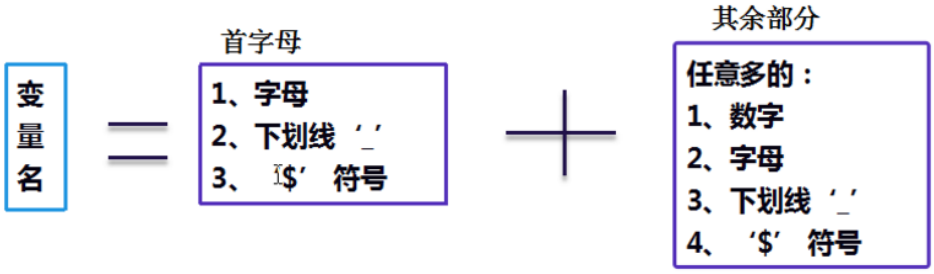
1. 关键字
-- 计算机语言里事先定义的，有特别意义的标识符，有时又叫保留字
Java 中的关键字
--对 Java 的编译器有特殊的意义，用来表示数据类型、程序结构等
--关键字不能用作变量名、方法名、类名、包名和参数。

数据类型：
boolean int long short byte float double char class interface
流程控制：
if else do while for switch case default break continue return try catch finally
修饰符：
public protected private final void static strictfp abstract transient
synchronized volatile native
动作：
package import throw throws extends implements this super instanceof new
保留字：
true false null goto const
关键字的注意事项
-- 所有 Java 关键字都是小写的
-- goto 和 const 虽然从未使用，但是也被作为 Java 的关键字保留
-- true,false 用在特殊的用途，但是不是 Java 的关键字

2. 使用变量的步骤：
第一步：**声明变量**，即“根据数据类型在内存申请空间”
数据类型 变量名;
例如：int money;
第二步：**赋值**，即“将数据存储至对应的内存空间”
变量名 = 数值;
例如：money = 1000 ;
第一步和第二步可以合并
数据类型 变量名 = 数值;
例如：int money = 1000;
第三步：**使用变量**，即“取出数据使用”
变量命名规范：
简短且能清楚地表明变量的作用(见名之意)，通常第一个单词的首字母小写，其后单词的首字母大写。
例如：myScore



字符串 String 类型变量的值是以双引号引起来的，字符 char 类型的值是单引号引起来的。

3.常量：
--代表程序运行过程中不能改变的值
作用：
--代表常数，便于程序的修改（例如：圆周率的值）
--增强程序的可读性
（例如：常量 UP、DOWN、LEFT 和 RIGHT 分 别代表上下左右，其数值分别是 1、2、3 和 4）
规范：
--常量名大写；多个单词全部大写，并以下划线分割，如 NAME, ROLE_NAME
声明常量并赋值：
final 数据类型 常量名称 = 值;
例如： final double PI = 3.1415926; //声明一个 double 常量并赋值
final String NAME = "JACK"; //声明一个 String 类型的常量并赋值
同时声明多个常量并赋值，多个常量之间以逗号 分隔
final 数据类型 常量名称 1 = 值 1， 常量名称 2 = 值 2，…常量名称 n = 值 n;
例如： //同时声明多个常量,以逗号分隔
final char MALE='M',FEMALE = 'F';
注意：Java 允许常量先声明后赋值；但是只能赋值一次
//常量只能赋值一次
final int num; //声明一个常量
num = 25; //首次给常量赋值
num =36; //再次给常量赋值则不允许的！
编辑器报错：The final local variable num may already have been assigned

4.java 分割符的种类
分号-“;”；逗号-“,”；空格-“ ”；圆点-“.”；花括号-“{}”
Java 分割符的作用
分号：语句必须以分号作为结束标记，for 循环中使用分号来分割不同的成份
逗号：用于方法中多个参数列表的分割，或者用于声明语句中同时声明多个属性或局部变量时起分割作用

空格或换行：Java 代码中各个组成部分之间可以插入任意数量的空格或换行---格式化代码

圆点：用于访问对象成员(属性或方法)标明调用或者隶属关系，其格式为“对象名.对象成员”

花括号：用于构造语句块---语句块

5.标识符

Java 所有的组成部分都需要有名字，为各种变量、常量、方法、类、接口等起的名字称为标识符(起别名)

标示符的命名规则

应以字母、下划线、美元符号开头(a-z/A-Z、_、$)

后跟字母、下划线、美元符号或数字

标示符的大小写敏感

标示符不能与关键字、保留字重名

见名之意。

- 合法标识符举例：age、$salary、_value、__1_value
- 非法标识符举例：123abc、-salary

6.数据类型

一组性质相同的值的集合以及定义在这个值集合上的一组操作的总称

按照对数据类型的使用和约束程度来划分

强类型语言(Strong Typed languages) -----典型代表 Java 语言

每个变量都要有确定的类型

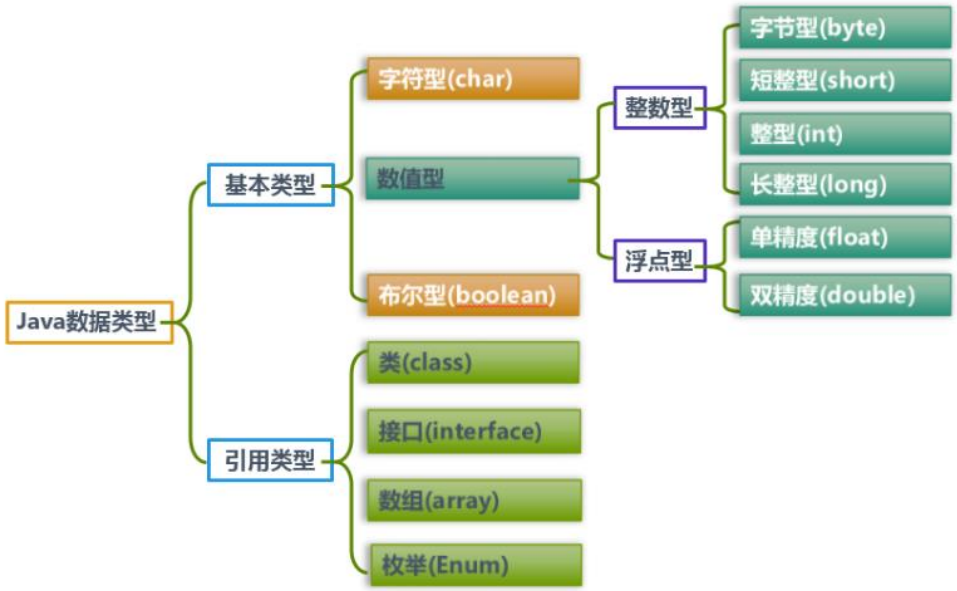
变量要先声明后使用

弱类型语言(Weakly Typed languages) -----典型代表 javaScript

每个变量的类型是可以变化的

变量可以不声明就使用

5.Java 数据类型图示：



public final class String-----说明字符串是属于引用类型的 class！

6. 基本类型

Java 中的基本类型的变量不是对象，它只能保存单一的值被称为原始类型、或简单类型

Java 中四类/八种基本数据类型

整数型：byte　short　int　long

浮点型：float　double

字符型：char

布尔型：boolean

7. 整数型

用于保存整数信息，java 提供了四种不同的整数类型，各有固定的表示范围和字段长度，不受具体操作系统的影响，以保证 Java 的可移植性

类型说明表

类型	占有存储空间	表示范围
byte	1 字节	-128—127
short	2 字节	-215—215-1
int	4 字节	-231—231-1
long	8 字节	-263—263-1

整数型字面量表示形式

-十进制：如：78，-65，0

-八进制：以 0 开头。如 023

-十六进制：以 0x 或 0X 开头。如 0x35

-默认的整数型字面量为 int 类型，声明为 long 类型可以在字面量后面加 l 或者 L，建议使用大写的 L 避免引起误解。

8.浮点型

包括 float 和 double 两种类型，分别用于保存单精度和双精度的浮点数

类型说明表

类型	占有存储空间	表示范围
float	4 字节	-3.403E38—3.403E38
double	8 字节	-1.798E308—1.798E308

浮点型字面量表示形式

-十进制：必须有小数点，也可以采用科学计数法表示。如：5.75，8.54e5，0.34e-8

-十六进制：从 JDK5.0 开始引入，十六进制浮点数只能采用科学计数法表示。

如：0x3.6p5=(3*160+6*16-1)*25

-默认的浮点型字面量为 double 类型，声明为 float 类型可以在字面量后面加 f 或 F

9.字符型

char 类型用来表示字符类型，Java 语言采用 16 位 Unicode 编码保存字符

字符型字面量的表示形式

1、使用**单引号**括起来的单个字符。如：char ch= ‘b’ ;

2、使用 16 进制的 Unicode 编码形式表示。如：char ch=“\u0045”;

3、使用转义字符”\”来将其后面的字符转换位其他的含义。

如：char ch=“\n”

10.布尔型

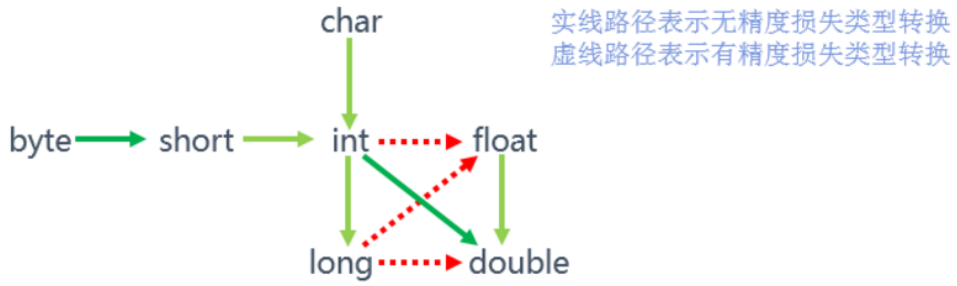
-逻辑运算和程序流程控制中用来表示两种状态的逻辑值，代表特定条件的成立与否

布尔型字面量的表示形式

-只允许两个值 true 和 false，不能用 0 和非 0 来替代 true 和 false

11.类型转换

将数值从一种类型转换为另一种类型的操作，我们称为类型转换



自动类型转换---小范围向大范围转没有问题

-与上图中箭头指向相同的数值类型转换会在表达式中自动完成

```
short price2 = (short)236;
```

```
//基本数据类型转换，小 short 转大 long，直接自动转换
long testlong;
testlong = price2;
```

强制类型转换---大范围向小范围转换可能会造成丢失精度甚至是错误

-与上图中箭头指向相反方向的数值类型转换需要强制进行，强制类型转换一般会造成数值的精度损失。

-强制类型转换语法格式：**（目标数据类型）<表达式>**

```
int money = 1000; // 声明并赋值
//基本数据类型转换，大 int 转小 short 需要强制转换
short price;
price = (short) money;
```

12.引用数据类型

Java 语言中出来 8 种基本数据类型以外，其他的数据类型统称位引用类型(Reference Type)。具体包括：类，接口，数组，枚举和注解类型

引用类型的数据：以对象的形式存在

引用类型的变量：它的值是某个对象的句柄/地址，而不是对象本身

声明引用类型变量时，系统只为该变量分配引用的空间，并未创建对象；

只有使用 new 运算符调用类的构造方法时才会创建对象，并返回该对象的句柄

13.一个引用类型的类，通常具有：

A.构造方法；----用于构造该类的一个实例对象；

B.成员属性；----用于描述该对象的特征、特点、特性等；

成员属性，又分为成员变量、成员常量；

C.成员方法；----用于体现该对象的行为、功能、作用等；

D.静态 static 的类层次的属性；

E. 静态 static 的类层次的方法。

14. 构造方法和普通方法的区别：

A. 普通方法的方法名前面有返回数据类型的声明，构造方法没有；

B. 构造方法的方法名和类名必须完全一致，不一致不是构造函数，有返回类型声明就不是构造方法；

C. 构造方法没有返回数据类型的声明，却会返回类类型的一个实例对象；

15. 变量(Variable) ---用于记录数值，并可以改变数值的数据

变量的组成：变量=变量名+变量值

变量名：用于标记一段特定的存储空间

变量值：以二进制的形式保存在存储空间中，且可以被访问和修改的数值

变量的分类

按变量所属的数据类型分类

基本类型

引用类型

按变量声明的位置分类

局部变量——方法或者语句块中声明的变量

成员变量——方法外部、类的内部声明的变量

16. 变量的声明和初始化：

声明变量：Java 是强类型语言，变量必须先声明后使用。声明变量即为变量分配相应的存储空间

初始化变量：为内存中已经存在的变量赋于初始值

语法格式：[<modifiers>] <type> <name1> [=defaultValue1][,<name2> [=defaultValue2]]…;

例如：public int age =18,hight=175;

[]中括号的内容表示可选的。

17. 变量的作用域和生存期：

变量的作用域：变量的作用域也称为变量的作用范围，即一个变量在多大的范围内可以使用

成员变量：作用域与其所属对象的作用域相同

局部变量：作用域就是它所在的方法或语句块

变量的生存期：变量的生存期就是变量的生存时间，即变量从创建到销毁所经历的时间

成员变量：成员变量的生存期与其所属对象相同，随着对象的创建而创建，随着对象的销毁而销毁

局部变量：局部变量的生存期就是其所属方法或语句块的单次执行期间

18. 变量的存储：

存储位置的分类

寄存器：寄存器在处理器的内部，空间最小，速度最快。寄存器根据系统的需要分配，程序员无法直接控制

栈：存储空间连续，以栈的方式进行管理容量小，访问速度快

堆：存储空间不连续，容量大，但是速度慢

程序代码内部：常量通常存储在程序代码内部，这样是安全的，因为它永远不会被改变。

外存储器：用于存储持久化数据

Java 中变量的存储

局部变量（基本类型、引用类型的句柄/引用）保存在**栈内存中**

对象及其成员变量（包含基本类型和引用类型）保存在**堆内存中**

19. java 数据存储：

20. 方法：

方法

Java 语言中的方法相当于其他计算机语言中的函数或子程序，是用来完成相对独立的功能的一段代码的集合。

从类的角度来看，方法是类的动态性能描述了该类事务共有的功能或行为

语法规则

Java 的方法必须定义在类中，而不允许直接定义在源文件中

方法的定义不允许出现嵌套

只有方法所在的类的对象才有资格调用该方法

语法格式

[<modifiers>] return_type <method_name>([<argu_list>]){
[<statements>]
}

21. 方法的四要素：----并不是全部必需的

返回类型(返回值)、形式参数、实际参数、方法体

返回类型

事先约定的返回值的数据类型。如果没有返回值，则返回类型定义为 void；定义构造方法时，则不允许给出返回值类型。

返回值

使用 return 关键字来返回值，return 的作用是终止方法的运行并返回其结果给调用它的环境。使用的格式是：return someValue;

如果方法的返回类型为 void，则可以省略 return;语句。编译时系统会自动添加该语句

形参和实参

形参：方法被调用时用于接受外界输入的数据

实参：调用方法时实际传给方法的数据

方法体

多条 Java 语句组成的语句块，用于完成特定的功能

22. 方法的参数传递：

目前主流的说法是**值传递和引用传递**，

其中基本类型的传递为值传递，传递前后变量的值不变； ----即便在方法内部更改值，也只是在方法内部有效！

引用类型的传递为引用传递，传递前后变量（对象）的值会改变。----如果在方法中改变了对象的属性值，就真的改掉了！

23.方法形参和方法实参：

形参：用于方法定义时，需要**明确参数的个数、顺序、参数的数据类型**；

例如：public boolean display2(**int height,int weight**) {...}

实参：用于方法调用时，需要**提供形参约定的参数个数及顺序**。不需要在实参前面加数据类型，但是如果参数类型不一致，直接报错！例如：boolean result= person.display2(156, 150);