**Java语言为什么流行起来**

* 外部环境

-互联网的爆发式发展

-互联网上的电脑硬件不同，软件环境差异较大，需要一个跨平台的语言

* Java核心优势：跨平台

**Java各版本和体系架构**

* J2EE （JavaEE）

-Java2 Enterprise Edition

-定位在服务器端的应用

* **J2SE （JavaSE）**

-Java2 Standard Edition

-定位在个人计算机上的应用

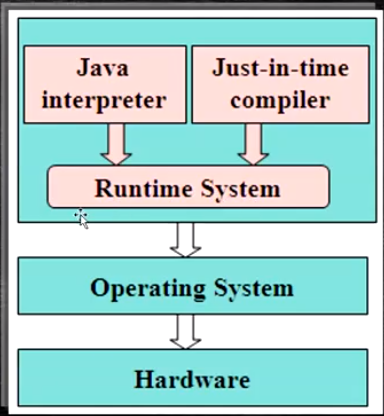
* J2ME （JavaME）

-Java2 Micro Edition

-定位在消费性电子产品的应用上

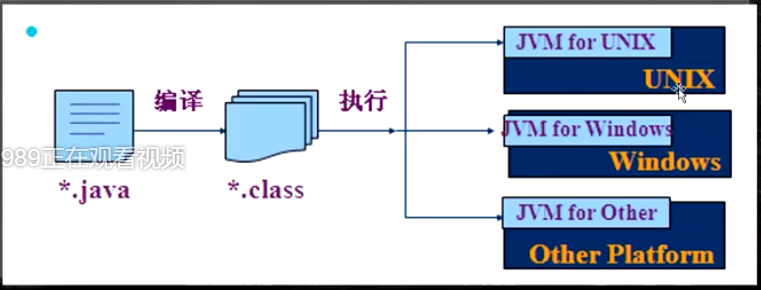
**JVM(Java Virtual Machine)**

* JVM是一种规范，就是一个虚拟的用于执行bytecodes字节码的计算机
* Java虚拟机是Java最核心技术，也是跨平台的基础
* 可以使用软件来实现，IBM，SUN，BEA等。你也可以自己写一个JVM
* 可以使用硬件来实现，比如sun/intel公司正在研发的java芯片



**JVM和Java跨平台特性**

* 通过编译器将Java源程序编译成Java字节码文件（.class）（字节码文件采用结构中立的中间文件格式）
* 通过解释器将Java字节码文件解释为对应机器语言并运行



* Java和C跨平台原理的比较

-Java：一次编译，到处运行。前提是不同的OS里要装不同版本的JVM

-C：多次编译，到处运行

JDK：Java Development Kit Java开发工具集

JRE：Java Runtime Environment Java运行环境

**Java的特点**

* 跨平台
* 简单，相比于C和C++来说简单。它去除了指针和内存管理
* 安全
* 完全面向对象
* 健壮，发展迅速

**开始Java开发之前的准备**

* **安装JDK**

卸载：

1. 控制面板→程序和功能→Java（TM）SE Development Kit 卸载

2. 此电脑右键→属性→高级系统设置→环境变量→系统变量里JAVA\_HOME删除，Path里的Java\_Home相关的也删除

3. cmd出来再输入java -version发现显示‘java’不是内部或外部命令，也不是可运行的程序或批处理文件的时候说明已经卸载干净了

安装：

1. 双击jdk-12.0.2\_windows-x64\_bin.exe进入安装

2. 安装到C:\Program Files\Java\jdk-12.0.2\

3. cmd里输入

cd ../.. cd “Program Files” cd Java cd jdk-12.0.2

dir 发现里面没有JRE文件夹，这时敲下面的命令可以自动在jdk-12.0.2文件夹里生成

jre文件夹

bin\jlink.exe --module-path jmods --add-modules java.desktop --output jre

发现报错了

关闭cmd窗口，在查找栏查找Windows系统里的命令提示符，并右键，更多，以管理员身份运行，再将刚才所有命令敲一遍，再进入到jdk-12.0.2文件夹里发现生成了jre文件夹

这时在cmd窗口输入java -version还是显示‘java’不是内部或外部命令

因为cmd它需要去环境变量里找相应的命令。如果有此命令就执行，没有就会报错。所以需要配置环境变量

4. 在此电脑右键 → 属性 → 高级系统设置 → 环境变量 → 系统变量处新建 变量名：JAVA\_HOME 变量值：浏览目录一直选到jdk-12.0.2文件夹处，确定 → Path → 新建 → %JAVA\_HOME%\bin → 新建 → %JAVA\_HOME%\jre\bin ,三个确定

重新启动cmd，再输入java -version就会显示相应的版本了，这样JAVA就安装成功了

如果电脑里安装了多个版本的JDK，想要切换它们在此电脑右键 → 属性 → 高级系统设置 → 环境变量 → 修改系统变量处的JAVA\_HOME的目录地点就可以

* **安装Java开发工具**

常用Java开发工具

-文本编辑器（任选一个）

UltraEdit，EditPlus，notepad++

-集成开发环境（IDE：Integrated Development Environment）

Eclipse（http://www.eclipse.org）解压即可

MyEclipse：若进行J2EE开发，还要加MyEclipse插件。MyEclipse是Eclipse的插件，

也是一款功能强大的J2EE集成开发环境。现在直接安装集成Eclipse的MyEclipse即

可。商业软件

IntelliJ IDEA：目前最火

* 准备JDK API
* 掌握简单DOS命令

dir directory 查看当前目录下的文件夹和子文件夹

cd change directory 改变当前文件夹

md make directory 创建当前文件夹

rd remove directory 删除当前文件夹

cls clear screen 清除屏幕

切换盘符 d：c：

Tab键 帮助自动完成文件名或文件夹

上下箭头键 显示之前或者之后的命令

**写JAVA**

新建一个txt文件，文件名比如叫HelloWorld，后缀名改成 .java 。

在文件上右键，用Notepad++打开

输入代码，写好保存

public class HelloWorld{

public static void main(String[] args){

**S**ystem.out.println("欢迎大家学习java");

}

}

在cmd画面进入到文件所放的路径，javac <文件名.java>，回车执行

在和该文件相同的文件夹内生成了一个以.class为后缀名的文件

在cmd画面输入java HelloWorld，发现回显是乱码

去文件中先把代码全部复制，菜单栏选编码，选择使用ANSI编码，再将刚才复制的代码将当前的代码覆盖，保存

在cmd画面输入java HelloWorld，发现回显显示欢迎大家学习java

**❀注意：**

1.java文件的名称必须跟public class的名称保持一致

比如这个文件叫做helloworld.java，那么public class的名称也必须为helloworld

2.一个java文件里可以包含多个class ，但是public class只能有一个。每个class都会生成属于它自己的.class文件

3. public static void main(String[] args) 是所有java程序的入口，如果想执行对应的java代码的话，则必须要添加这个方法，且格式是固定的

4.main方法中参数列表可以支持多种写法：String[] args, String [] args, String args[]

5.main方法中参数的名称无所谓，但是一般写成args

6.java代码在编写的时候每行结尾的时候需要使用；结束，为英文的；

7.java代码的代码块需要使用{}括起来，前后匹配。（）也要成对出现

8.一定要有缩进

9.见名知意

10.注释(注释中的内容此处用aaa表示)。Java编译器编译时会跳过注释语句

分类：单行注释：/aaa/

多行注释：/\*aaa\*/

文档注释：/\*\*

\*aaa

\*aaa

\*/

11.反编译工具

编译：源代码 → class

反编译：class → 源代码

反编译软件 jd-gui.exe

因为编译的时候不会对注释进行处理，所以反编译时不可能得到注释

什么时候会用到反编译软件？

比如在源代码丢失，只有jar包的情况下，就可以将jar包反编译，然后将源代码做修改

之后再打成jar包再运行

**Java 数据类型和运算符**

* 标识符Identifier

即起名字

-作用：常量，变量，方法，类和包等的名称

-命名规则：

* 硬性规定

必须以字母，下划线，美元符开头

其他部分可以使字母，下划线，美元符和数字的任意组合

大小写敏感，长度无限制

不可以是java的关键字（这些关键字是留给系统使用的表示特殊含义的字符串）

* 非硬性规定

驼峰标识：

类名，接口名在命名的时候是几个单词拼在一起的要首字母大写

如：public class IdentifiedDemo

方法，变量命名的时候首字母要小写

如：int getValueById

多个单词拼接标识一个标识符的时候，每个单词的首字母都大写

见名知义：通过标识符名称能知道代表的含义，千万不要写拼音

-注意：

java不采用通常语言使用的ASCII字符集

java采用**unicode**这样的标准的国际字符集。因此，这里的字母的含义：英文，汉字等等。（不建议使用汉字来定义标识符）

**字符集简介**

ASCII 英文字符集 1个字节（即8位表示）

ISO8859-1 西欧字符集 1个字节

GB2312 大陆使用最早，最广的简体中文字符集 2个字节

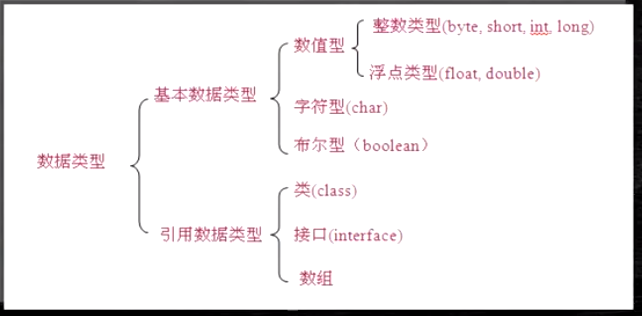
**Unicode** 国际通用字符集 **2**个字节 （有UTF-8和UTF-16）

**Java基本数据类型**

Java是一种强类型语言

-常量是有数据类型的

-变量都必须声明其数据类型



弱类型表示变量会根据值自己去推断，不需要指定类型是什么。比如Python

**基本数据类型（4类8种）**

* 整数类型：byte（使用1个字节存储，因此范围为-128~127）

short （使用2个字节存储，因此范围为-32768~32767）

**int** （使用**4个字节**存储，因此范围为+-21亿）

long （使用8个字节存储，因此范围为超大）

（不同类型表示不同长度）

整型常量：十进制整数，如99，-500，0

八进制整数，要求以0开头，如015

十六进制数，要求0x或0X开头，如0x15

整型常量**默认为int型**，声明long型常量可以后加“l”或“L”

如long a=55555555; //不出错，在int表示的范围内

long b=55555555555；//不加L出错，已经超过int表示的范围

* 浮点类型：float （单精度，尾数可以精确到7位有效数字，占用**4字节**，表示范围为

-3.403E38~3.403E38）

double（双精度，精度是float类型的两倍，绝部分应用程序都采用这个。

表示范围为-1.798E308~1.798E308）

浮点类型常量：十进制数形式，如3.14 314.0 0.314

科学记数法形式，如

314e2 即 314\*10^2 结果为31400.0

314E-2 即 314\*10^(-2) 结果为3.14

注意：浮点常量默认为double。要变为float，需在后面增加F/f，如3.14F

浮点数存在舍入误差，很多数字不能精确表示。如果需要进行不产生舍入误

差的精确数字计算，需要使用BigDecimal类

* 字符类型：**char**

占用**2个字节**，使用的时候使用单引号’ ’表示，如char a = ‘A’

字符串使用双引号” ”表示，如string s = “A”

这里的两个A是不同的意思。char的A表示一个字符，而string的A表示一个字符串，即字符序列，也就是多个字符组合起来称为的东西

char类型用来表示在Unicode编码表中的字符。char类型在内存中存储的是该字符的Unicode编码值，所以char类型可以当作int类型来处理

如定义char a = ‘A’ ，System.out.println(a==65)，返回值为true。因为在Unicode中，A对应的数为65

Java语言中还允许使用转义字符\来将其后面的字符转变为其他的含义

如char c2 = ‘\n’ 代表换行符

如char c3 = ‘\t’ 代表制表符（tab）

* 布尔类型：**boolean**，只有true和false两值，在存储的时候占**1位**

**引用数据类型**

* 类
* 接口
* 数组

**常量和变量**

* 常量：在程序运行过程中，值不会发生改变的量
* 变量：在程序运行过程中，值会发生改变的量

变量中要有变量名称，变量类型，作用域

1. 先声明，后赋值

int a ; 声明

a = 10 ; 赋值

1. 声明+赋值

int b =20 ;

注意：在类内，方法外定义的变量叫成员变量，会存在默认值

在方法内定义的变量必须要进行初始化操作，不会存在默认值

如：Public class ConstantAndVar{

static int **d**;

public static void main(String[] args){

int **c**;

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

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

}

}

打印c会出错，因为它没有初始值，打印d会出现0，为它的初始值

在一行中可以定义多个变量，但是不推荐，每个变量最好占一行

给变量赋值过程中，变量的值成为常量。即int a = 10，a为变量，10为常量

使用final关键字修饰的变量称之为常量或者叫做最终常量，表示不可修改

如： a = 10;

System.out.println(a);

a = 30;

System.out.println(a);

final int c = 20;

System.out.println(c);

c = 30;

System.out.println(c);

这时会报错，说c = 30那行错误，因为无法为最终变量c分配值。因为final int c=20已经给它分配一个不可修改的常量了。如果把这行和下一行注释上，会显示结果 10，30，20

**运算符**

* Java语言支持如下运算符：

-算术运算符：+，-，\*，/，%，++，--

++：表示在变量原有的基础上+1，谁在前，先运算谁

int a = 1;

int b = 2;

System.out.println(a++); 结果为1，直接打印出a的结果后+1

System.out.println(++a); 结果为3，先算1+a，再打印出a+1的结果

System.out.println(++b+b++); 结果为6，++的优先级先于+。++b为3，这时b

的值为3，因此++b+b就为3+3=6，而后面的++这次结果体现不出来

这时System.out.println(b); 结果为4 上一次的++b时，b的值由2变成了3，

b++的时候，3+1，b的值变为4，因此结果为4

--：表示在变量原有的基础上-1，谁在前，先运算谁

-赋值运算符：=

java中=表示赋值运算符，==表示相等的操作

-扩展赋值运算符：+=，-=，\*=，/=

c = c+2 和c += 2 表达的意思一样

但是假如现在定义一个byte d=10；d=d+1的话会报错，因为d+1整体会变成int类型。 需要将int类型的值转换成byte类型，会有精度上的损耗，只能进行强转。但是如果byte d=10；d+=1的话就不会报错。因此建议在操作时使用扩展赋值运算符

-关系运算符：>, < , >= , <= , ==，!=

用来比较两值的关系，返回的值是布尔类型，也就是只有true和false两种情况

-逻辑运算符：&&，||， ！

逻辑运算符一般两边的值不是具体的值，而是一个表达式

&&：表示短路与。两边表达式从左向右开始对比，如果左边的表达是false，右边不需要进行判断。即两边表达式中只要有一个是false，整体结果就是false

&：与运算符，但是两边都会参与运算

System.out.println(3>5 && 3<4); 结果为false

||：表示短路或，两边表达式从左向右开始对比，如果左边的表达是true，右边不需要进行判断。即两边表达式中只要有一个是true，整体结果就是true

|：或运算符，但是两边都会参与运算

System.out.println(3>5 || 3<4); 结果为true

！：取反。如果是true，取反为false。如果是false，取反为true

System.out.println(!(3<4)); 结果为false

-位运算符：&，|，^，~，>>, <<

只能操作数值，操作的时候会转成二进制进行运算

&与运算，只要有0就为0，只有全为1才为1

4：100，5：101

System.out.println(4 & 5); 100&101=100为4

|或运算，只要有一个为1，即为1

System.out.println(4 | 5); 100|101=101为5

^异或运算，相异为1，相同为0

System.out.println(4 ^ 5); 100^101=001 为1

<<,>>位移运算

System.out.println(2<<3); 结果为16。2：0010，左移3位变成10000，为16

即21\*23=24=16

System.out.println(8<<3); 结果为64。23\*23=26=64

~取反

System.out.println(~4)); 结果为-5

4：0000 0100 取反：1111 1011，这是一个负数，是一个补码，要变为原

码需要先-1，变为1111 1010，再按位取反变为10000101，为-5

★↑~取反的补充讲解★

（123.45）10= 1\*102+2\*101+3\*100+4\*10-1+5\*10-2

（10101.11）2= 1\*24+1\*22+1\*20+1\*2-1+1\*2-2

（375.4）8= 3\*82+7\*81+5\*80+4\*8-1

原码：是指将最高位作为符号位（0表示正，1表示负），其他数字位代表数值本身的绝

对值的数字表示方式。数字6在计算机原码表示为0000 0110，数字-6在计算机

中原码表示为1000 0110

以上是在8位计算机中的原码表示，如果在32位或16位计算机中，表示方法

是一样的，只是多了几个0而已。比如在16位计算机中表示-6的话就是这样

1000 0000 0000 0110

反码：反码表示规则为如果是正数，则表示方法和原码一样，如果是负数，则保留符号

位1，然后将这个数字的原码按照每位取反，则得到这个数的反码表现形式

如数字6在计算机中反码就是它的原码：0000 0110

而数字-6在计算机中反码为：1111 1001

补码：在计算机中0是有正负之分的。负数的补码是对反码+1，而正数不变，**正数的原**

**码反码补码都是一样的**，在补码中用（-128）代替了（-0），所以补码的表示范

围为（-128~0~127），共256个。补码是计算机表示数据的一般方式，其规则为：

如果是正数，则表示方法和原码一样；**如果是负数，则将数字的反码+1（相当于**

**将原码数值按位取反后在对低位+1）**

❀注意：（-128）没有相对应的原码和反码，（-128）=（10000000）的补码的加减运算如下：(1)10-(1)10=(1)10+(-1)10=(0)10

(00000001)补+(11111111)补=(00000000)补=（0）

(1)10-(2)10=(1)10+(-2)10=(-1)10

(00000001)补+(11111110)补=(11111111)补=（-1）

这里(11111111)补要换成原码，要先-1，变成11111110，再取反变成10000001，

即为-1

机器数和真值：

实际运算中，数是有正负的，计算机中数也有正负，通过用一个数的最高位表示符

号，如果字长为8位，分别为D7~D0，那么D7为符号位，0表示正数，1表示负

数，D6~D0为数值位。例如11010111=-87 。第一位为1即表示负数，后面的

1010111=87，因此为-87。这样，在计算机中，连同符号一起数码化的数被称为机

器数，如上面的11010111；而使用正负号加其绝对值的表示方法称为该数的真值，

如-87

-条件运算符/三目运算符 ?:

使用的时候需要跟一个表达式，表达式如果为true则返回**?**后的结果。如果是false则返回**:**后的结果

System.out.println(3>2?3:2); 结果为3.

System.out.println(true**?**false**:**true**?**false**:**true); 结果为false

表达式为true，因此看?后面的内容，?后面的内容为false，再:后面的不用看了

System.out.println(false**?**false**:**true**?**false**:**true); 结果为false

表达式为false，所以应该看：后面的内容，冒号后面为true**?**false**:**true这个整体。

这个整体里表达式为true,看?后面的部分，为false，所以最后结果为false

运算优先级：

赋值 < 三目 < 逻辑 < 关系 < 算术 < 单目

* 相关概念辨析

-+运算符 操作符 Operator

-5+6 表达式 expression

-5 6 操作数 Operand

-int m=5+6；语句 sentence

**基本数据类型之间的转换**

在赋值运算或算术运算时，要求数据类型相同，否则要进行类型转换。除布尔类型外，所有的基本数据类型（包括整型 ，浮点型，字符型）因为各自的精度不同，赋值时都要考虑这个问题

转换方式：

-自动转换（隐形转换）

在运算过程中如果两个值的类型不一致会**自动将小的类型转换为大的类型**

byte b = 10;

int a = 20;

byte c;

c = a+b; 会显示c = a+b;不兼容的类型：从int转换到byte可能会有损失

byte b = 10;

int a = 20;

int c;

c = a+b; 这时不会报错，因为c(和)为int类型，将b的byte类型转换成了int类型

-强制转换，会发生精度损失，结果可能不准确

byte b = 10;

int a = 20;

byte c;

c = (byte) (a+b); 这时如果就是想转换成byte类型，就←这样**手动强制转换**，将大

的类型转换成小的类型

❀注意：

强制转换时当心丢失数据或失真。当一种类型强制转换成另一种类型，而且又超出

了目标类型的表示范围，就会被截断称为一个完全不同的值，如↓

int型（32位） 00000000 00000000 00000001 00001110 270

byte型（8位） 00001110 14

byte b = 10;

int a = 200;

byte c;

c = (byte)(a+b);

System.out.println(c);

int型（32位） 00000000 00000000 00000000 11010010 210

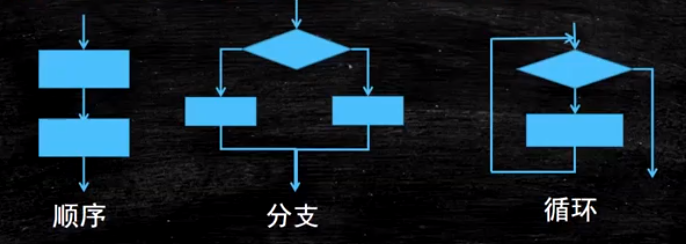
byte型（8位） 11010010

11010010为负数，是一个补码，要-1（变为11010001）再取反（变为10101110）

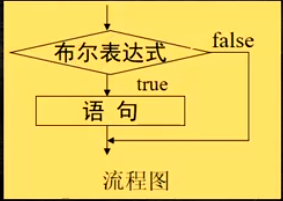
才能得到原码，即为-46

**Java流程控制语句**

流程控制语句是用来控制程序中各语句执行顺序的语句，可以把语句组合成能完成一定功能的小逻辑模块。其流程控制方式采用结构化程序设计中规定的三种基本流程结构，即：顺序结构，分支结构和循环结构，如下图：



* 分支结构
* 单分支结构：if语句对条件表达式进行一次测试，若测试为真，则执行下面的语句，否则跳过该语句



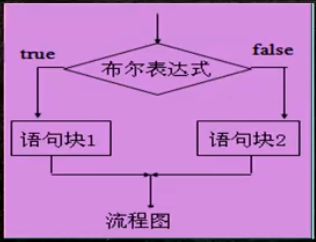
Math.random()产生数据的范围是左闭右开[ 0,1)，即包含0，不包含1

想要得到0-5之间的随机数：

**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **int** i = (**int**)(Math.*random*()\*6);  
 **if**(i>3){  
 System.***out***.println(i+**"大于3"**);  
 }  
 System.***out***.println(**"数字为："**+i);  
 }  
}

Math.random()返回的是double类型的值，但这里我们只要整数，因此强制转换成int类型。如果System.out.println(“number:”+i);写在if{}里面，表示只有i>3的时候才打印出来，否则没有任何回显

* 双分支结构：if-else双分支选择结构当条件表达式为真时，执行语句块1，否则执行语句块2，也就是else部分



**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **int** r = 3;  
 **double** PI = 3.14;  
 **double** area = PI \* r \* r;  
 **double** length = 2 \* PI \* r;  
 **if**(area > length){  
 System.***out***.println(**"面积大于等于周长"**);  
 }**else**{  
 System.***out***.println(**"面积小于周长"**);  
 }  
 }  
}

**//导入包  
import** java.util.Scanner;  
**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **/\*  
 创建文件扫描器对象  
 System.in表示的是标准输入，可以从控制台读取数据。  
 nextLine()是Scanner的一个方法，表示读取当前行的数据  
 注意：每次读取回来的值都是字符串类型，需要进行类型转换  
 \*/** Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入数据"**);  
 String str = input.nextLine();  
 System.***out***.println(str);  
 }  
}

执行结果：

请输入数据

**nihao //自己输入的**

nihao **//自动出现的**

**import** java.util.Scanner;  
**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 System.***out***.println(**"欢迎光临马士兵教育会员系统"**);  
 System.***out***.println(**"添加用户信息"**);  
 Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入会员号码<4位整数>"**);  
 String number = input.nextLine();  
 System.***out***.println(**"请输入会员生日<日/月>"**);  
 String birthday = input.nextLine();  
 System.***out***.println(**"请输入会员积分"**);  
 String score = input.nextLine();  
  
 **if**(number.length()==4){  
 System.***out***.println(**"会员信息如下："**);  
 System.***out***.println(number+**"\t"**+birthday+**"\t"**+score);  
 }**else**{  
 System.***out***.println(**"会员号码输入错误，请重新输入："**);  
 }  
 }  
}

* 多分支结构：If-else。逐条if语句进行判断。条件匹配，进入语句体，否则对if语句继续匹配

if（布尔表达式1）{

-语句块1；

} else if (布尔表达式2) {

-语句块2；

}……

else if (布尔表达式n) {

-语句块n；

} else {

-语句块n+1；

}

**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **int** age = (**int**)(Math.*random*()\*100);  
 **if**(age<10){  
 System.***out***.println(**"儿童"**);  
 }**else if**(age<20){  
 System.***out***.println(**"少年"**);  
 }**else if**(age<30){  
 System.***out***.println(**"青年"**);  
 }**else if**(age<50){  
 System.***out***.println(**"中年"**);  
 }**else if**(age<70){  
 System.***out***.println(**"老年"**);  
 }**else**{  
 System.***out***.println(**"耄耋"**);  
 }

}  
}

**import** java.util.Scanner;  
**public class** HelloWorld {  
 **public static void** main(String[] args) {  
Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入成绩查询："**);  
 String score1 = input.nextLine();  
 **int** score = Integer.*valueOf*(score1);  
 **if**(score>=90){  
 System.***out***.println(**"优秀"**);  
 }**else if**(score>=80){  
 System.***out***.println(**"良好"**);  
 }**else if**(score>=60){  
 System.***out***.println(**"中等"**);  
 }**else**{  
 System.***out***.println(**"差"**);  
 }  
 }  
}

因为input.nextLine()返回的为string类型，所以定义score1也要为string类型。int

score = Integer.valueOf(score1); 中**Integer.valueOf（）**将括号中变量**转换为Int**类

型。这里不能通过int score = (int)(score1)强制转换类型。会显示String无法转换

为int。这样屏幕上显示请输入成绩查询，自己输入成绩，之后会显示优秀，良好，

中等还是差

也可以这样写↓：

**int** score = input.nextInt();

这里Scanner提供nextInt方法，可以直接用转换成int数据类型

* 嵌套分支结构if。在分支结构中嵌套分支结构

学校举行运动会，百米赛跑跑入20秒内的学生有资格进决赛。根据性别分别进入男子组和女子组。要判断是否能够进入决赛。在确定进入决赛的情况下，还要判断是进入男子组还是女子组

**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **int** time = (**int**)(Math.*random*()\*40);  
 **if**(time<20){  
 System.***out***.println(**"恭喜进入决赛！"**);  
 String gender = ((**int**)(Math.*random*()\*2))==0?**"girl"**:**"boy"**;  
 **if**(gender == **"girl"**){  
 System.***out***.println(**"欢迎进入女子组"**);  
 }**else**{  
 System.***out***.println(**"欢迎进入男子组"**);  
 }  
 }**else**{  
 System.***out***.println(**"成绩太差，淘汰"**);  
 }  
 }  
}

* switch多分支选择结构

根据表达式值的不同执行许多不同的操作

switch语句会根据表达式的值从相匹配的执行，一直执行到break标签处开始ak语句处或者是switch语句的末尾

每个case模块中要添加break，防止多次匹配

如果多个case处理的逻辑代码块的功能一致，可以考虑只在最后添加一次处理

与任一case值不匹配，则进入default语句（如果有的话），default可以没有

只能处理等值条件判断的情况，且表达式必须为byte, short, int 或char类型，不能是String或double，float。1.7之后可以使用string

常量值必须是与表达式类型兼容的特定的一个常量

不允许有重复的case值。default子句为可选

switch(表达式) {

case值1:

语句序列；

[break];

case值2:

语句序列；

[break];

… … … … …

[default:

默认语句;]

}

**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **int** random = (**int**)(Math.*random*()\*26);  
 **char** alphabet = (**char**)(**'a'**+random);  
 **switch** (alphabet){  
 **case 'a'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'o'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'e'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'i'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'u'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'y'**:  
 System.***out***.println(**"半元音："**+alphabet);  
 **break**;  
 **case 'w'**:  
 System.***out***.println(**"半元音："**+alphabet);  
 **break**;  
 **default**:  
 System.***out***.println(**"辅音："**+alphabet);  
 }  
 }  
}

上面的代码可以简化成下面这样

代码的功能就是随机出现26个字母，并打印出是元音，半元音还是辅音

**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **int** random = (**int**)(Math.*random*()\*26);  
 **char** alphabet = (**char**)(**'a'**+random);  
 **switch** (alphabet){  
 **case 'a'**:  
 **case 'o'**:  
 **case 'e'**:  
 **case 'i'**:  
 **case 'u'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'y'**:  
 **case 'w'**:  
 System.***out***.println(**"半元音："**+alphabet);  
 **break**;  
 **default**:  
 System.***out***.println(**"辅音："**+alphabet);  
 }  
 }  
}

也可以写作下面这种

**public class** HelloWorld {  
 **public static void** main(String[] args) {  
 **char** alphabet = (**char**)(**'a'**+Math.*random*()\*26);  
 **switch** (alphabet){  
 **case 'a'**:  
 **case 'o'**:  
 **case 'e'**:  
 **case 'i'**:  
 **case 'u'**:  
 System.***out***.println(**"元音："**+alphabet);  
 **break**;  
 **case 'y'**:  
 **case 'w'**:  
 System.***out***.println(**"半元音："**+alphabet);  
 **break**;  
 **default**:  
 System.***out***.println(**"辅音："**+alphabet);  
 }  
 }  
}

比较switch和多重if选择结构

相同点：都是用来处理分支条件的结构

不同点：

-switch选择结构只能处理等值条件判断的情况，而且条件必须是整型变量或字符型变

量或者字符串

-多重if选择结构没有switch选择架构的限制，特别适合某个变量处于某个连续区间时

的情况