.out.println(String.format("%.2f", d));

### 练习2-14 猜数字游戏



提示：

随机数：Math.random()方法可以产生[0.0,1.0]区间内的一个双精度浮点数

如何得到0到10之间的随机数？

## 方法

### 1、为什么要用方法

打印一个宽为5个\*，高为3个\*的矩形

打印一个宽为4个\*，高为2个\*的矩形

打印一个宽为10个\*，高为6个\*的矩形

…

方法是一段可以**重复调用**的代码，完成**特定功能**的代码块。

方法==函数

### 2、方法的格式

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...) {

方法体语句;

return 返回值;

}

public static void main(String[] args) {

修饰符：目前就用 public static。后面再详细讲解其他修饰符

返回值类型：就是功能结果的数据类型

方法名：就是起了一个名字，方便我们调用该方法。

参数类型：就是参数的数据类型

参数名：就是变量

参数分类：

实参：实际参与运算的数据

形参：方法上定义的，用于接收实际参数的变量

方法体语句：就是完成功能的代码块

return：结束方法

返回值：就是功能的结果，由return带给调用者。

### 练习2-15 方法的定义与调用

**1）有明确返回值的方法**

**赋值调用**

**2）void修饰的方法（无返回值）**

**单独调用**

以下用方法来实现：

练习1：比较两个数的最大值

练习2：求m到n的之间的偶数和

练习3：输出m行n列的星形

### 3、方法重载

在同一个类中，方法名相同，参数列表不同。与返回值无关。

参数列表不同：

1）参数的个数不同。

2）参数的对应的数据类型不同。

#### 练习2-16 方法重载案例

实现数字求和案例：实现2个整数相加，3个整数相加，2个小数相加

add1() add2() add3()——> add() 重载

## 数组

### 1、数组定义与初始化

存储同一种数据类型的多个元素的容器

在数组中可以存放任意类型的元素，但同一个数组中存放的元素类型必须一致。

Java中的数组必须先初始化,然后才能使用。

所谓初始化：就是为数组中的数组元素分配内存空间，并为每个数组元素赋值。

1）动态初始化数组：（只给长度，系统给默认值）

int[] x=new int[100];

初始化时，只定义数组长度，由系统为数组分配初始值。

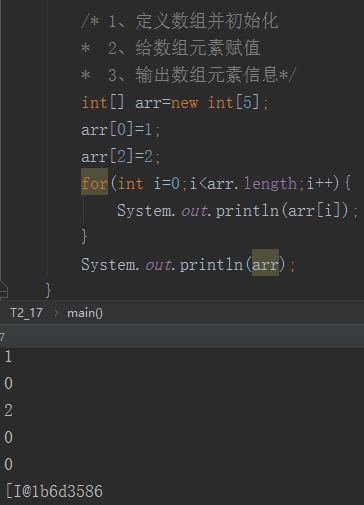
在内存中定义了100个int类型的变量，第一个变量的名称为x[0]，第100个变量的名称为x[99]，这些变量的初始值都是0。

2）静态初始化数组：（给出值，系统决定长度）

int[] y=new int[]{1,2,3};  
int[] y2={1,2,3,4}; // 简便

初始化时指定每个数组元素的初始值，由系统决定数组长度。

#### 练习2-17 数组的定义与访问



arr[i]输出元素值，arr输出数组地址？？

### 2、内存分配

Java 程序在运行时，需要在内存中的分配空间。为了提高运算效率，就对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存管理方式：

1）栈

存储局部变量。（局部变量：在方法定义中或者方法声明上定义的变量）

**数据使用完毕，就消失。**

2）堆

存储所有new出来的。

特点：**每一个new出来的东西都有地址**

**每一个变量都有默认值**

byte,short,int,long 0

float,double 0.0

char '\u0000'

boolean false

引用类型 null

**数据使用完毕后，在垃圾回收器空闲的时候回收。**

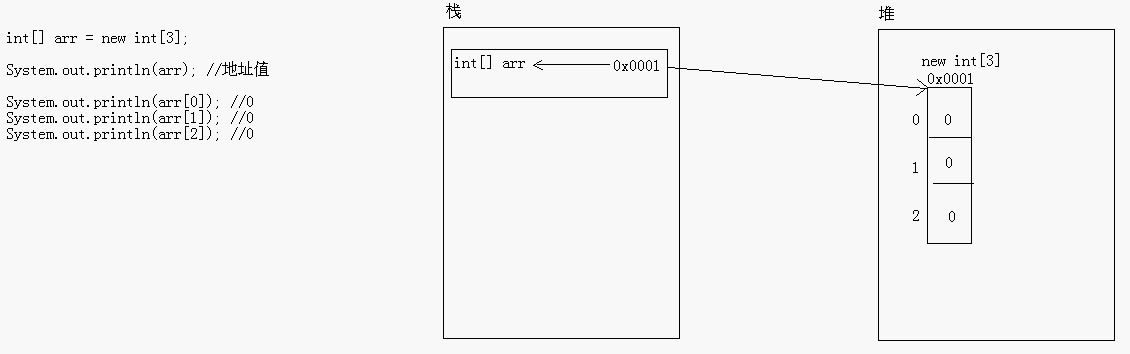
3）方法区

(面向对象部分详细讲解)

4）本地方法区(系统相关)

5）寄存器(CPU使用)

**视频：05.16\_Java语言基础(Java中的内存分配以及栈和堆的区别.avi**



指向：类似C语言中的指针

#### 练习2-18 数组内存分配

在main()方法中：

创建一个一维数组x，其元素分别4、3、7；

创建一个长度为4的一维数组y；

给y[0]赋值为1， y[1]赋值为2， y[2]赋值为x[2]；

输出数组y的所有元素；

执行y=x; 输出数组y的所有元素。

x[0]=10; 输出y[0]的值。

问题：

1、数组y的长度为4，当为y赋值时，y的长度还是4；

执行y=x后，y的长度为什么变成3了？？

2、为什么改变x[0]的值，y[0]的值会跟着一起变？？

### 3、常见操作

1）遍历

数组的遍历：依次访问数据中每个元素

2）最值

获取数组元素的最大值或最小值

3）排序

对数组元素进行从大到小或从小到大排序

#### 练习2-19 数组操作

练习1：创建一个长度为10的Int 数组，依次给数组每个元素赋值为0、2、4、6、8…，输出数组元素的值以及输出数组地址

练习2：定义一个有这些数的数组：{20, 10, 45, 24, 78, 53, 100, 60}。输出该数组的最大值和最小值。

练习3：将练习2中的数组按照从大到小的顺序排序后输出。

注意：1）数组索引越界

ArrayIndexOutOfBoundsException

访问到了数组中的不存在的索引时发生。

2）空指针异常

NullPointerException

数组引用没有指向实体，却在操作实体中的元素时。

### 4、多维数组

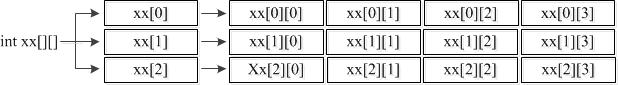
二维数组其实就是一个元素为一维数组的数组。

#### 1）定义

方法一：

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

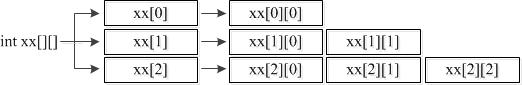
定义一个3行4列的二维数组，  
 表示：这个二维数组中有3个一维数组：xx[0]、xx[1]、xx[2]  
 每个一维数组有4个元素：xx[0][0]、xx[0][1]...



方法二：

int[][] xx=new int[3][];  
xx[0]=new int[1];  
xx[1]=new int[2];  
xx[2]=new int[3];

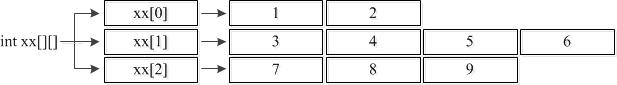
创建一个有3个一维数组的二维数组，每个一维数组的元素个数不确定或不同，可以动态定义每个一维数组的长度



方法三：

int[][] xx={{1,2},{3,4,5,6},{7,8,9}};

定义的二维数组有三个元素，这三个元素都是数组，分别是{1,2}、{3,4,5,6}、{7,8,9}

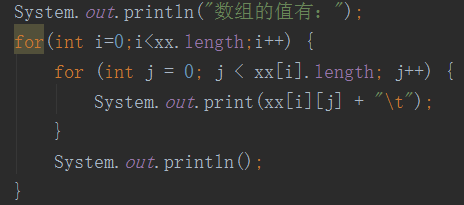


#### 2）遍历

双重循环

xx.length：一维数组的长度

xx[0].length：第一个一维数组中元素的个数。



#### 练习2-20 多维数组练习

求公司每个季度销售额以及年销售总额

某公司按照季度和月份统计的数据如下：单位(万元)

第一季度：22，66，44

第二季度：77，33，88

第三季度：25，45，65

第四季度：11，66，99

### 练习2-21 随机点名器

任务描述：1）存储全班同学姓名；

2）查看全部同学姓名；

3）随机点取其中一人姓名。

# 面向对象

## 思想

### 1、面向过程 VS 面向对象

面向过程开发，其实就是面向着具体的每一个步骤和过程，把每一个步骤和过程完成，然后由这些功能方法相互调用，完成需求。（**做什么，怎么做**）

面向对象：万物皆对象，

不断的创建对象，使用对象，指挥对象做事情

我们从执行者变成了指挥者（**强调对象，由对象去调用功能**）

举例：

1）买电脑：

面向过程：我的了解电脑--了解我自己的需求--找对应的参数信息--去中关村买电脑--讨价还价--买回电脑

面向对象：我知道我要买电脑 -- 班长去给我买 -- 班长就买回来了

2）洗衣服：

面向过程：把衣服脱下--找一个盆--放点洗衣粉--加点水--把衣服扔进去--搓一搓--清洗衣服--拧干--晾起来

面向对象：把衣服脱下--打开全自动洗衣机--扔进去--一键即可--晾起来

3）吃饭：

面向过程：去超市买菜--摘菜--洗菜--切菜--炒菜--盛起来--吃

面向对象：上饭店吃饭，你--服务员(点菜)--厨师(做菜)--服务员(端菜)--吃

### 2、案例：把大象装进冰箱

我们怎么才能更符合面向对象思想呢?

**1）有哪些类呢?**

**2）每个类有哪些东西呢?**

**3）类与类直接的关系是什么呢?**

把大象装进冰箱的分析? (如何分析有哪些类呢? 名词提取法。)

1）有哪些类呢?

大象

冰箱

Demo

2）每个类有哪些东西呢?

大象：进去

冰箱：开门、关门

Demo： main方法

3）类与类直接的关系是什么呢?

Demo中使用大象和冰箱类的功能。

代码体现：

class 大象 {

public static void in() {

System.out.println("装进大象");

} }

class 冰箱 {

public static void open() {

System.out.println("打开冰箱门");

}

public static void close() {

System.out.println("关闭冰箱门");

} }

class Demo {

public static void main(String[] args) {

冰箱**对象调用**开门**方法**

大象**对象调用**进去**方法**

冰箱**对象调用**关门**方法**

} }

### 3、面向对象特征

封装、继承、多态

## 类与对象

### 1、为什么引入类

String name int age pyt应该是什么类型？

Java定义的基本数据类型不够我们使用了，所以自己定义Person类等

### 2、概念

类：是一组相关的属性和行为的集合

可以理解为构造对象的一个蓝图或者**模版**，是**抽象**的概念

对象：是该类事物的具体体现

是以类为模型创建的具体实例，是对类的一种**具体**化。

举例：

类 学生

对象 班长就是一个对象



### 3、类的定义

类是对象的抽象，它用于描述一组对象的共同特征和行为。

类中可以定义成员变量和成员方法，

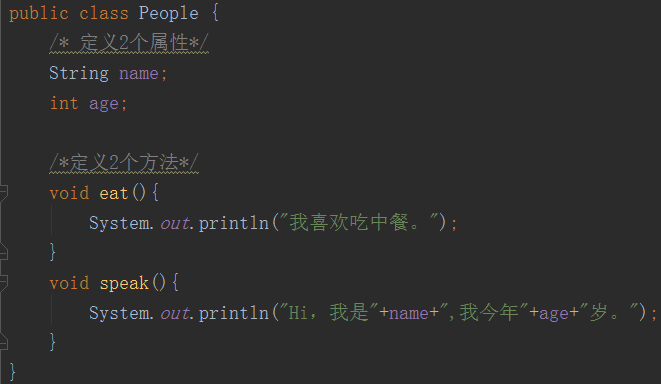
其中成员变量用于描述对象的特征，也被称作属性，

成员方法用于描述对象的行为，可简称为方法。

人类：

属性 ：姓名、年龄…

行为（方法）：吃饭、说话…



### 4、对象的创建使用

创建对象：

类名 对象名 = new 类名();

使用：

对象名.成员变量

对象名.成员方法



#### 练习3-1 类和对象的使用

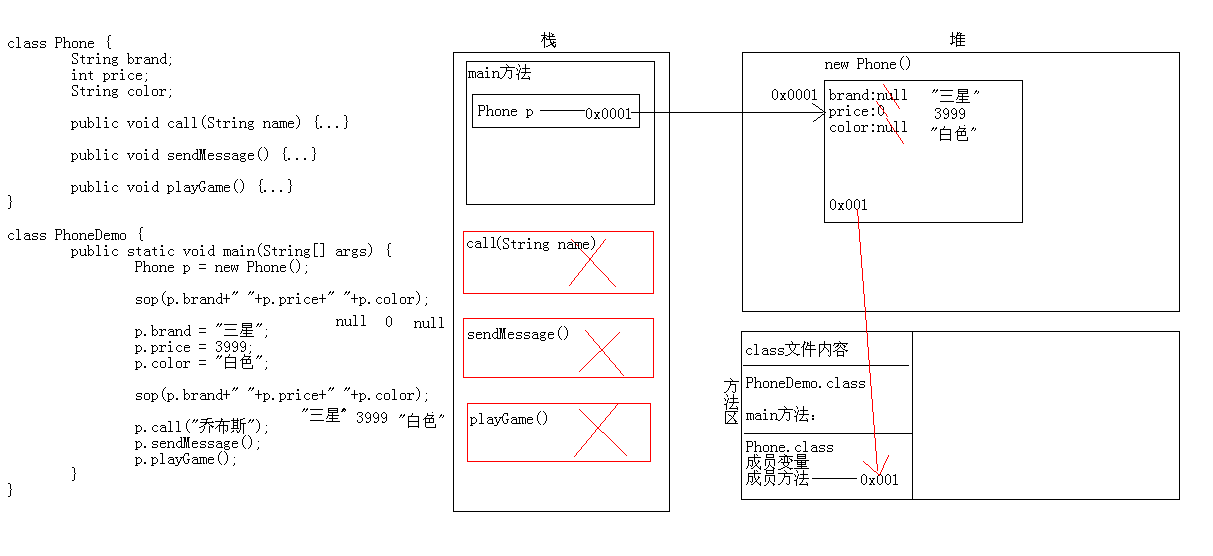
练习1：设计并创建一个手机类，生成一个手机对象，并调用其相应的属性和方法。

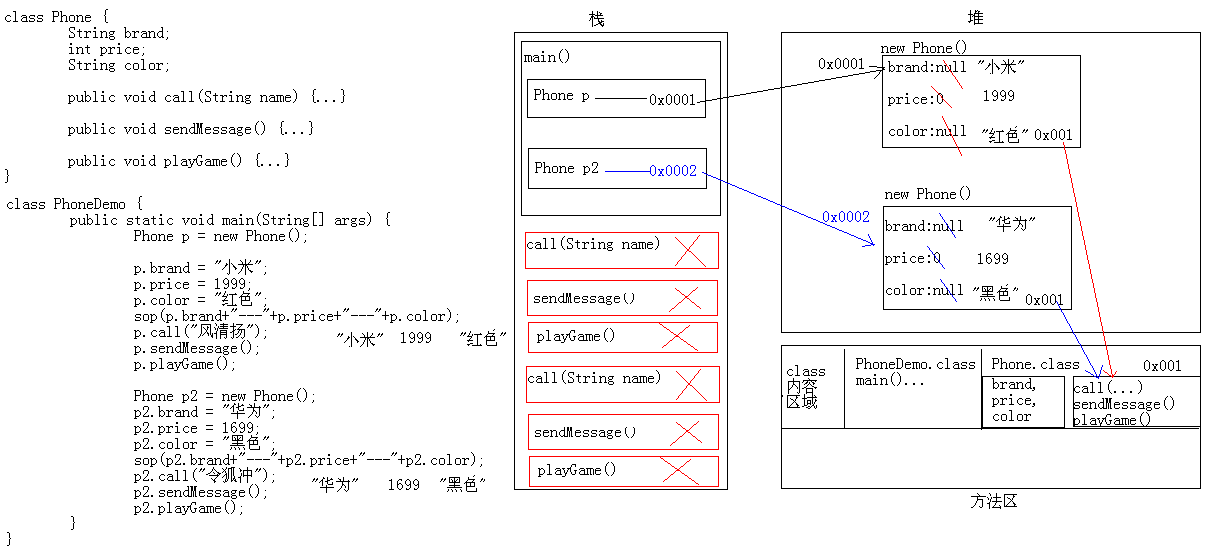
练习2：设计并创建一个学生类，生成一个学生对象，并调用其相应的属性和方法。

### 5、对象内存图

**视频：06.24\_面向对象(一个对象的内存图).avi**

**06.25\_面向对象(二个对象的内存图).avi**





## 封装

在设计一个类时，应该对成员变量的访问作出一些限定，不允许外界随意访问，这就需要实现类的封装。

pyt.age = -20; 合法吗？

### 1、为什么引入封装

**隐藏**对象的属性和实现**细节**，仅对外提供公共访问方式

增强了安全性。

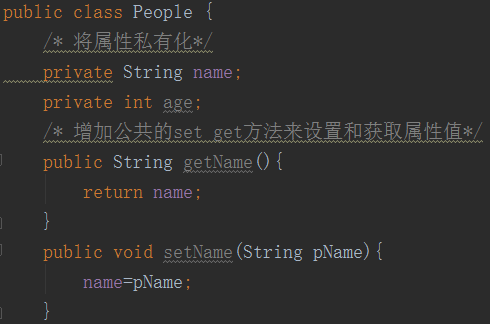
操作：

属性前加private修饰，表示该属性是私有的。

通过setter和getter，设置、取得实例变量的值。

#### 练习3-2 类的封装

练习1：利用封装修改People类， 并创建对象调用。



setAge()方法要对传入参数进行判断



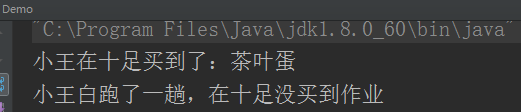
练习2：修改手机类（封装）

#### 练习3-3 超市购物程序设计

1）描述：

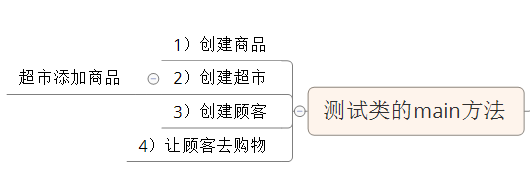
顾客去超市购物，如果超市中存在该商品，则输出：顾客在某超市买到了商品；否则输出：顾客白跑一趟，在某超市没有买到。

2）结果：



3）分析：





4）代码：

5）生成setter/gettter方法

右键【Generate】，或者快捷键Alt+Insert，选择Getter and Setter，按住Shift键属性可以多选。

6）优化

每次实例化对象后，都要调用setter方法给属性赋值，很啰嗦，可以通过构造方法实现实例化对象的同时对这个对象的属性进行赋值。

### 2、构造方法

#### （1）作用

用于对对象的数据进行**初始化**

#### （2）格式

方法名与类名相同

没有返回值类型，连void都没有

没有具体的返回值

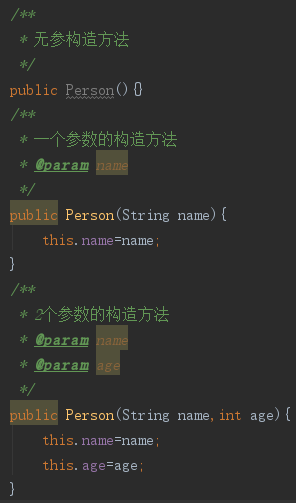
#### （3）注意

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，系统将不再提供默认的，最好手动写上

构造方法也是可以重载的

#### 练习3-4 构造方法练习



测试类：创建2个对象，分别调用构造方法，输出

#### 练习3-5 标准类的练习

1、标准类（封装、构造方法）：

1）成员变量

2）构造方法

无参构造方法

带参构造方法

3）成员方法

getXxx()

setXxx()

2、给变量赋值

1）用无参构造方法+setXxx()

2）有参构造方法

练习1：定义一个类Company：

定义1个String类型的name属性，表示公司名

定义1个int类型的num属性，表示员工人数

定义1个空构造方法

定义1个为name，num属性初始化的构造方法

定义1个output()方法：输出公司名以及员工人数

定义一个类Test1，在main()方法中完成：

创建1个Company类的对象taobao，调用无参构造方法。

为taobao对象的name和num属性分别赋值为：淘宝，4000

创建1个Company类的对象qq，调用有参构造方法初始化

调用taobao对象的Output()方法。

调用qq对象的Output()方法。

练习2：定义一个标准长方形类，定义求周长和面积的方法，然后定义一个测试类Test2，进行测试。

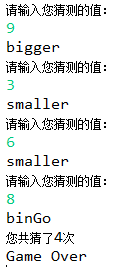
练习3：定义一个类MyMath，提供基本的加减乘除功能，然后进行测试。

#### 练习3-6 猜数字游戏

产生一个随机数，让用户从键盘输入数字猜测（直到猜对为止猜结束，否则一直重复猜测）；如果猜大了 提示“bigger”，如果猜小了，提示：“smaller”，如果猜中，提示“binGo” 输出：总共猜测的次数。

**分析需要哪些对象？每个对象需要做什么事情？**

运行结果如图所示：



### 3、关键字

#### （1）private&public

权限修饰符

private：私有的，可以修饰成员变量和成员方法，只能在本类中访问

private是封装的一种体现

public：公共的。

#### （2）this

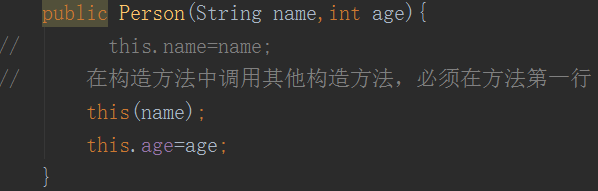
代表当前类的引用对象

哪个对象调用方法，该方法内部的this就代表那个对象

**this的应用场景：**

1）解决了局部变量隐藏成员变量的问题 this.name=name

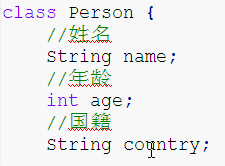
2）在构造方法中调用其他构造方法this(…)，必须在方法第一行



#### （3）static

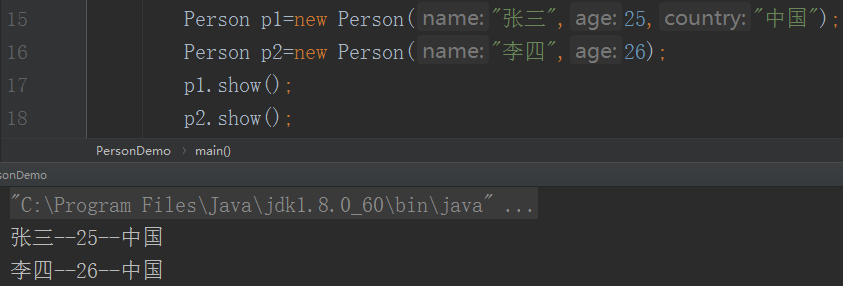
静态的。main方法就是静态的，不用创建对象调用

**1）为什么要引入static**



实例化对象时，每个对象的姓名和年龄都不一样，都需要初始化赋值，但国籍都一样的情况下，每次创建对象，在堆内存都要开辟这样的控件，会太浪费了。

将这种变量定义为static，就可以被所有对象共享了，如下：



**2）特点**

随着类的加载而加载

优先与对象存在

被类的所有对象共享

这其实也是我们判断该不该使用静态的依据。

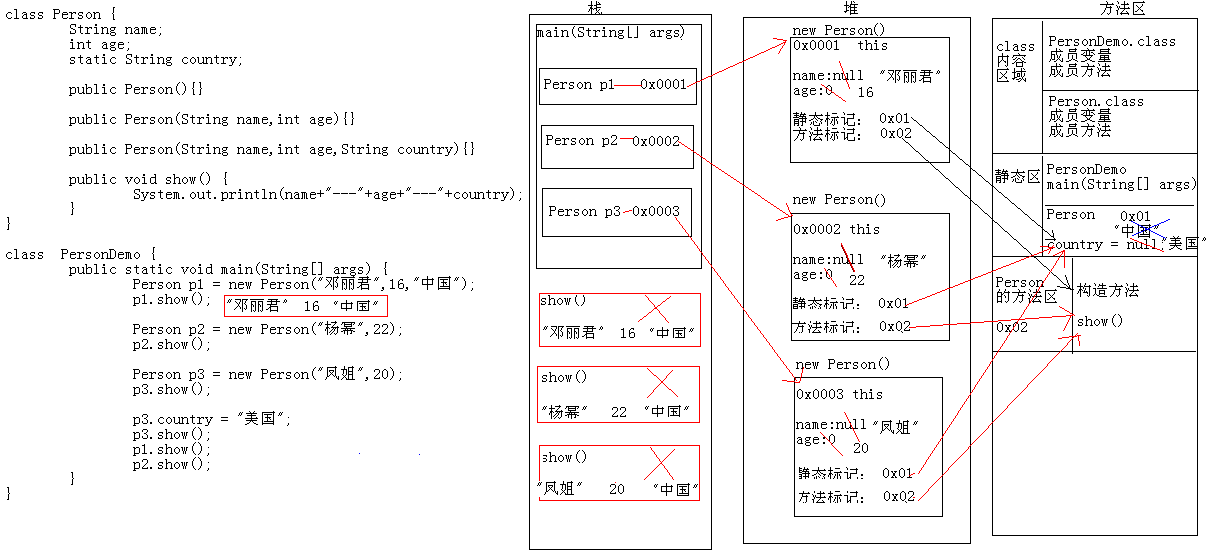
举例：饮水机和水杯的问题思考

可以通过类名调用

既可以通过对象名调用，也可以通过类名调用，建议通过类名调用。

**3）内存图**

07.23\_面向对象(static的内存图解).avi



### 练习3-7 银行新用户现金业务办理

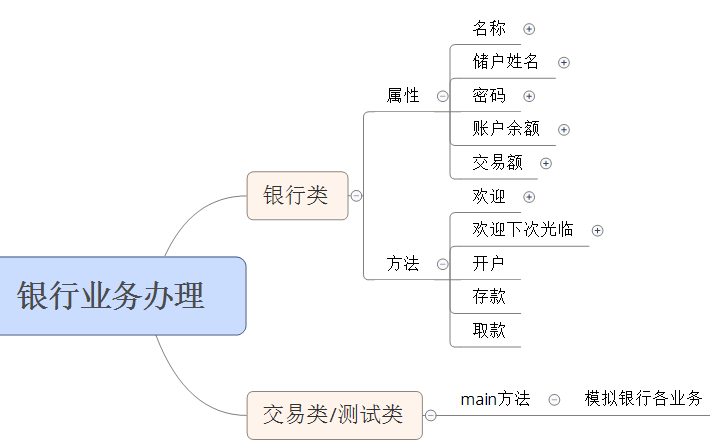
1）任务描述：

模拟新用户到银行办理开户、存取现金业务。

银行对用户的到来表示欢迎；对用户离开表示欢迎下次光临；用户完成开户、存款、取款操作后，要提示用户的账户余额。

取款操作时，若密码错误提示，若余额不足提示，取款成功显示余额。

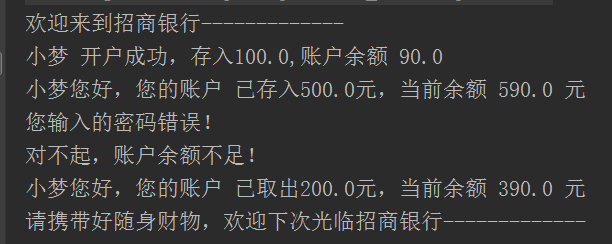
2）思路分析：



注意：封装、构造方法、静态、this的使用

3）代码实现：

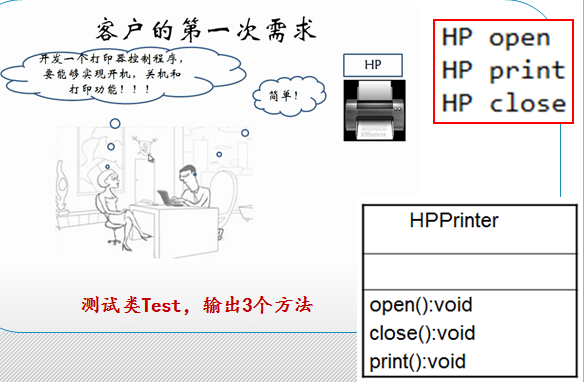
4）运行结果：



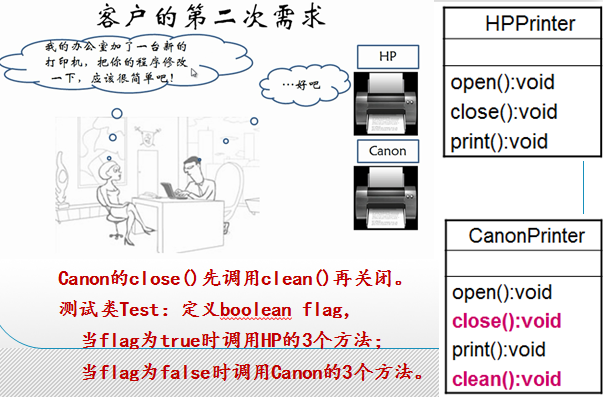
## 继承

### 1、为什么引入继承

需求一：



需求二：



需求三：



都是打印机，总有相同功能的吧…

**目标：消除重复代码，** 抽取共同的属性和方法

### 2、继承概念

把多个类中**相同的成员给提取出来定义**到一个独立的类中。然后让这多个类和该独立的类**产生一个关系**，这多个类就具备了这些内容。**这个关系叫继承**。

类似子女继承父母财产，子女遗传父母相貌…

类的继承是指在一个现有类的基础上去构建一个新的类，构建出来的新类被称作子类，现有类被称作父类，子类会自动拥有父类所有可继承的属性和方法。

### 3、继承的使用

在程序中，如果想声明一个类继承另一个类，需要使用extends关键字。

格式：

class 子类名 extends 父类名 {}

#### 练习3-8 继承的使用练习



1）定义1个University类，3个属性：math表示数学成绩，english表示英语成绩，chinese表示语文成绩，sport表示体育成绩。1个方法：enter()，若英语、数学、语文总分大于等于180分，且体育成绩大于等于60，则输出“恭喜！3门课总分为：xxx，且体育达标，达到大学录取线”；否则，输出“您落榜了！”

2）定义1个VIPUniversity类，继承自University类，重写enter()方法，若四门课总分大于等于290，则输出“恭喜！4门课总分为：xxx，达到重点大学录取线”；否则，输出“您未被重点大学录取！”

3）定义1个测试类Test1：创建一个VIPUniversity类的对象vip，给vip对象的属性赋值math=82.5，chinese=80,English=75,sport=61。调用vip对象的enter()方法。运行查看结果。

### 4、继承总结

#### 1）好处

提高了代码的复用性和维护性（消除重复代码）

让类与类产生了一个关系，是多态的前提

#### 2）弊端

类的耦合性增强，某个类的改变，就会影响其他和该类相关的类

打破了封装性

所以：

**不要为了部分功能而去继承**

低耦合高内聚

耦合：类与类的关系

内聚：自己完成某件事情的能力

#### 3）何时使用继承

继承体现的是**is a**的关系

采用假设法

比如猫 is a 动物（猫是动物），那么就可以设计为继承

比如汽车和引擎，就是**has a**的关系，表示组合，一个类中包含了其他类。

举例：等边三角形 三角形；CPU 电脑

#### 4）特点

**特点：继承、重写、新增**

1、子类可继承父类所有可继承的方法和属性（说明还有不能继承的）

子类不能继承父类的私有成员；

不能继承父类的构造方法，但可以通过super访问

2、单继承：只有一个直接父类（只有一个爸爸，还可以有爷爷…）

3、子类可新增属性、方法

4、子类可修改/重写父类的方法

5、多个类可以继承一个父类（一个爸爸可以有多个孩子）

#### 5）继承中成员关系

（1）成员变量

子类和父类有不同名的成员变量（子类有新增的成员变量，当然OK）

同名呢，怎么访问？（**覆盖**）

**子类的方法访问变量的查找顺序：**

在子类方法的局部范围找，有就使用。

在子类的成员范围找，有就使用。

在父类的成员范围找，有就使用。

找不到，就报错。

（2）成员方法

不同名（新增，当然OK）

同名呢？（**覆盖**）

通过子类对象访问一个方法的查找顺序：

在子类中找，有就使用

在父类中找，有就使用

找不到，就报错

（3）构造方法

子类能继承父类的构造方法吗？

子类中所有的构造方法默认都会访问父类中空参数的构造方法

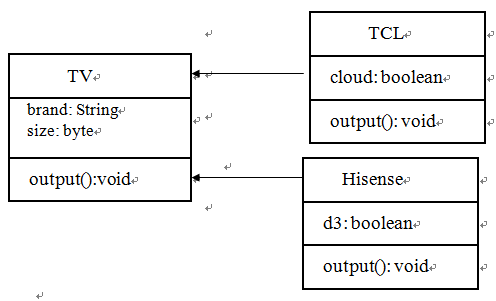
如何父类没有默认构造方法怎么办？

子类通过super去显示调用父类其他的带参的构造方法

子类通过this去调用本类的其他构造方法，但一定会有一个去访问了父类的构造方法

**让父类提供无参构造方法**

#### 练习3-9 继承练习



**1）定义TV类**，其有2个属性：String brand表示电视机牌子

byte size表示电视机尺寸

定义其构造方法（无参数的和初始化2个参数）

定义output()方法，输出电视机牌子和尺寸

**2）定义TCL类，**继承TV类，

新增1个属性：boolean cloud，表示是否是云电视

定义构造方法（无参数的、初始化3个参数的构造方法）

重写output()方法，若cloud为false，则调用父类的output()方法；否则输出电视机牌子、尺寸、是云电视。

**3）定义Hisense类**，继承TV类

新增1个属性 boolean d3，表示是否是3D电视

定义构造方法（无参数的、初始化3个参数的构造方法）

重写output()方法，若d3为true，输出电视机牌子、尺寸、是3D电视。否则输出电视机牌子、尺寸。

**4）定义1个测试类Test1：**

创建TCL类的对象tcl，调用output()方法；

创建Hisense类的对象hisense，调用output()方法。

### 5、final&super

#### 1）super

与this类似，父类引用。

使用super访问父类的成员变量和成员方法：

super.成员变量

super.成员方法()

使用super访问父类的构造方法：

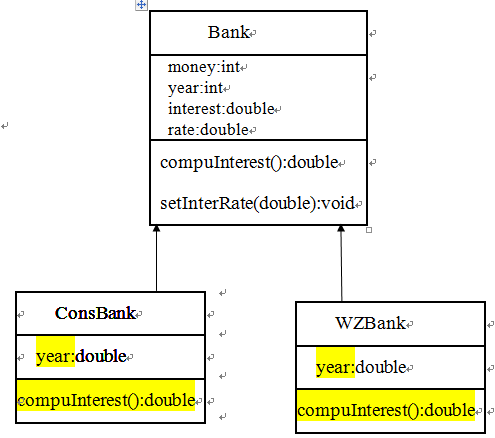
super(…)

#### 练习3-10 super的使用—代码填空

Bank.java ConsBank.java WZBank.java Test\_Bank.java

Bank：存款利息计算：按年来计算。不考虑多出的天数。

各个银行的存款利息计算：整年 按照Bank算法；多出的天数各家银行算法不一样。最终利息就是整年的利息+天数的利息



**年（小数），包括天数**

**计算利息（按整年算）**

**设置利率**

**利率**

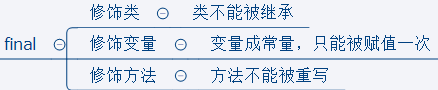
**利息**

**年（整数）**

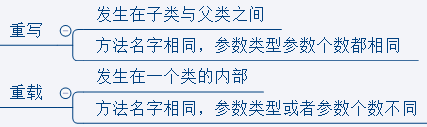
**钱**

#### 2）final

最终的意思



### 6、重写 VS 重载



## 多态

### 1、为什么引入多态

屏蔽子类的差异性，提高扩展性。

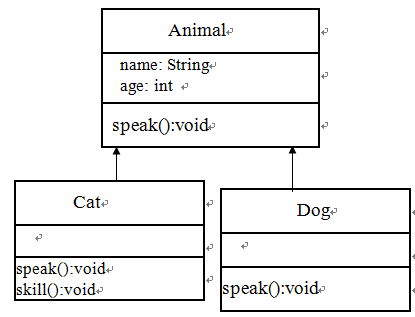
多态：同一个对象在不同时刻体现出来的不同状态。

### 2、多态的前提

为了实现多态，允许使用一个父类类型的变量来引用一个子类类型的对象，根据被引用子类对象特征的不同，得到不同的运行结果。



### 练习3-11 多态的使用



1）Animal的属性设置为private，使用setter/getter设置/获取属性值。

2）测试类TestAnimal中，创建以下2个对象：

Animal kitty=new Cat();

Animal bobi=new Dog()

3）测试类TestAnimal中，观察以下语句的输出结果：

kitty.speak();

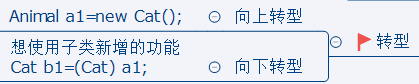
bobi.speak();

kitty.skill();

### 3、为什么转型

想使用子类新增的功能（向下转型）

提高扩展性（练习3-12 向上转型）



### 练习3-12 多态的作用

（1）定义Car类：

1个方法：run()，输出“car is running”

（2）定义BMW类，继承Car

重写run()方法，输出“BMW is running”

（3）定义QQ类，继承Car

重写run()方法，输出“QQ is running”

（4）定义Driver类：

1、定义方法run(BMW bmw)：调用bmw的run()

2、重载run方法run(QQ qq)：调用qq的run()

3、main()方法中：

1）创建Driver类的对象driver

2）调用driver的run()，表示司机开BMW

3）调用driver的run()，表示司机开QQ

思考：如果增加车型benz，要如何表示司机开benz

如果再增加10种车型呢？

如何修改，可以减少代码？

### 4、优缺点

优点：提高代码的维护性(继承体现)

提高代码的扩展性(多态体现)

缺点：父不能使用子的特有功能。

### 5、成员访问的特点

1）成员变量：

访问的是父类的变量

不能访问子类新增的变量

2）成员方法： **（为啥与众不同）**

**访问的是子类重写的方法**

不能访问子类新增的方法

3）静态方法：

访问的是父类的静态方法

不能访问子类新增的静态方法

4）构造方法：

创建子类对象时，默认调用父类构造方法

成员方法有重写！！

## 抽象

### 1、为什么引入抽象

强调子类必须具有某项功能（必须重写某方法），但不关心其具体实现

### 2、抽象类天生就是当“爹”

就是为了被子类继承，重写其所有抽象方法

### 3、面向抽象编程

抽象类和抽象方法必须用关键字abstract修饰

抽象类不能被实例化

一个没有具体的方法体的方法是抽象的方法。

#### 练习3-13 抽象与多态

**面向抽象编程**：

让该类的设计者只关心操作，不关心具体实现；

让类的每种可能变化交给各个子类去实现。

1）定义一个抽象类Shape；方法的具体实现由各子类去完成。

area()：求面积。 getName()：返回是何图形，如“三角形”

output()：调用getName()和area()方法，输出是何图形，面积是多少。如“三角形的面积是3.5”

2）实现这3个类。

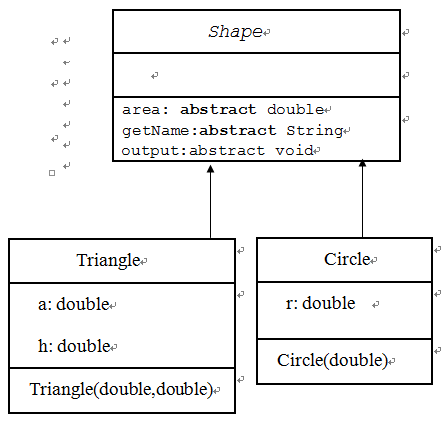
另编写Shape\_Test类测试生成对象，利用多态，

调用output()：Shape s1=**new** Triangle(2,3.5);

Shape s2=**new** Circle(4);

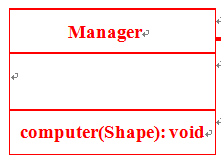
s1.output();

s2.output();



3）思考：若再定义平行四边形、五边形、椭圆等等形状，要得到每种形状的名称和面积，代码量大，如何简化？

4）增加一个类Manager，定义方法computer(Shape s)：输出形状名字和面积。



## 接口

### 1、为什么引入接口

**想增加额外功能（扩展功能）**

回顾猫狗案例，它们仅仅提供一些基本功能。

比如：猫钻火圈，狗跳高等功能，不是动物本身就具备的，

是在后面的培养中训练出来的，这种额外的功能，java提供了接口表示。

### 2、接口是“纯爷们”

**所有方法都是抽象方法**

### 3、定义与实现

接口用关键字interface修饰

interface 接口名 {}

类实现接口用implements修饰

class 类名 implements 接口名 {}

#### 练习3-14 接口练习

1）定义接口Pet。其中定义2个无返回值方法： beFriendly()、 play()

2）Cat类实现Pet接口

3）在AnimalTest中，调用方法

### 4、特点

1）成员变量

只能是常量

默认修饰符：public static final

2）构造方法

没有构造方法

3）成员方法

只能是抽象的

默认修饰符：public abstract

接口是有**常量和抽象方法**组成的特殊类。

### 5、类与接口的关系

1）类与类

继承关系

只能单继承，可以多层继承（只能一个爹，可以有爷爷）

2）类与接口

实现关系

可以单实现，也可以多实现

可以再继承一个类的同时，实现多个接口

3）接口与接口

继承关系，可以单继承，也可以多继承

### 6、抽象与接口的区别

1）成员区别

抽象类：变量、常量、抽象方法、非抽象方法

接口：常量、抽象方法

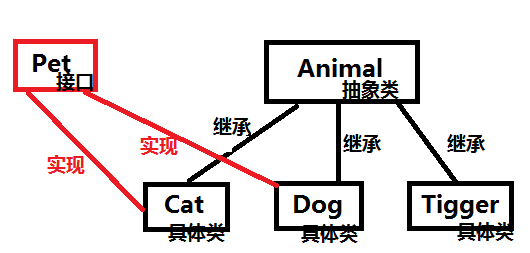
2）关系区别



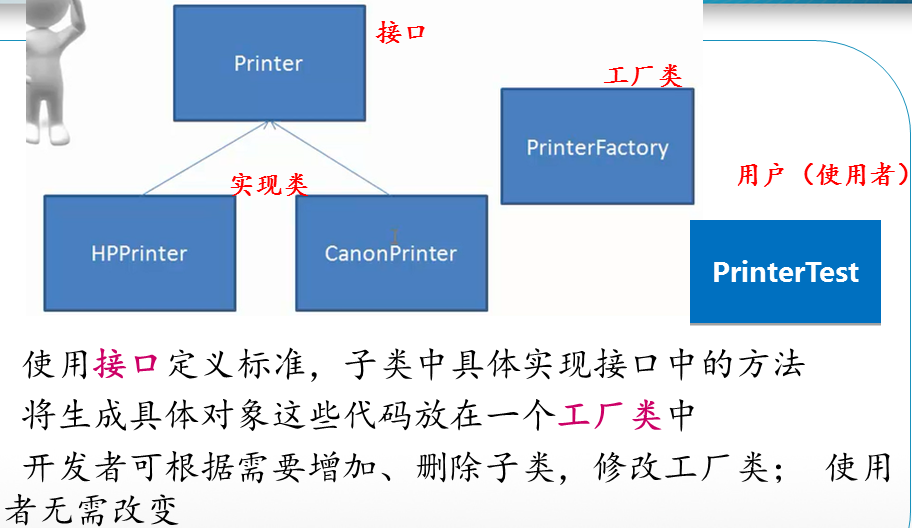
3）设计理念区别

**抽象类：共性功能，is a的关系**

**接口：扩展功能，like a的关系**



### 7、面向接口编程



#### 练习3-15 面向接口编程

1. 定义**接口Printer**，有3个方法：open() print() close()
2. 定义类HPrinter，实现接口Printer。

方法都是输出一句话，表示：HP打开、HP打印、HP关闭

1. 定义类CPrinter，实现接口Printer。

新增clean()方法：输出Canon clean

close()方法：先调用clean()方法，再输出Canon关闭

1. 定义测试类PrinterTest，

接收键盘输入，当输入是0使用HPrinter 打印功能，当输入是1使用CPrinter打印功能。

**【以上完成了接口的定义和实现；接下来要修改程序，用接口作为多态类型】**

1. 思考：如果增加打印机类型，则需要修改PrinterTest。
2. 增加1个类：PrinterFactory，用于实现PrinterTest中创建对象的功能。

其中定义一个方法：Printer getPrinter(int)，根据传入的方法来决定要返回什么类型的对象。

7、修改测试类PrinterTest：创建PrinterFactory类型的对象PF，调用其getPrinter()方法来实现。

【让每个类的功能尽量单一化。**（面向对象：单一职责），**多人负责，出了问题，谁都不负责（井盖）】

#### 练习3-16 接口与多态



1. **定义接口Car**的2个方法。

use()：输出车的用途

getName()：返回车名，比如出租车

1. **定义子类Bus和Taxi**，实现Car接口。
2. **定义Company类**，定义Car manage(int)方法：

根据传入的参数，决定返回的对象

1. **定义CarTest类**：在main()方法中：

根据键盘输入来决定要乘坐出租车还是公共汽车（利用Company类的方法）。

#### 练习3-17 接口与多态

1、定义**接口ChatTool**，有3个方法：

login()：表示登陆； send()：表示发送消息；exit()：表示退出

2、定义类WeiBo，实现ChatTool接口

增加fans()方法，输出“WeiBo求粉”

send()方法：调用fans()方法，再输出“WeiBo发消息”

login()：输出“WeiBo登录”；exit()：输出“WeiBo退出”

3、定义类WeiXin，实现ChatTool接口

增加shake()方法，输出“WeiXin摇一摇”

send()方法：调用shake()方法，再输出“WeiXin发消息”

login()：输出“WeiXin登录”；exit()：输出“WeiXin退出”

4、定义类ChatFactory，定义方法： ChatTool getTool(int)

根据传入的参数决定返回什么对象

6、定义测试类ChatTest，在main()方法中：

根据键盘输入来决定要使用微博还是微信进行通信

### 练习 3-18 模拟计算机开机关机过程

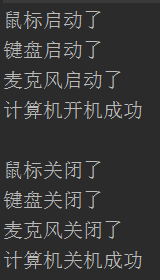
计算机上都有鼠标、键盘、麦克风设备，这些设备都是通过USB接口连接到计算机。当计算机开机时，这些设备都开机了，当计算机关机时，这些设备也会随之关机。

1）请分析这里的名词：计算机、鼠标、键盘、麦克风、USB之间有什么关系？如何设计他们之间的关系

2）只考虑开机和关机这2个功能

3）测试类中：首先创建计算机对象，往该对象中添加鼠标、键盘、麦克风设备；执行计算机开机功能（这3个设备开机了，计算机就启动好了），再次执行计算机关机功能（这3个设备关机了，计算机就关闭好了）

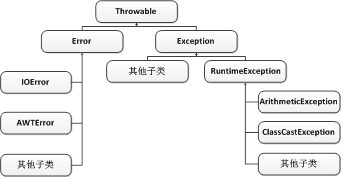
4）运行结果如下所示，请编码模拟这一过程



### 练习3-19 模拟物流快递系统程序设计

## 异常

### 1、异常 VS错误



Error代表程序中产生的**错误**：

它表示Java运行时产生的系统内部错误或资源耗尽的错误，是比较严重的，仅靠修改程序本身是不能恢复执行的

Exception代表程序中产生的异常：

它表示程序本身可以处理的错误

1）**运行时异常**

RuntimeException类

运行时异常一般是由程序中的逻辑错误引起的，在程序运行时无法恢复。比如通过数组的角标访问数组的元素时，如果超过了数组的最大角标，就会发生运行时异常

2）**编译时异常**

Exception类下所有其他的子类都用于表示编译时异常

使用后面介绍的这2种方法处理。

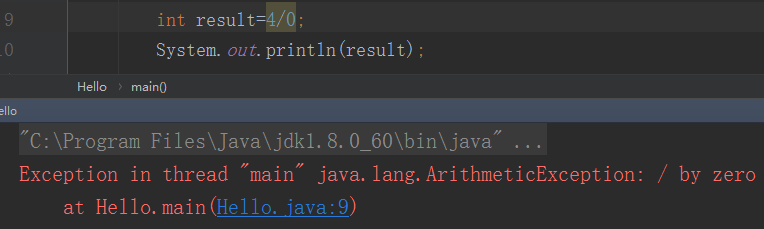
### 2、捕获异常try catch finally

/\* try: 可能发生异常的Java语句  
\* catch: 针对异常进行处理的代码  
\* finally: 无论程序是否发生异常都要执行的代码\*/  
try {  
   
}catch (Exception e){

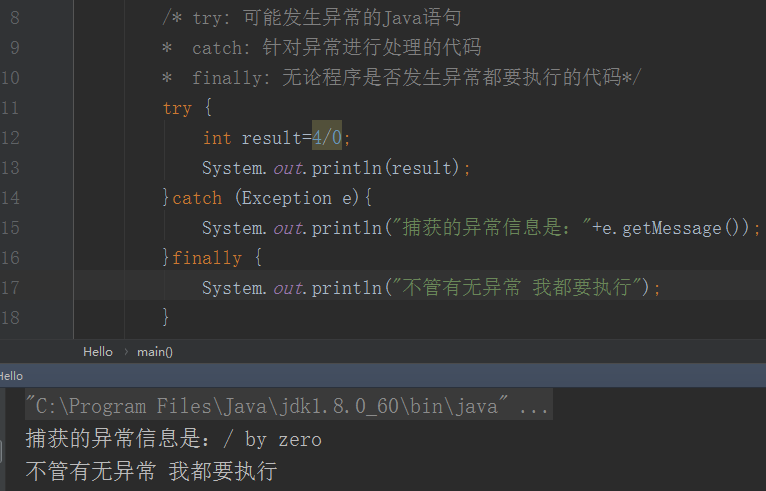
}finally {  
   
}

#### 练习3-20 异常捕获

若不用异常捕获，除数为0时会出现异常。



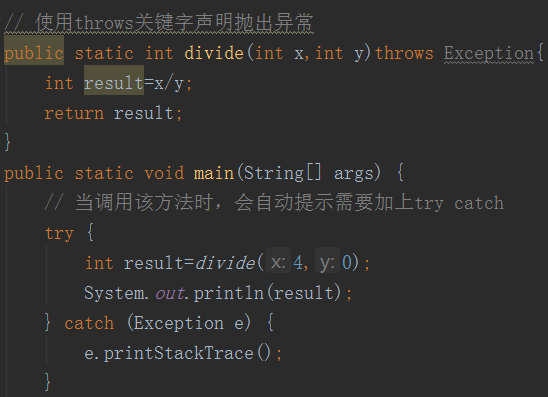
使用异常捕获，增加代码健壮性：



### 3、抛出异常throws

如果去调用一个别人写的方法时，是否能知道别人写的方法是否会有异常呢？这是很难做出判断的。针对这种情况，**Java中允许在方法的后面使用throws关键字对外声明该方法有可能发生的异常**，这样调用者在调用方法时，就明确地知道该方法有异常，并且必须在程序中对异常进行处理，否则编译无法通过。

#### 练习3-21 抛出异常



### 4、自定义异常

自定义的异常类只需继承Exception类，在构造方法中使用super()语句调用Exception的构造方法即可。

#### 练习3-22 自定义异常

1）自定义异常



2）使用自定义异常类

**throw**



# Java API

## 是什么

API(Application Programming Interface)

应用程序编程接口(帮助文档)

Java提供给我们使用的类

只要学会使用这些类，不用关心具体实现

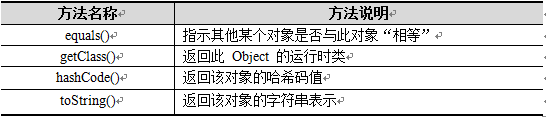
使用帮助文档 JDK API

打开文档，点击“显示”——“索引”，输入框



## Object类

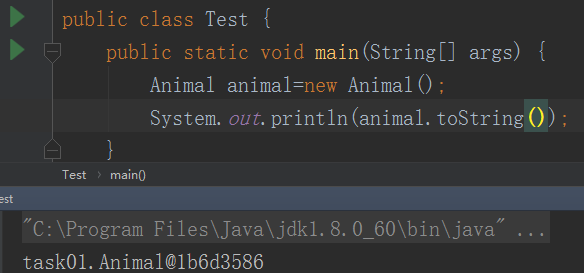
它是类层次结构的根类，每个类都直接或间接继承自该类，所有对象（包括数组）都实现了这个类的方法。

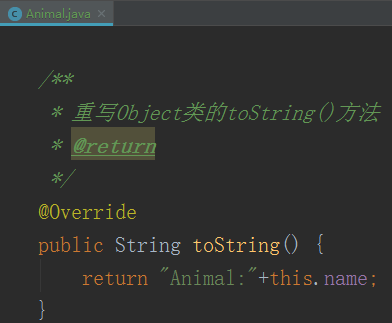


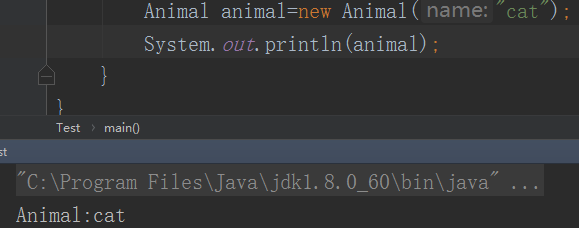
### 1、toString()

返回对象的字符串表示，默认是由类的全路径+'@'+哈希值的十六进制表示。

这个表示其实是没有意义的，一般子类都会重写该方法。







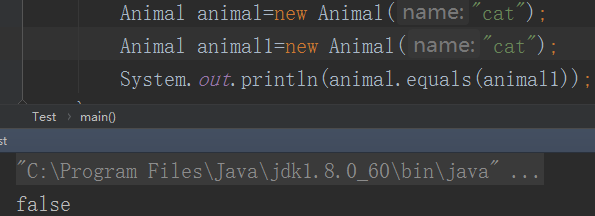
**直接输出一个对象名称，其实默认调用了该对象的toString()方法。**

### 2、equals()

比较两个对象是否相同。默认情况下，比较的是地址值是否相同。

而比较地址值是没有意义的，所以，一般子类也会重写该方法。

**1）不重写equals()方法，比较俩对象：**



2）**重写equals()方法，比较俩对象：**

**3）==和equals()的区别**

==

基本类型：比较的是值是否相同

引用类型：比较的是地址值是否相同

equals()

只能比较引用类型。默认情况下，比较的是地址值是否相同。

但是，我们可以根据自己的需要重写该方法。

## 常见类

### 1、String

#### 1）概念

多个字符组成的一串数据，由许多单个字符连接而成。

用双引号

其实它可以和**字符数组**进行相互转换

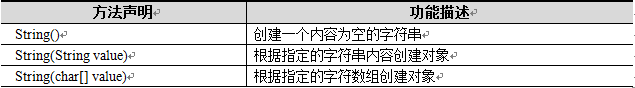
#### 2）初始化

**（1）字符串常量初始化**

String str1=”hello”;

结果是一个字符串对象

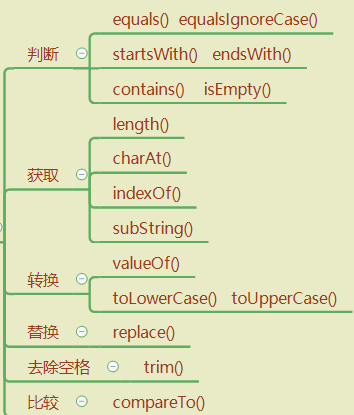
**（2）构造方法初始化**





#### 3）常见操作





#### 练习4-1 模拟登录

模拟登录，给3次机会，并提示还有几次。

分析：

1）定义用户名和密码。已存在的。

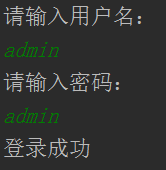
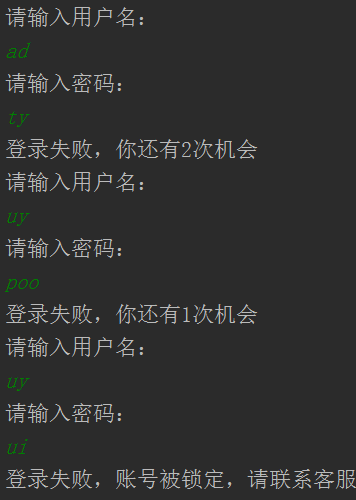
2）键盘录入用户名和密码。

3）比较用户名和密码。

如果都相同，则提示登录成功

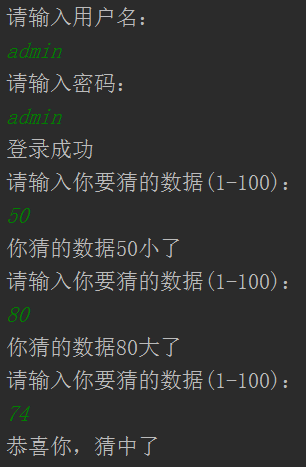
如果有一个不同，则提示登录失败，还有几次机会

3次登录都失败，则提示账号被锁定。

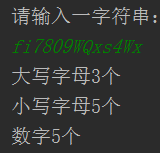
改进一下：

模拟登录,给三次机会,并提示还有几次。如果登录成功，就可以玩猜数字小游戏了。



#### 练习4-2 统计字符个数

接收键盘输入一字符串，统计字符串中大写，小写及数字字符的个数，不考虑其他字符。

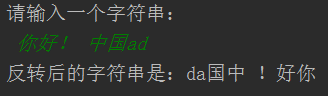


提示：使用charAt() 可以得到字符串中Index位置上的字符。

#### 练习4-3 字符串反转

举例：键盘录入”abc”

输出结果：”cba”



提示：toCharArray()将字符串转为一个字符数组

#### 练习4-4 大串中小串出现的次数

举例：

在字符串"woaijavawozhenaijavawozhendeaijavawozhendehenaijava"

结果：

java出现了4次

分析：

1）已知大串和小串

2）定义一个统计变量count，初始化值是0

3）先在大串中查找一次小串第一次出现的位置index

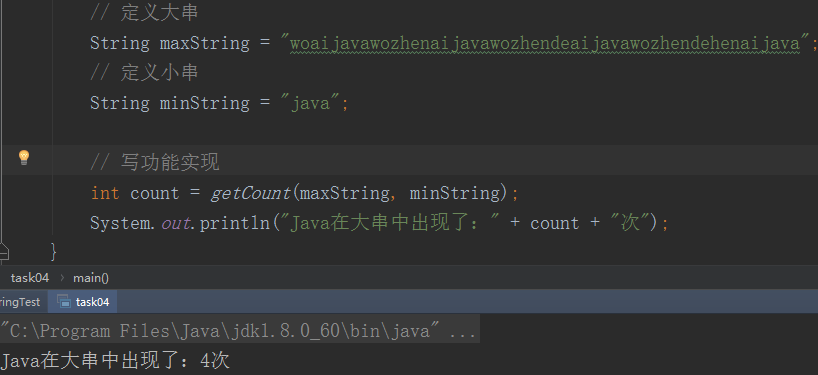
4）循环（index不是-1）：

index是-1，说明不存在了，就返回count

index不是-1，说明存在：

（1）count++

（2）大串重新赋值：把刚才的index+小串的长度作为开始位置截取上一次的大串



### 2、StringBuffer

#### 1）概述

字符串是常量，因此一旦创建，其内容和长度是不可改变的。

为了便于对字符串进行修改，在JDK中提供了一个StringBuffer类（也称字符串缓冲区）。

StringBuffer类和String类最大的区别在于它的内容和长度都是可以改变的。

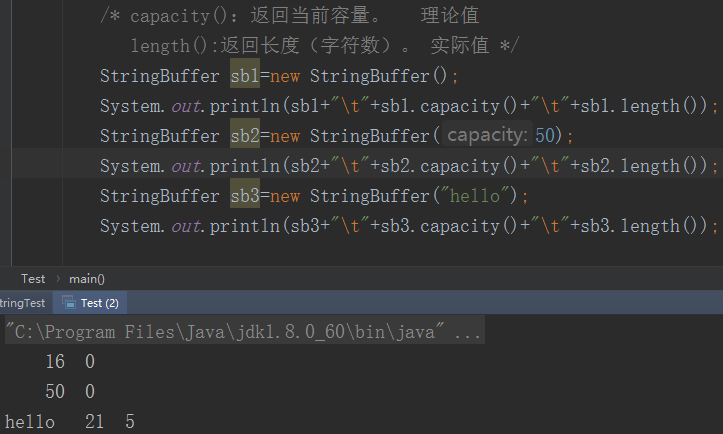
StringBuffer类似一个字符容器，当在其中添加或删除字符时，并不会产生新的StringBuffer对象。

#### 2）初始化

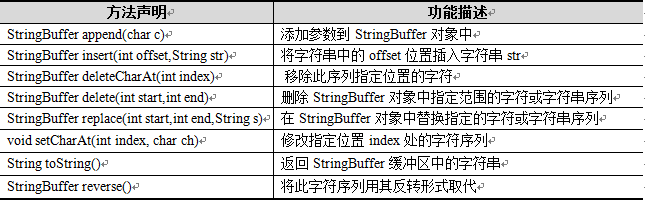
StringBuffer()

StringBuffer(int size)

StringBuffer(String str)

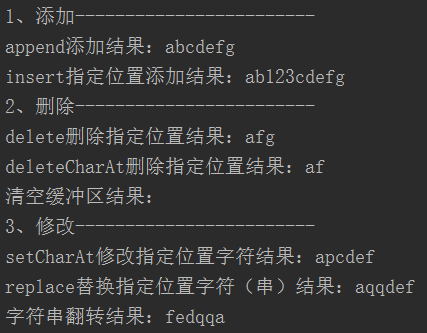


#### 3）操作



#### 练习4-5 StringBuffer常用方法

练习使用StringBuffer方法完成字符串的添加、删除、修改，运行结果如下图所示：



### 3、Math

Math类是数学操作类，提供了一系列用于数学运算的静态方法，包括求绝对值、三角函数等。Math类中有两个静态常量PI和E，代表数学常量π和e

常见方法：

1）绝对值 abs()

2）向上取整 ceil()

3）向下取整 floor()

4）两个数据中的大值 max()

5）a的b次幂 pow(a,b)

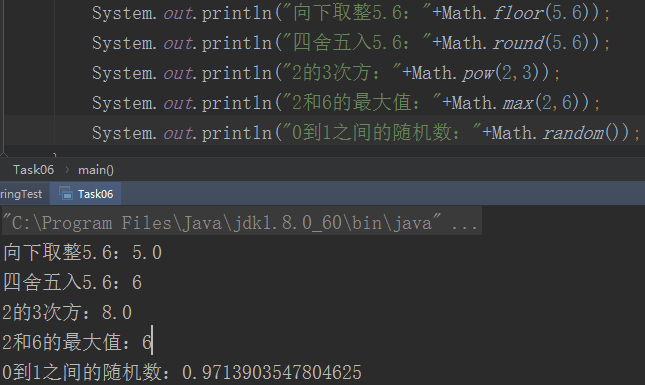
6）随机数 random()

7）四舍五入 round()

8）正平方根 sqrt()

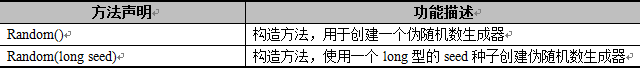
……查API

#### 练习4-6 Math方法

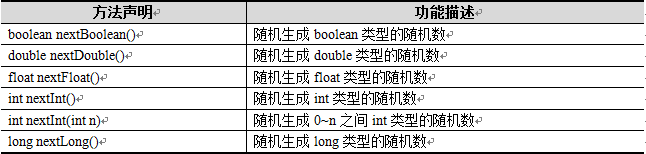


### 4、Random

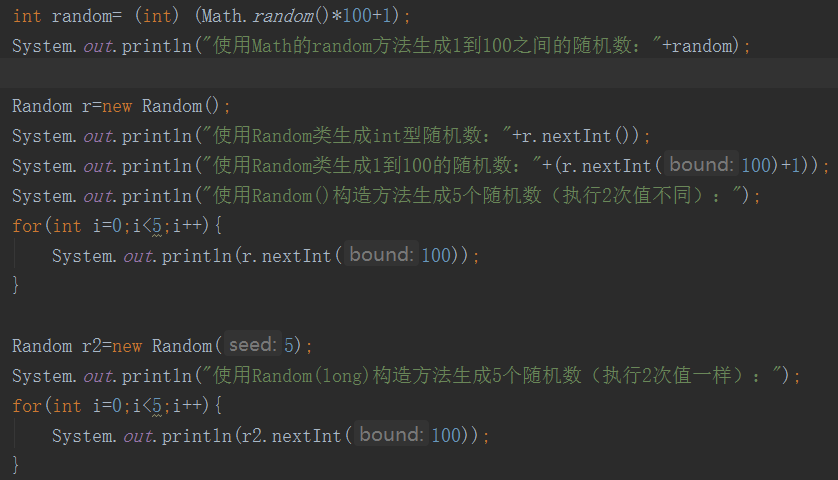
可以在指定的取值范围内随机产生数字



相对于Math的random()方法而言，Random类提供了更多的方法来生成各种伪随机数，不仅可以生成整数类型的随机数，还可以生成浮点类型的随机数，表中列举了Random类中的常用方法。



#### 练习4-7 生成随机数



### 5、System

System类定义了一些与**系统相关的属性和方法**，它所提供的属性和方法都是**静态的**，因此，想要引用这些属性和方法，直接使用System类调用即可。

输入： System.in

输出：System.out.println()





### 6、Runntime

Runtime类用于表示虚拟机运行时的状态，它用于封装JVM虚拟机进程。

在程序中通常会通过该类的实例对象来获取当前虚拟机的相关信息。

运行这段代码，查看结果：

Runtime rt = Runtime.getRuntime(); // 创建一个Runtime实例对象

Process process = rt.exec("notepad.exe");// 得到表示进程的Process对象

Thread.sleep(3000); // 程序休眠3秒

process.destroy(); // 杀掉进程

## 包装类

### 1、概述

为了让基本类型的数据进行更多的操作，为此提供了对应的包装类类型。

通过包装类可以**将基本数据类型的值包装为引用数据类型的对象**。



### 2、拆箱&装箱

包装类和基本数据类型在进行转换时，引入了装箱和拆箱的概念。

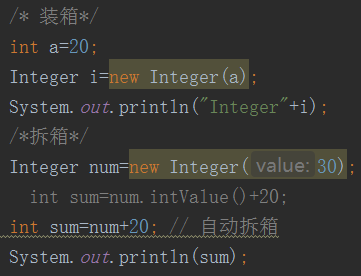
装箱是指将基本数据类型的值转为引用数据类型；

拆箱是指将引用数据类型的对象转为基本数据类型。

### 3、Integer包装类

Integer类除了具有Object类的所有方法外，还有一些特有的方法：





在JDK5.0版本之前，数学运算表达式中的操作数必须是基本类型的，并且运行结果也是基本类型，包装类和基本类型是不允许进行混合数学运算的。

在JDK5.0的版本中提供了自动拆箱和装箱技术，也就是可以自动进行基本数据类型和包装类对象之间的转换。

# 集合类

## 概述

### 1、为什么引入集合

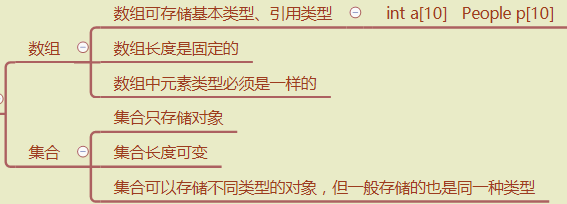
int a ——》 int[] a

People p——》People[] p——》集合

集合类似数组，方便对对象的操作

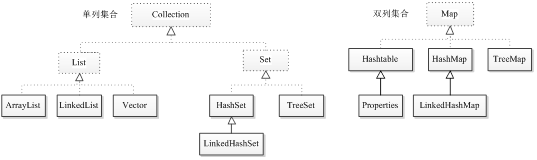
集合类就像一个容器，专门用来存储Java类的对象

### 2、集合VS 数组



### 3、集合的继承体系

由于需求不同，Java就提供了不同的集合类。这多个集合类的数据结构不同，但是它们都是要提供存储和遍历功能



## Collection接口

Collection中定义了单列集合(List和Set)通用的一些方法，这些方法可用于操作所有的单列集合：



迭代器：是集合的获取元素的方式

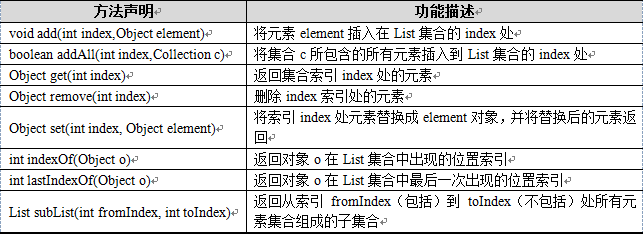
## List接口

### 1、概述

List的特点是**元素有序、元素可重复**

在List集合中允许出现重复的元素，所有的元素是以一种线性方式进行存储的，在程序中可以通过索引来访问集合中的指定元素。另外，List集合还有一个特点就是元素有序，即元素的存入顺序和取出顺序一致。

List不但继承了Collection接口中的全部方法，而且还增加了一些根据元素索引来操作集合的特有方法：



### 2、ArrayList集合

#### 1）特点

最常见的一种集合。

可以将ArrayList集合看作一个长度可变的数组

ArrayList特点：

**底层数据结构是数组**，查询快，增删慢。

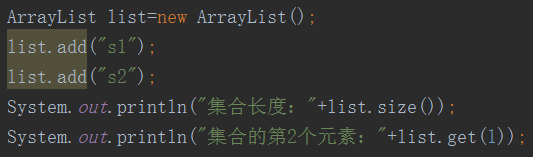
线程不安全，效率高。

（ArrayList集合的底层是使用一个数组来保存元素的，在增加或删除指定位置的元素时，**会导致创建新的数组**，效率比较低，不适合做大量的增删操作）

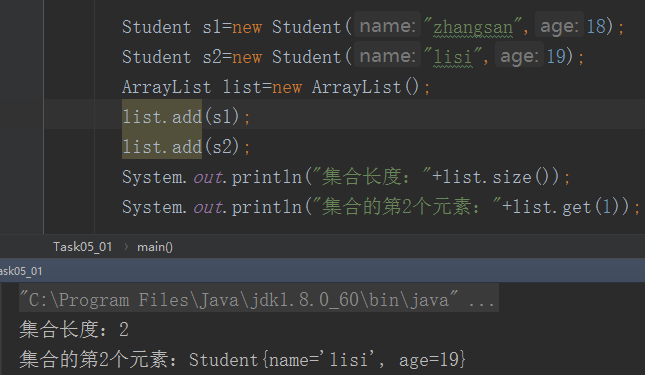
#### 2）存取

#### 练习5-1 ArrayList存取遍历

存储String型对象

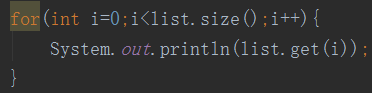


存储Student型对象



#### 3）3种遍历

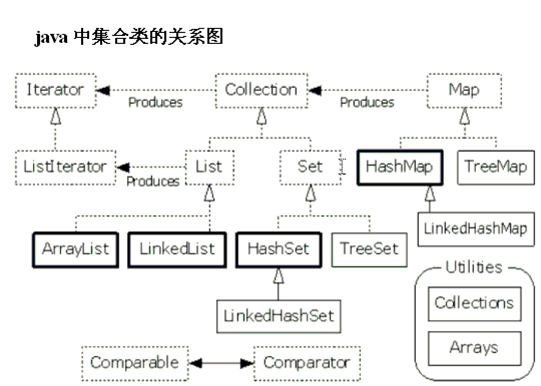
（1）传统for循环语句遍历

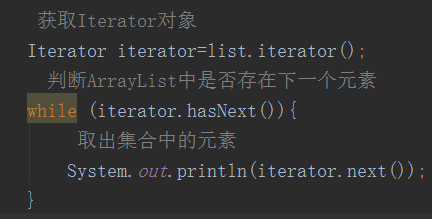


（2）Iterator接口遍历

集合的遍历使用**Iterator接口**

**Collection接口与Map接口主要用于存储元素**，而Iterator主要用于迭代访问（即遍历）Collection中的元素，因此Iterator对象也被称为**迭代器**。



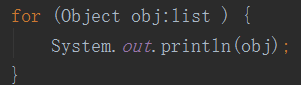


Iterator迭代器对象在遍历集合时，内部采用**指针**的方式来跟踪集合中的元素

当通过迭代器获取ArrayList集合中的元素时，都会将这些元素当做**Object类型**来看待，如果想得到特定类型的元素，则需要进行强制类型转换

（3）foreach语句

foreach循环用于遍历数组或集合中的元素



#### 练习5-2 最后得分

1、提示用户输入裁判数、各裁判的分数。接收用户从键盘输入的内容。

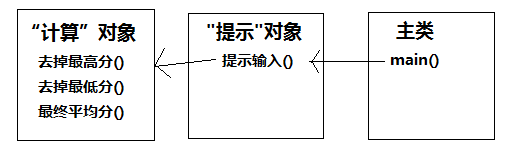
2、将分数存放在ArrayList中。

3、去掉最高分、最低分、算出最后的平均分。

要求：使用面向对象思维来思考问题。

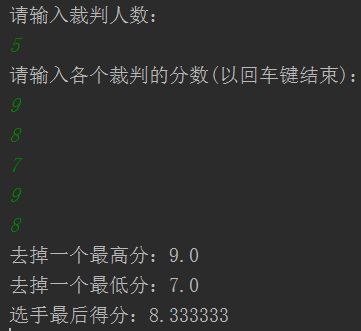
离开main()：也就是不要所有代码都放在一个main()里完成

分析：有几个对象，每个对象负责做什么



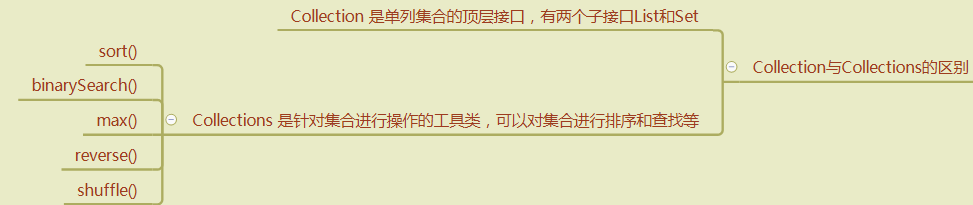
代码：

结果：



提示：

**Collections**是针对集合进行操作的工具类，可以对集合进行排序和查找等



#### 练习5-3 去除重复值

练习1：ArrayList去除集合中字符串的重复值(字符串的内容相同)

// 创建集合对象

ArrayList array = new ArrayList();

// 添加多个字符串元素(包含内容相同的)

array.add("hello");

array.add("world");

array.add("java");

array.add("world");

array.add("java");

array.add("world");

array.add("world");

array.add("world");

array.add("world");

array.add("java");

array.add("world");

练习2：去除集合中自定义对象的重复值(对象的成员变量值都相同)

// 创建学生对象

Student s1 = new Student("林青霞", 27);

Student s2 = new Student("林志玲", 40);

Student s3 = new Student("凤姐", 35);

Student s4 = new Student("芙蓉姐姐", 18);

Student s5 = new Student("翠花", 16);

Student s6 = new Student("林青霞", 27);

Student s7 = new Student("林青霞", 18);

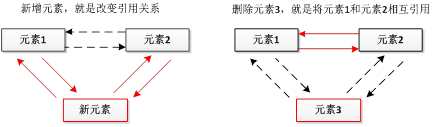
### 3、LinkedList结合

#### 1）特点

**底层数据结构是链表**，查询慢，增删快。

线程不安全，效率高。

该集合内部维护了一个双向循环链表，链表中的每一个元素都使用引用的方式来记住它的前一个元素和后一个元素，从而可以将所有的元素彼此连接起来。当插入一个新元素时，只需要修改元素之间的这种引用关系即可，删除一个节点也是如此。





#### 2）存取遍历

#### 练习5-4 LinkedList存取遍历

创建LinkedList对象，使用add()插入3个String型元素，遍历所有元素；

在第2个元素之后插入一个新元素，在第一个位置插入一个新元素，遍历所有元素，获取第一个元素和最后一个元素；

删除第2个元素，删除第一个元素，遍历所有元素。

### 4、Vector集合

**底层数据结构是数组**，查询快，增删慢。

线程安全，效率低。

### 练习5-5 获取不重复的随机数

模拟随机点名，获取10个1-40之间的随机数，要求不能重复



### 5、泛型

集合可以存储任何类型的对象，但是当把一个对象存入集合后，集合会“忘记”这个对象的类型，将该对象从集合中取出时，这个对象的编译类型就变成了Object类型。

泛型是一种把明确类型的工作推迟到创建对象或者调用方法的时候才去明确的**特殊的类型**。

一般在集合中使用泛型

ArrayList<Student> array = new ArrayList<Student>()

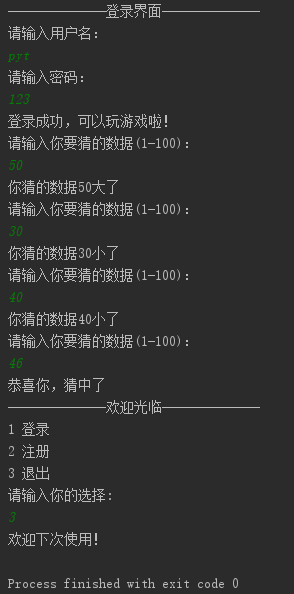
ArrayList<Integer> array = new ArrayList< Integer>()

创建对象集合时指定类型为Student/Integer

使用泛型后，遍历时，就可以指定元素类型 而不是Object。

### 练习5-6 用户登录注册案例



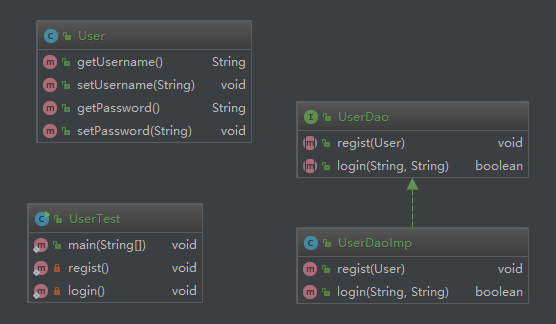


模拟用户注册登录功能，注册信息保存在集合中，登录的用户密码跟集合中比较，登录成功可以进行猜数字游戏。

分析：

代码——》类图：在idea中打开类源码文件，在类的大括号内右键，选择Diagrams->Show Diagrams...，或者使用快捷键Ctrl+Shift+Alt+U

在类图上右键，show Categories可以选择显示属性方法。



### 练习5-7 模拟KTV点歌系统

## Set接口

### 1、特点

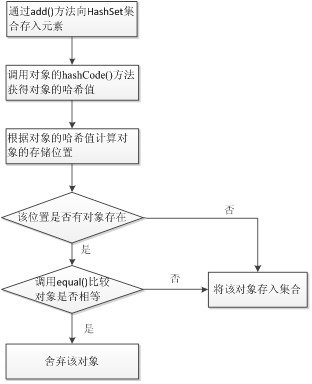
Set的特点是**元素无序，唯一**

### 2、HashSet集合

HashSet是根据对象的哈希值来确定元素在集合中的存储位置，因此具有良好的存取和查找性能

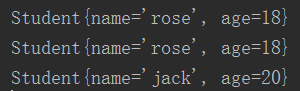
底层数据结构是**哈希表**(是一个元素为链表的数组)

由hashCode()和equals()保证元素唯一 （Alt+Insert自动生成）



#### 练习5-8 HashSet练习

定义3个Student对象（其中有2个对象元素相同），定义一个HashSet对象，往HashSet里加入这3个Student对象，遍历。



为什么会有重复呢？

如何解决？

#### 练习5-9 不重复的随机数

获取10个1至40的随机数，要求随机数不能重复。

用ArrayList实现的时候还要判断集合中是否已经包含该元素，用HashSet则不需要，因为HashSet元素唯一。

#### 练习5-10 模拟新浪微博用户注册

### 3、TreeSet集合

TreeSet是以二叉树的方式来存储元素，它可以实现对集合中的元素进行排序

底层数据结构是红黑树(是一个自平衡的**二叉树**)

保证元素的排序

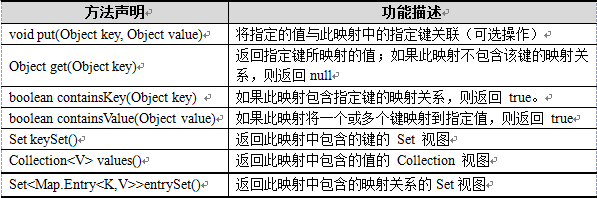


## Map接口

### 1、概述

用于存储具有键(Key)、值(Value)映射关系的元素，每个元素都包含一对键值，在使用Map集合时可以通过指定的Key找到对应的Value

将键映射到值的对象。一个映射不能包含重复的键；每个键最多只能映射到一个值。



### 2、Map和Collection的区别

Map 存储的是键值对形式的元素，**键唯一，值可以重复**。夫妻对

Collection 存储的是单独出现的元素，子接口Set元素唯一，子接口List元素可重复。光棍

### 3、HashMap集合

HashMap：是基于哈希表的Map接口实现

哈希表的作用是用来保证键的唯一性的

**遍历方式：**

**1）键找值**

获取所有键的集合

遍历键的集合,得到每一个键

根据键到集合中去找值

**2）键值对对象找键和值**

获取所有的键值对对象的集合

遍历键值对对象的集合，获取每一个键值对对象

根据键值对对象去获取键和值

#### 练习5-11 HashMap遍历练习

1、创建HashMap<Integer,String>对象，添加元素（键值对），遍历输出所有元素

/\* 遍历方式1：键找值  
 \* 1）获取所有键的集合  
\* 2）遍历键的集合,得到每一个键  
\* 3）根据键到集合中去找值 \*/

/\*遍历方式2：键值对对象找键和值  
 1）获取所有的键值对对象的集合  
2）遍历键值对对象的集合，获取每一个键值对对象  
3）根据键值对对象去获取键和值\*/

2、创建HashMap<String,Student>对象，添加元素，遍历输出

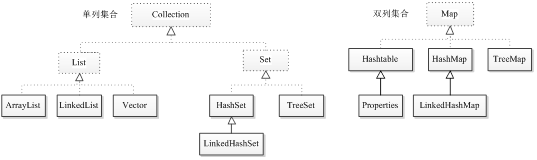
### 4、Properties集合

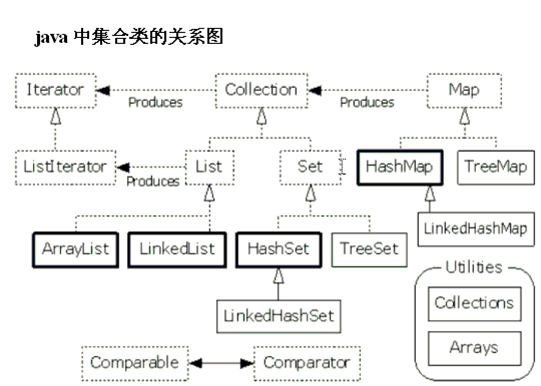
Properties主要用来存储字符串类型的键和值，在实际开发中，经常使用Properties集合来存取应用的配置项。



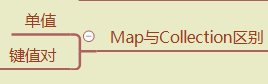
## 小结

### 1、集合的继承体系





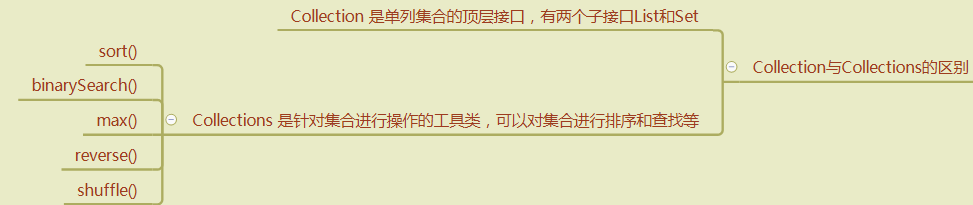
### 2、Collection与Map







### 3、Collection与Collections



# IO（输入输出）

## 概述

Java代码写的数据存储在集合中，是保存在内存还是硬盘？

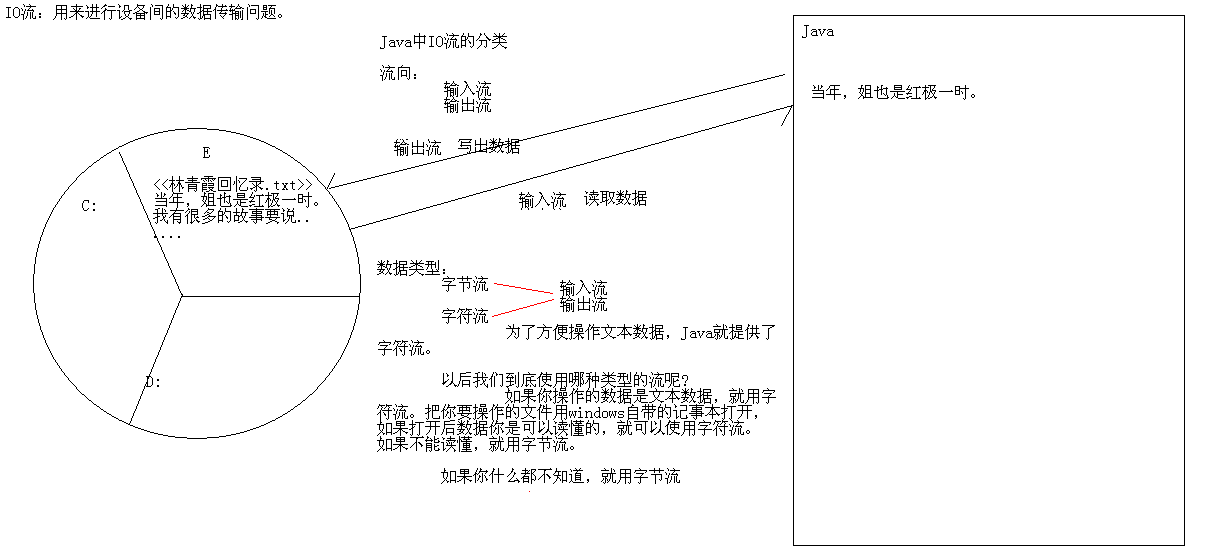
如果要保存在磁盘上，则可以通过输入输出流来实现。

通过不同输入输出设备(键盘，内存，显示器，网络等)之间的数据传输抽象表述为“流”。

**IO用于在设备间进行数据传输的操作**

Java对数据的操作是通过流的方式





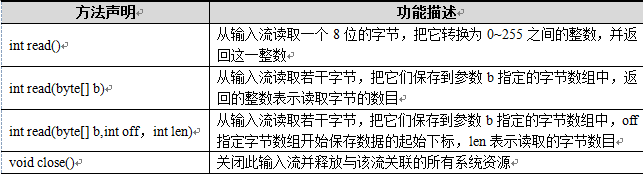
## 字节流

### 1、概述

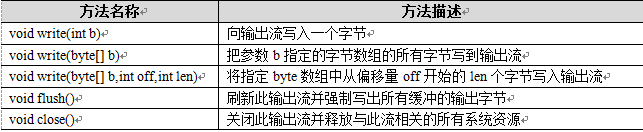
在计算机中，无论是文本、图片、音频还是视频，所有文件都是以二进制(字节)形式存在的，IO流中针对字节的输入输出提供了一系列的流，统称为字节流。

字节流是程序中最常用的流，根据数据的传输方向可将其分为字节输入流和字节输出流。JDK提供了两个抽象类InputStream和OutputStream。

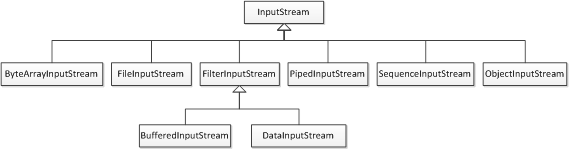
InputStream的常用方法：



OutputStream提供了一些与写数据有关的方法：



抽象类不能被实例化，具体子类如下：





### 2、FileInputStream & FileOutputStream

字节输入输出流

在操作文件时，最常见的就是从文件中读取数据并将数据写入文件，即文件的读写

#### 1）读写文件

#### 练习6-1 字节流读写文件

用Java代码实现，在当前目录创建一个example.txt文件，往example.txt文件写入“我爱Java！”，在磁盘上查看是否写入成功。

然后通过Java代码查看example.txt文件内容。

继续往该文件追加内容“我真的爱Java”，并查看文件内容。

/\*FileOutputStream写文件步骤：  
\* 1）创建字节输出流对象  
\* 构造方法FileOutputStream("fos.txt")写入新数据，会覆盖原有数据  
\* 构造方法FileOutputStream("fos.txt"，true)文件内容之后追加新内容;  
\* 2）调用write()方法写数据  
\* write(int b):写一个字节  
\* write(byte[] b):写一个字节数组  
\* 3）调用close()方法释放资源 \*/  
  
/\* FileInputStream读取文件步骤：  
\* 1）创建字节输入流对象  
\* 2）调用read()方法读取数据  
\* int read():一次读取一个字节  
\* int read(byte[] b):一次读取一个字节数组  
\* 3）调用close()方法释放资源\*/

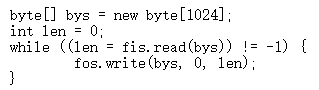
**加入异常处理，完整版**

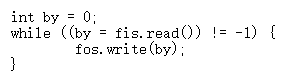
#### 2）文件的拷贝

文件的拷贝需要通过输入流来读取文件中的数据，通过输出流将数据写入文件

#### 练习6-2 拷贝文本/图片/mp4文件

基本字节流一次读写一个字节数组/ 基本字节流一次读写一个字节





1、将当前文件example.txt拷贝到D盘根目录

2、将D盘图片拷贝到当前目录

3、将D盘mp4拷贝到当前目录

### 3、BufferedInputStream & BuffereOutputStream

#### 1）为什么引入字节缓冲流

一个字节一个字节的读写，需要频繁的操作文件，效率非常低，通过一次读写一个字节数组的方式会更好，那么Java就专门提供了带缓冲区的字节类——字节缓冲流。

字节缓冲区流仅仅提供缓冲区，为高效而设计的；

真正的读写操作还得靠基本的流对象实现



缓冲区的大小也是可以设置的（构造方法），但一般用默认大小就可以。

#### 练习6-3字节缓冲流读写文件

创建字节缓冲流对象，通过缓冲流对象实现信息写入文件，以及读取文件内容

一次读取一个字节 VS 一次读取一个字节数组

注意：创建字节缓冲流对象，需要传入基本字节流。因为字节缓冲区流仅仅提供缓冲区，为高效而设计的；真正的读写操作还得靠基本的流对象实现

#### 练习6-4字节缓冲流实现文件的拷贝

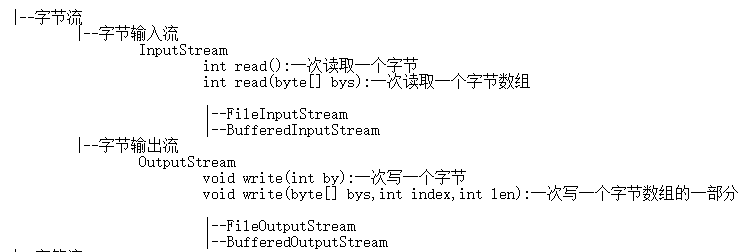
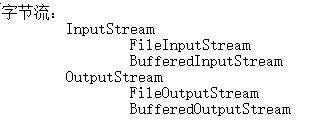
从缓冲区一次读写一个字节，从缓冲区一次读写一个字节数组

1、将当前文件1.txt拷贝到D盘根目录

2、将当前目录图片1.png拷贝到D盘根目录

3、将当前目录1.mp4拷贝到D盘根努力

### 4、字节流小结



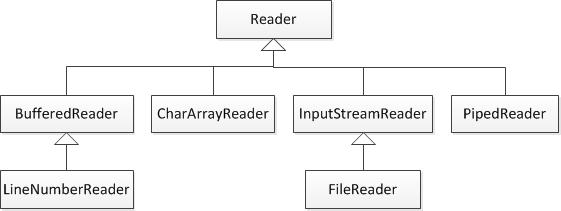


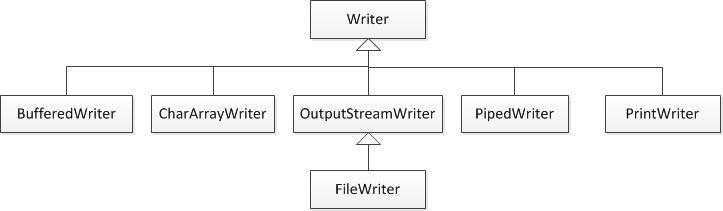
**推荐使用“字节缓冲流一次读写一个字符数组”**

## 字符流

### 1、概述

InputStream类和OutputStream类在读写文件时操作的都是字节，如果希望在程序中操作字符呢？JDK提供了字符流。





### 2、FileReader & FileWriter

基本字符流

一次读写一个字符、一次读写一个字符数组

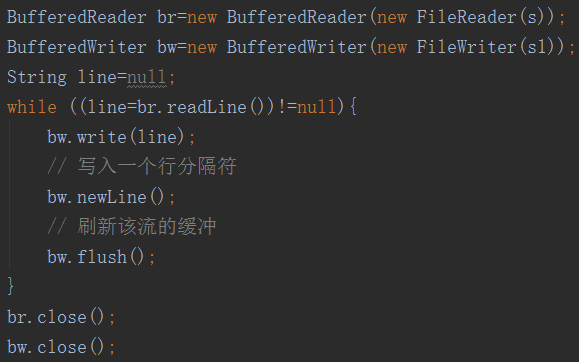
### 3、BufferedReader & BufferedWriter

字符缓冲流

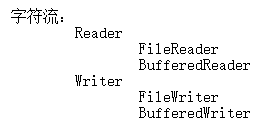
一次读写一个字符、一次读写一个字符数组、一次读写一个文本行

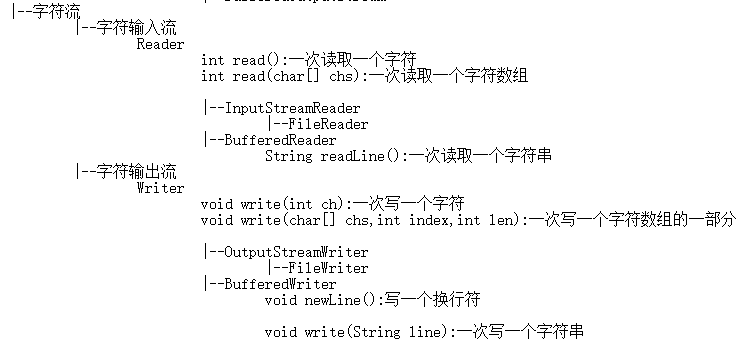
#### 练习6-5 基本字符流/字符缓冲流读写文件

5种方法



### 4、字符流小结





**推荐使用“字符缓冲流一次写一行文本”**

该如何选择用字节流还是字符流呢？？

复制数据，如果我们知道用记事本打开并能够读懂，就用字符流，否则用字节流。所以拷贝文件可以用字符流，拷贝图片mp4可以用字节流。

### 练习 6-6 集合与文本文件之间的操作

练习1：把ArrayList集合中的字符串数据存储到文本文件

练习2：从文本文件中读取数据(每一行为一个字符串数据)到集合中，并遍历集合

练习3：有一个文本文件中存储了几个名称，请实现随机获取一个人的名字

### 练习6-7 登录注册IO版

将用户注册时输入的用户信息保存在文件。

只要增加一个实现类即可：UserDaoImpIO implements UserDao

注册时的信息有用户名和密码，将它们组合成一条记录保存在文件的一行，可以自定义规则，比如：用户名=密码 或者 用户名&密码 都可以

登录时，比较用户名和密码，需要将文件读取的一行数据根据符号拆分成2个，分别比较用户名和密码是否相等。

### 练习6-8 保存书店每日交易记录

## File类

# JDBC

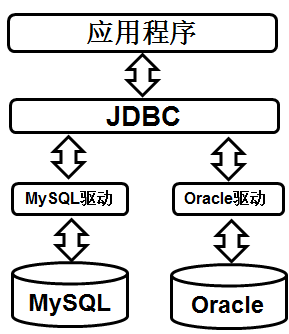
## 概述

### 1、什么是JDBC

JDBC的全称是Java数据库连接（Java Database Connectivity），它是一套用于执行SQL语句的Java API。

应用程序可通过这套API连接到关系型数据库，并使用SQL语句来完成对数据库中数据的查询、新增、更新和删除等操作。

### 2、应用程序如何访问数据库



JDBC是应用程序与数据库之间的桥梁

## JDBC常用API

### 1、DriverManager类

用于加载JDBC驱动并且创建与数据库的连接

### 2、Connection接口

代表Java程序和数据库的连接，只有获得该连接对象后，才能访问数据库，并操作数据表。

### 3、Statement接口

用于执行静态的SQL语句，并返回一个结果对象

### 4、PreparedStatement

是Statement的子接口，用于执行预编译的SQL语句

### 5、ResultSet接口

用于保存JDBC执行查询时返回的结果集，该结果集封装在一个逻辑表格中。

## 第一个JDBC程序

### 1、步骤

（1）加载并注册数据库驱动。

（2）通过DriverManager获取数据库连接。

（3）通过Connection对象获取Statement对象。

（4）使用Statement执行SQL语句。

（5）操作ResultSet结果集。

（6）关闭连接，释放资源。

### 2、实现

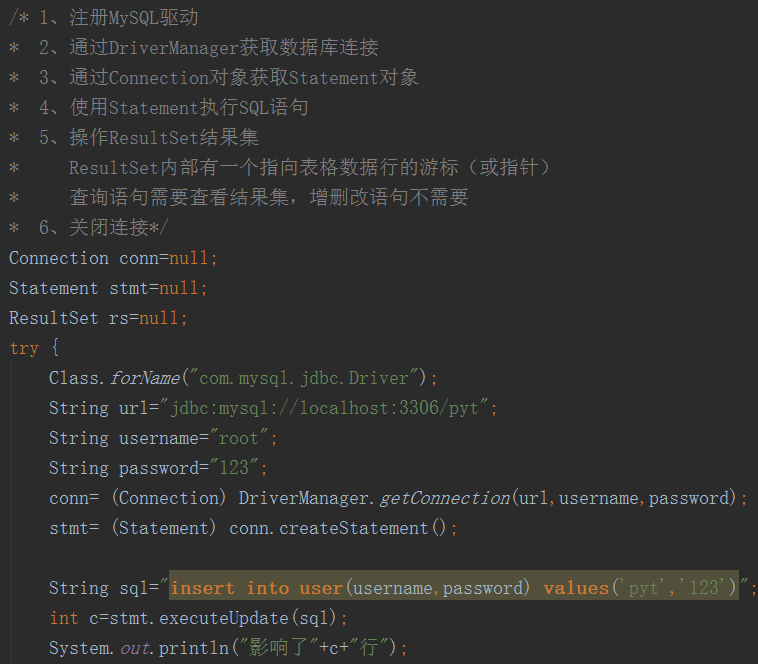
#### 练习7-1 通过JDBC访问MySQL

1）导入jar包

File——Project Structure——Modules——Dependencies——“+”选择jar包

2）代码实现

**增删改功能**



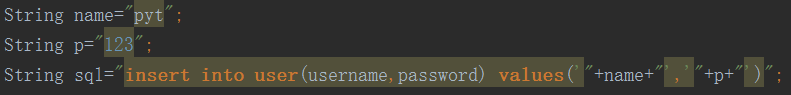
**查询功能**

（比增删改多了第5步）

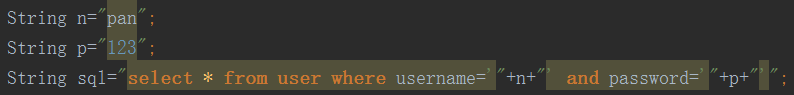


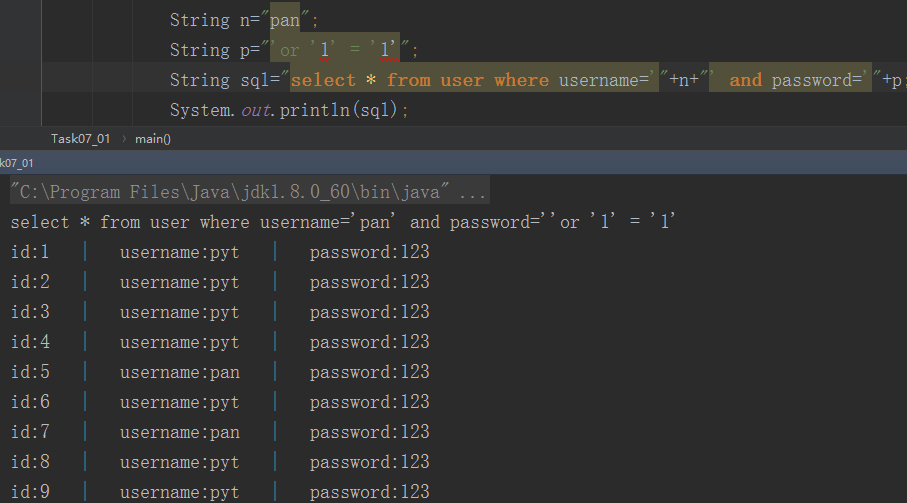
### 3、问题

如果SQL语句中某些值是传入的呢



代码的可读性和可维护性较差





代码的安全性较差

另外，**Statement对象每次执行SQL语句时，都会对其进行编译**。当相同的SQL语句执行多次时，Statement对象就会使数据库频繁编译相同的SQL语句，从而降低数据库的访问效率。

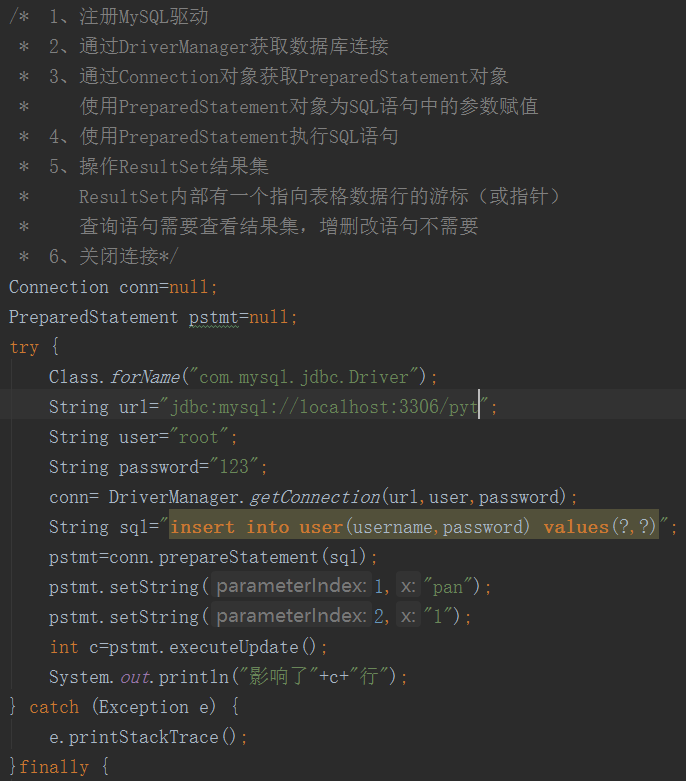
代码的执行效率较差

### 4、改进

PreparedStatement对象可以对SQL语句进行**预编译**，预编译的信息会存储在该对象中。当相同的SQL语句再次执行时，程序会使用PreparedStatement对象中的数据，而不需要对SQL语句再次编译去查询数据库，这样就大大的提高了数据的访问效率。

#### 练习7-2 使用PreparedStatement提高效率和安全

第3步增加了：使用PreparedStatement对象为SQL语句中的参数赋值。除了setString()方法还有setInt()等，要跟数据库中的字段类型对应。



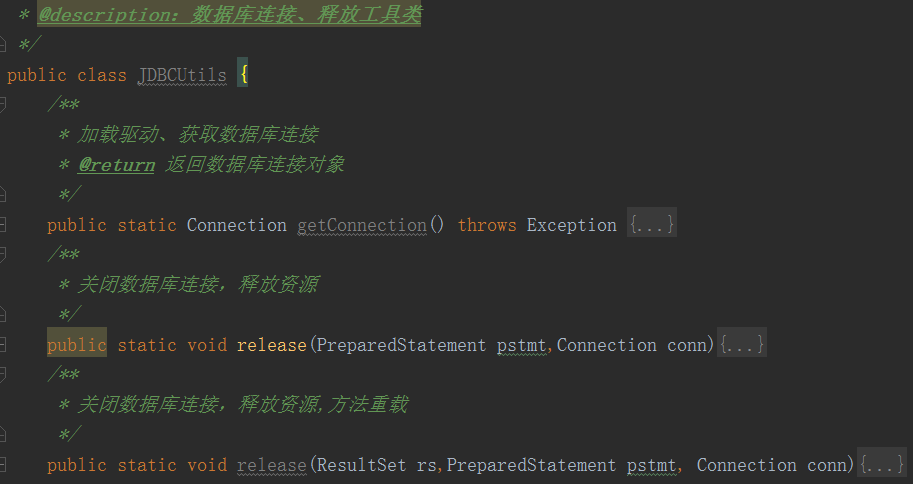
1、插入几条语句后，练习更新语句和删除语句（步骤重新写吧！）

2、练习查询语句

### 练习7-3 登录注册DB版

一些重复的代码抽取出来作为工具类

#### 1、新建JDBCUtils工具类



#### 2、新建UserDaoImpDB用户操作的具体实现类（DB版）

class UserDaoImpDB implements UserDao

#### 3、测试类，使用UserDaoImpDB多态

// 多态，建立用户操作对象  
private static UserDao *userDao*=new UserDaoImpDB();

# 综合实训