1. 常量和变量

硬盘

内存 16G

酷我音乐播放器

光辉岁月.mp3

真的爱你.mp3

不再犹豫.mp3

开辟内存区域

不再犹豫.mp3

硬盘

内存 16G

Jvm

HelloWorld.class

HelloWorld.class

int age

20

java HelloWorld

程序在运行期间，不断发生变化的量就是变量

程序在运行期间，不发生变化的量就是常量

生活中的常量：

整数的常量： 24小时，12月，60秒

小数常量：重力系数9.8， 圆周率3.14

字符串常量：“旺财”, “小强”

布尔类型的常量：true， false

1. Java的数据类型
   1. 基本数据类型

一共有8中基本数据类型

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

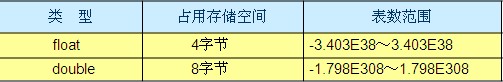
八位二进制位组成一个字节

一个字节表示的范围 -128 到 127

整数类型：



数值类型中最常用的类型就是int，其次是long



Float和long比较由于存储结构不同，float比long表示的范围大。

1. 局部变量的使用

局部变量定义在方法内或者语句块内。

变量的定义位置：定义在方法中或者语句块中。

变量的一定要赋初始值，否则使用该变量的时候会报错的。

在同一个作用域内变量名不能重复

变量的作用范围：**只在当前的大括号内有效（包括子括号）**，大括号外的所有程序不能使用该变量，所以这种变量也称为局部变量。

1. java基本数据类型的转换
   1. 自动转换

**byte---🡪short--🡪int ----🡪long--🡪float---🡪double**

转换规则：

1.所有的byte类型和short类型和char值运算的时候都会提供成int类型，int最大值2147483647。

2.如果计算中一个操作数据类型是long，另一个数的空间小于等于long，那么结果就是long

3.如果计算中一个操作数据类型是float，另一个数的空间小于等于float，那么结果就是float

4.如果计算中一个操作数据类型是double，另一个数的空间小于等于double，那么结果就是double

|  |
| --- |
| class Demo2{    public static void main(String[] args){  //定义一个字节类型    byte b = 1;  //基本数据类型在计算的时候， 数值常量默认是int  int b1 = b + 1;    short b2 = 12;  //byte, short , int, char, 四种类型只要是发生计算都会自动的把结果提升成int类型  int b3 = b + b2;    //定义一个char类型, 字符和int类型可以自动的转换  char c = 'a';  //int i2 = c;  int i1 = c + 1;    System.out.println(i1);    //如果两个int类型计算会不会自动的提升类型呢  int x = 2147483647;  int y = 12;  //int long float, double 自身来计算的时候没有类型的提升，如果超出范围了就会出现精度的损失  int z = x + y;  System.out.println(z);      //int long float, double 不同的类型来计算的时候没有类型的会向大的表数范围提升、    float f = 1.3f;  //int类型和float类型计算转换成float  float f1 = f + y;  //小数的常量的默认值是double  double d = f + 1.3;  // 12.5f是表示float类型的常量， 小的表数范围的常量或者变量可以自动的向大的表数范围的类型转换  double d1 = 12.5f;    double d2 = y;      short s = 5;  //大的表数范围的值不能自动的赋值小的表数范围的变量  s = s + 5;  }  } |

* 1. 强制转换

**当大容量类型的变量向小容量的类型去转换需要强制转换。**

|  |
| --- |
| class Demo3{    public static void main(String[] args){    int i = 999;  //数值类型的转换的前提是精度不会损失  byte b = (byte)i;  System.out.println(b);    //定义long类型  long l = 10000;  int j = (int)l;  System.out.println(j);  //小数类型的强转  float f = 199.5f;  //转换成int, 小数转换成整数，就是把小数去掉  short i1 = (short)f;    System.out.println(i1);    //double向float转换  double d = 10.4;  float f1 = (float)d;  System.out.println(f1);    //字符 的转换  char c = 'a';  int x = c;  System.out.println(x);    int y = 98;  char c1 = (char)y;  System.out.println(c1);  }  } |

1. 运算符
   1. 算数运算符

* 四则运算：

+，-，\*，/

|  |
| --- |
| class Demo4{    public static void main(String[] args){    //定义两个整数类型变量  int a = 10;  int b = 3;  //+，-，\*，/ %  int c = a + b;  System.out.println("a + b = "+ c);    int d = a - b;  System.out.println("a - b = "+ d);    int e = a \* b;  System.out.println("a \* b = "+ e);  //int直接做除法是整除  int f = a / b;  System.out.println("a / b = "+ f);    float f1 = 10.0f;  //由于f1是float类型b是int类型， 在做计算的时候类型做了提升  float f2 = f1 / b;  System.out.println("f1 / b = "+ f2);    //取余数  int g = a % b;  System.out.println("a % b = "+g);        }        } |

* ++、--给数值变量自身加1或减一

++： 后加加，

|  |
| --- |
| class Demo5{    public static void main(String[] args){  //后++  int a = 1;    a++;  System.out.println(a);    int b = 1;  //如果后加加和使用这个后加加在一起运算那么使用的时候加一之前的值，然后变量自身再加一  System.out.println(b++);  System.out.println(b);    }    } |

前加加

|  |
| --- |
| class Demo6{    public static void main(String[] args){  //前++  int a = 1;    ++a;  System.out.println(a);    int b = 1;  //如果前加加和使用这个前加加在一起运算那么使用的时候加一之后的值。  System.out.println(++b); //2    }    } |

后减减

|  |
| --- |
| class Demo7{    public static void main(String[] args){  //后--  int a = 1;    a--;  System.out.println(a); //0    int b = 1;  //如果后减减和使用这个后减减在一起运算那么使用的时候加一之前的值，然后变量自身再一减  System.out.println(b--); //1  System.out.println(b); //0    }  } |

前减减

|  |
| --- |
| class Demo8{    public static void main(String[] args){  //前--  int a = 1;    --a;  System.out.println(a); //0    int b = 1;  //如果前加加和使用这个前加加在一起运算那么使用的时候加一之后的值。  System.out.println(--b); //0    }    } |

练习：

|  |
| --- |
| class Demo9{    public static void main(String[] args){  int a = 1;    int b = 2;  // 1 \* 3 + 1 \* 3  int c = (a++) \* (++b) + (--a)\*(b++);    System.out.println("a = "+a); //1  System.out.println("b = "+b); //4  System.out.println("c = "+c); //6  }  } |

* 1. 逻辑运算符

&，|，&&, ||，！

&：多个表达式在计算的时候&与（每一个表达式的计算结果都是true的情况下），整体结果是true

&

|  |  |  |
| --- | --- | --- |
| 表达式1（数据是否及格） | 表达式2（语文是否及格） | 结果 |
| True | True | True |
| False | True | False |
| True | False | False |
| False | False | False |

|

多个表达式在计算的时候|或（每一个表达式的计算结果只要有一个表达式是true的情况下），整体结果是true

|  |  |  |
| --- | --- | --- |
| 表达式1（数据是否及格） | 表达式2（语文是否及格） | 结果 |
| True | True | True |
| False | True | True |
| True | False | True |
| False | False | False |

&&：双&号和单&号效果是相同的，运行机制不同

|  |
| --- |
| class Demo14{  /\*\*    \*/  public static void main(String[] args){    int math = 45;  int china = 23;    //单&号不够智能，即使前面的表达式的计算结果能决定总体结果，后面的表达式依然要计算  //System.out.println((math >= 60) & (++china >= 60));    //&&号断路与，前面的表达式的计算结果能决定总体结果，后面的表达式不计算  System.out.println((math >= 60) && (++china >= 60));  System.out.println(china);    }    } |

||和|的区别：

|  |
| --- |
| class Demo15{  /\*\*    \*/  public static void main(String[] args){    int math = 88;  int china = 23;    //单|号不够智能，即使前面的表达式的计算结果能决定总体结果，后面的表达式依然要计算  //System.out.println((math >= 60) | (++china >= 60));    //||是短路或，前面的表达式的计算结果能决定总体结果，后面的不计算  System.out.println((math >= 60) || (++china >= 60));  System.out.println(china);    }    } |