# day02

## Java基本概念

### 注释

1. 注释 : 对于代码起到解释说明作用, 也可以将代码变成无效代码
2. 注释分类 :

a : 单行注释 : // 表示每次可以注释一行内容, 不能有回车换行, 可以嵌套使用

b : 多行注释 :

/\*

多行注释中, 可以有回车换行

多行注释不能嵌套使用

\*/

1. 注意:
2. 注释不会影响代码的运行效果

a : 编译时, 不检查注释语法结构

b : 运行时, 不会运行注释内容

1. 每一个开发人员在实际开发中, 必须写注释

代码

|  |
| --- |
| package com.ujiuye.base;  // 单行注释 : 定义出一个类, 类名Demo01\_注释  public class Demo01\_注释 {  /\*  \* 多行注释 :  \* 定义出程序唯一执行入口, main方法  \* main方法中实现的功能就是输出打印 "你好"  \* // 我是嵌套的单行注释  \*/  public static void main(String[] args) {  System.out.println("你好");  //System.out.println("你好吗?");  System.out.println("你好不好?");  }  } |

### 关键字

1. 关键字 : Java中表示特殊含义, 具有特殊语法结构英文单词



1. 注意 :
2. 关键字全部都是小写字母
3. 在进行命名(起名字)时, 单独的关键字不能作为名字出现
4. 在编辑工具中, 关键字具有特殊颜色

### 标识符

1. 标识符 : 在进行java代码开发时, 如果涉及到命名(起名字), 类, 接口, 变量, 方法, 常量命名需要符合标识符规范(规则)
2. 标识符组成 :

a : 大小写字母, a-z A-Z

b : 数字

c : 特殊字符, \_ $

d : 其他国家语言文字可以作为标识符出现(例如中文), 但是实际开发中,中文一般不会作为名字出现

1. 标识符规则 :

a : 关键字不能单独作为名字出现

b : 数字不作为名字的开头

c : 命名最好见名知义, 名字长度不限

代码

|  |
| --- |
| package com.ujiuye.base;  public class Demo02\_标识符 {  public static void main(String[] args) {  // int 名字 = 10;  int i = 10;  // int class = 10; 关键字不能单独作为名字出现  int class1 = 10;    int \_123 = 10;    // int my name = 10; 名字中不能出现空格,特殊符号只有\_和$可以作为名字出现  int my\_name$ = 10;    // int 123abc = 10; 数字不能作为名字的开头  }  } |

### Java命名行业内部规则

在满足标识符命名规范的基础上, 对于Java代码中的命名也有行业内部规则

1. 类和接口名 : 满足大驼峰原则, 每一个英文单词的首字母都大写

举例 : HelloWorld Demo01 GoldSale

1. 方法和变量名 : 满足小驼峰原则, 第一个英文单词全部小写, 从第二个英文单词开始首字母大写

举例 : getSum schoolName

1. 常量名 : 全大写, 多个英文单词之间使用\_进行分隔

举例 : COPY\_NAME PI

1. 包命名: 全球唯一, 公司域名倒序写作

举例 : com.ujiuye.base (.表示文件夹级别)

## 常量

1. 常量 : 在代码运行过程中, 其值不会发生变量的量(数据), 称为常量

举例 : 圆周率 π = 3.1415... 身份证号 学校名称

1. 常量按照表现形式划分 :

a : 字面值常量, 就是直接写出常量数据, 例如 : 5 , 6.77

b : 符号常量,目前不需要理解, 到面向对象环节,讲到final关键字, 再详细说明符号常量

1. 常量按照数据类型进行划分:

a : 基本数据类型, 一共4类, 用于表示最简单最直接数据

1. 整数类型 : 5 , 0 , -7
2. 浮点类型(小数类型) : 5.12 , 0.0 . -9.1
3. 字符类型: 使用一对英文的单引号包裹起来的单个字符

‘a’ ‘W’ ‘1’ ‘?’ ‘家’ ‘-1’// 不是字符类型, -1是两个字符,不是一个

1. 布尔类型 : 表示真假值, 是与否, 布尔类型的值只有两种 : true 真 false 假

b : 引用数据类型, 用于表示比较复杂数据, 例如 : 客户信息(客户姓名, 电话号码, 地址, 身份证号...)

1. 字符串类型 : 表示使用一对英文双引号包裹起来的一系列字符序列, 称为字符串

举例 : “Hello???”

1. 空常量 : null, 引用数据类型的默认值可以设置为null, null表示空
2. 字符串类型与其他任意数据类型相加(做加法), 只表示字符串的拼接, 结果还是一个字符串类型

代码

|  |
| --- |
| package com.ujiuye.base;  public class Demo03\_常量 {  public static void main(String[] args) {  // 整数类型  System.out.println(5);  // 浮点类型  System.out.println(5.2);  // 字符类型  System.out.println('a');  System.out.println('W');  System.out.println('9');  System.out.println(';');  System.out.println('家');  // System.out.println('-1'); 这不是一个字符, 两个字符    // 布尔类型  System.out.println(true);  System.out.println(false);    // 字符串常量  System.out.println("123abc");  String s = null;  System.out.println(s);  System.out.println("12 ??00--456YU你几号?");  System.out.println("2020年-10月-13日");    System.out.println("--------------");    // 字符串常量的加法运算  System.out.println("6"+ 5);// "65"  // 1) "6" + 5 = "65" 2)"65" + 4 = "654"  System.out.println("6"+ 5 + 4); // "654"  // 1) 5 + 4 = 9 2) "6" + 9 = "69"  System.out.println("6" + (5 + 4));// "69"  // 1) 5 + 4 = 9 2) 9 + "6" = "96" 3) "96" + 1 = "961"  System.out.println(5 + 4 + "6" + 1);// "961"  }  } |

## 变量

### 变量的定义

1. 变量 : 在Java代码运行过程中, 数据的值有可能发生变化, 这样的数据称为变量

举例 : 表示一个人的年龄, 表示体重, 客户电话号码, 客户家庭住址...

1. 变量的定义:

数据类型 变量名 = 值;

a : 数据类型, 表示变量所属数据是什么类型

b : 变量名 : 符合标识符规范, 小驼峰原则

c : = : 赋值运算符, 表示将等号右边的数值赋值给等号左边的变量名, 以后在代码中, 这个变量名就可以表示指定的数值

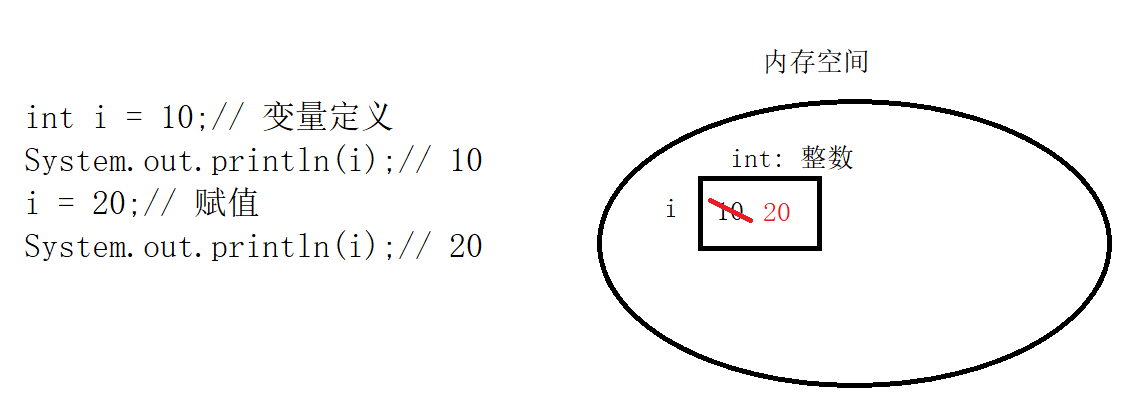
d : 值 : 数值需要与数据类型匹配

举例 : int i = 10; // int表示整数类型数据 i变量名 10表示整数的数值结果 在代码中使用变量i, 表示使用10数值

1. 定义出一个变量, 这个变量实际上就是在内存中占有一块空间区域, 内存空间中存储的就是变量对应的数值

内存 : 例如 : 电脑配置中内存4GB

内存作用 : 如果有程序或者应用需要运行, 进入到内存中占有一定空间运行, 当程序或者应用运行完毕, 程序或者应用占有的内存空间就会被释放掉



代码

|  |
| --- |
| package com.ujiuye.base;  public class Demo04\_变量的定义 {  public static void main(String[] args) {  // 数据类型 变量名 = 值;  int i = 10;  System.out.println(i + 1);// 11  // 变量赋值 : 给变量i设置一个新的数值, 达到数值变量过程  i = 20; // 从第10行开始, 变量i的值已经变成20, 原来的10的值被20覆盖掉  System.out.println(i);// 20  }  } |

### 基本数据类型

扩展: 内存中存储单位换算

1. 计算机中, 对于文件的最小丈量单位是字节(Byte)
2. 计算机世界中对于数据的存储, 所有的数据都是以数值方式进行存储, 计算机中最小的数据单位一个2进制位(逢2进1), 称为比特位(bit), 0 1 10--->2 11--->3 100--->4
3. 内存中进制单位转换:

连续的8个2进制位,就是1个字节大小 0000 0000 ---- 1111 1111 --->1个字节

8bit = 1B(1Byte-->字节)

1024B = 1KB

1024KB = 1MB

1024MB = 1GB

1024GB = 1TB

1024TB = 1PB ...

1. Java语言在定义一个变量时, 必须给出变量对应所属的数据类型, 因为不同的数据类型在内存中占有空间大小不同
2. 基本数据类型 : 一共4类8种, 基本数据类型就是用于表示最简单最直接的数据

a : 整数类型(4种)

byte 1字节 -128----127

short 2字节 -2^15----2^15-1

int 4字节 -2^31----2^31-1 (最常用整数类型)

long 8字节 -2^63---2^63-1

b : 浮点类型(小数类型) 2种

float 4字节 单精度(小数位数精度不太高)

double 8字节 双精度(小数位数精度相对较高) 最常用浮点类型

c : 字符类型

char 2字节 0-65535

d : 布尔类型

boolean 未知 true false

注意 : 四类八种数据类型中的每一种, 都是关键字

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo01\_四类八种基本数据类型定义 {  public static void main(String[] args) {  // 整数类型  byte b = 15;  System.out.println(b);  short s = 15000;  System.out.println(s);  int i = 170000;  System.out.println(i);  long l = 123456789;  System.out.println(l);    // 浮点类型  // float类型的变量定义时, 数据后面需要添加F, 作为类型标识  float f = 3.14F;  System.out.println(f);  double d = 2.456789;  System.out.println(d);    // 字符类型  char ch = 'W';  System.out.println(ch);    // 布尔类型  boolean boo = true;  boolean boo1 = false;  System.out.println(boo);  System.out.println(boo1);  }  } |

### 常量的默认类型

1. Java中, 定义出一个整数常量, 默认是int类型, 于是随意写出的整数, 其数据范围就需要在int范围中, 如果定义一个long类型数据, 数值超过了int类型范围, 那么需要在long类型数值后添加L, 作为长整型数据标识
2. Java中, 定义出的浮点类型,默认double类型, 因此float类型单精度的变量定义, 数值后需要添加F作为标识

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo02\_常量默认类型 {  public static void main(String[] args) {  // 5常量int类型数据  System.out.println(5);  //The literal 12345678912 of type int is out of  //System.out.println(12345678912);  long l = 12345678912L;    // 3.14常量double类型  System.out.println(3.14);  //Type mismatch: cannot convert from double to float  //float f = 3.14;  float f = 3.14F;  }  } |

### 变量定义注意事项

1. 变量具有作用范围:

自变量定义位置起,包含这个变量,离这个变量最近的一对大括号就是这个变量使用范围

1. 相同作用域中的变量名不能重复
2. 变量声明和变量赋值可以分开

解释 : int i; // 变量声明 i = 20; // 变量赋值

1. 定义在方法中变量, 如果没有赋值不能使用
2. 相同类型变量可以连续定义

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo03\_变量定义注意事项 {  public static void main(String[] args) {  /\*1.变量具有作用范围:  自变量定义位置起,包含这个变量,离这个变量最近的一对大括号就是这个变量使用范围\*/  int i = 10;    {  int y = 20;  System.out.println(i + y);// 30  }  // y cannot be resolved to a variable(y无法解析成一个变量)  // System.out.println(y); 错误原因是y使用超出变量作用范围  System.out.println(i);    // 2.相同作用域中的变量名不能重复  int w = 10;  // int w = 19;  // Duplicate local variable w : 重复本地变量w  // double w = 3.14;  System.out.println(w);    // 3.变量声明和变量赋值可以分开  // 解释 : int i; // 变量声明 i = 20; // 变量赋值  int t;  // 4. 定义在方法中变量, 如果没有赋值不能使用  //The local variable t may not have been initialized(初始化,表示赋值)  // 本地变量t, 可能没有赋值  // System.out.println(t);  t = 99;  System.out.println(t);// 99    // 5.相同类型变量可以连续定义(了解)  int z = 80, v = 20;  }  } |

### 自动类型转换

1. 结论: 不同数据类型(基本数据类型)数据彼此之间可以进行转换和运算
2. 自动类型转换 : 也称为隐式转换, 小类型数据可以直接转换成大类型数据; 小类型数据与大类型数据进行运算, 小类型数据自动提升类型为大类型, 计算结果以大数据类型为准
3. 四类八种基本数据类型之间从小到大排序

计算机世界都是2进制0和1, 因此对于浮点类型计算与整数计算不一致, 因此导致浮点类型可以表示的数据范围整体上大于整数

byte < short = char < int < long < float < double boolean类型不参与排序

1. 注意 :

byte, short, char, int类型数据,自身或者彼此之间进行运算, 统统默认提升类型为int, 如果运算过程中, 有大于int类型的数据, 那么以最大类型数据为提升标准

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo04\_自动类型转换 {  public static void main(String[] args) {  // 1. 小数据类型直接转换成大数据类型  byte b = 15;  int i = b;    // 2. 小数据类型与大数据类型进行运算, 小类型数据自动提升类型为大类型, 计算结果与大数据类型为准  short s = 15;  //Type mismatch: cannot convert from int to short  //short ss = s + 1;    int ss = s + 1;  System.out.println(ss);// 16  System.out.println(i);// 15    double d = ss;  System.out.println(d);// 16.0    //byte, short, char, int类型数据,自身或者彼此之间进行运算, 统统默认提升类型为int, 如果运算过程中, 有大于int类型的数据, 那么以最大类型数据为提升标准  byte b1 = 88;  short s1 = 99;  //Type mismatch: cannot convert from int to short  //short re = b1 + s1;  int re = b1 + s1;  System.out.println(re);// 187  }  } |

### 强制类型转换

1. 强制类型转换 : 表示将大数据类型转换成小数据类型, 需要公式

公式 :小数据类型 = (小数据类型)需要转换的大类型数据;

注意 : 强制类型转换有可能造成数据丢失, 于是进行强制转换的时候, 需谨慎

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo05\_强制类型转换 {  public static void main(String[] args) {  // 1. 强制类型转换数据不丢失  //公式 :小数据类型 = (小数据类型)需要转换大类型数据;  int i = 88;  byte b = (byte)i;  System.out.println(b);// 88    // 2. 强制类型转换数据丢失  int w = 1000;  byte b1 = (byte)w;  System.out.println(b1);// -24    double d = 3.001;  int t = (int)d;  System.out.println(t);// 3  }  } |

### 字符类型

1. 字符类型,属于基本数据类型, 使用关键字char进行类型表示, 使用一对英文的单引号包裹起来的单个字符, 就是一个字符类型, char ch = ‘a’;
2. 计算机世界中, 存储数据都是数值, 计算机中只有0和1二进制数据表示, 人类语言文字如何可以被计算机识别和存储, 于是需要一个桥梁, 需要把人类语言文字与整数(数值)建立对应关系 , 举例 : 0-->00110000 A-->01000001
3. 初期, 美国人将美国符号, 字母, 数字与整数建立关系, 形成了ASCII编码表(美国标准信息交换码表)
4. 有了编码表, 字符类型与其整数之间有对应关系, 因此字符类型也可以通过编码表转换成整数, 从而与其他基本数据类型进行运算
5. ASCII编码表中常用的字符与整数对应关系需要记住

0-9 : 48--57, 记0是48即可

1. Z : 65-90, 记A是65即可
2. z : 97-122, 记a是97即可
3. 中国有自己编码表:
4. GBK : 中国标准编码表,兼容ASCII,同时具有所有中文文字,一个中文占有2个字节大小
5. UTF-8 : 万国码表, 兼容ASCII, 同时具有全世界所有国家语言文字, 包括中文, 一个中文占有3个字节大小



代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo06\_字符类型与ASCII编码表 {  public static void main(String[] args) {  char ch = 'a';  // 1) ch参考编码表, 得到97,提升类型为int  // 2) 97 + 5 = 102(int类型)  int result = ch + 5;  System.out.println(result);// 102    char ch1 = (char)result;  System.out.println(ch1);// f  }  } |

### 字符串类型

1. 字符串类型 : 使用一对英文双引号包裹起来的一系列字符序列, 引用数据类型
2. 字符串类型使用String表示

String s = “hello?”;

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo07\_String字符串类型 {  public static void main(String[] args) {  String s = "你好";  System.out.println(s+1);    String s1 = "89";  System.out.println(s1 + 5);// "895"  }  } |

## 运算符

运算符:

基本算数运算符

自增自减运算符

赋值运算符

比较运算符

逻辑运算符

位移运算符(了解)

三元运算

### 基本算数运算符

1. + - \* / %

a : + 加法运算

1. 直接作用在数值之前, 表示正数 +3
2. 做数值之间的加法运算, 3 + 2 = 5
3. 做字符串加法, 字符串类型与任意数据类型相加, 只表示字符串拼接, 结果仍然是字符串类型

b : - 减法运算

1. 直接作用在数值之前, 表示负数 -3
2. 做数值之间的加法运算, 3 - 2 = 1

c : \* 乘法运算

1. 做数值之间的乘法运算 3 \* 2 = 6

d : / 除法运算

1. 如果/符号两端的数据都是整数类型, 那么相除的结果也是整数, 对于小数部分全部舍掉 5 / 3 = 1
2. 如果/符号两端有其中一端或者两端是浮点类型, 那么结果就可以带有小数

5.0 / 3 = 1.677777...

e : % 取模运算(取余运算) : 两数相除, 获取到余数

%取模运算使用场景, 通常可以进行数据筛选操作

7 % 3 = 7 / 3 = 2...1 取模结果就是1

10 % 5 = 10 / 5 = 2...0 获取结果就是0

%取模运算符注意 : 取模运算也带有符号位(正负), 取模运算结果的符号位与 %前面的被除数符号位一致

需求 : 定义出一个任意正整数, 判断这个数是奇数还是偶数

分析 : 偶数, 可以被2整除数, 数 / 2 除法, 没有余数,整除

数 % 2 与0数值进行比较大小, 结果就是0 ,证明数是偶数

代码

|  |
| --- |
| package com.ujiuye.variable;  public class Demo01\_基本算数运算符 {  public static void main(String[] args) {  // +  System.out.println(+3);// 3  System.out.println(3 + 2);// 5  System.out.println("6" + 5);// "65"    // -  System.out.println(-3);// -3  System.out.println(-3 - 2);// -5    // \*  System.out.println(3 \* 2);// 6    // /  System.out.println(5 / 3);// 1  System.out.println(5.0 / 3);// 1.6666666666666667    // %  // 7 % 3 = 7 / 3 = 2...1 取模结果就是1  System.out.println(7 % 3);// 1  // 10 % 5 = 10 / 5 = 2...0 获取结果就是0  System.out.println(10 % 5);// 0  // 取模运算结果的符号位与 %前面的被除数符号位一致  System.out.println(-7 % 3);// -1  System.out.println(7 % -3);// 1  System.out.println(-7 / 3);// -7 / 3 = -2...-1  System.out.println(7 / -3);// 7 / -3 = -2...1  }  } |