2. 数据类型与变量

【本节目标】

1. 字面常量
2. 数据类型
3. 变量

1. 字面常量

在上节课HelloWorld程序中，`System.out.println("Hello World");`语句，不论程序何时运行，输出的都是Hello World，其实"Hello World"就是字面常量。

常量即程序运行期间，固定不变的量称为常量，比如：一个礼拜七天，一年12个月等。

```
1   public class Demo{
2       public static void main(String[] args){
3           System.Out.println("hello world!");
4           System.Out.println(100);
5           System.Out.println(3.14);
6           System.Out.println('A');
7           System.Out.println(true);
8           System.Out.println(false);
9       }
10  }
```

其中：100、3.14、‘A’、true/false都是常量，将其称为字面常量。

字面常量的分类：

1. 字符串常量：由`""`括起来的，比如"12345"、"hello"、"你好"。
2. 整型常量：程序中直接写的数字(注意没有小数点)，比如：100、1000
3. 浮点数常量：程序中直接写的小数，比如：3.14、0.49
4. 字符常量：由 单引号 括起来的当个字符，比如：‘A’、‘1’
5. 布尔常量：只有两种true和false
6. 空常量：null(后面了讲)

注意：字符串、整型、浮点型、字符型以及布尔型，在Java中都称为数据类型。

2. 数据类型

在Java中数据类型主要分为两类：**基本数据类型**和**引用数据类型**。

基本数据类型有四类八种：

1. **四类：整型、浮点型、字符型以及布尔型**
2. **八种：**

数据类型	关键字	内存占用	范围
字节型	byte	1 字节	-128 ~ 127
短整型	short	2 字节	-32768 ~ 32767
整型	int	4 字节	-2^31 ~ (2^31)-1
长整型	long	8 字节	-2^63 ~ (2^63)-1
单精度浮点数	float	4 字节	有范围，一般不关注
双精度浮点数	double	8 字节	有范围，一般不关注
字符型	char	2 字节	0 ~ 65535
布尔型	boolean	没有明确规定	true 和 false

注意：

- 不论是在16位系统还是32位系统，int都占用4个字节，long都占8个字节
- 整型和浮点型都是带有符号的
- 整型默认为int型，浮点型默认为double
- 字符串属于引用类型，该种类型后序介绍。

> **什么是字节?**
>
> 字节是计算机中表示空间大小的基本单位.
>
> 计算机使用二进制表示数据. 我们认为 8 个二进制位(bit) 为一个字节(Byte).
>
> 我们平时的计算机为 8GB 内存, 意思是 8G 个字节.
>
> 其中 1KB = 1024 Byte, 1MB = 1024 KB, 1GB = 1024 MB.
>
> 所以 8GB 相当于 80 多亿个字节.

3. 变量

3.1 变量概念

在程序中，除了有始终不变的常量外，有些内容可能会经常改变，比如：人的年龄、身高、成绩分数、数学函数的计算结果等，对**于这些经常改变的内容，在Java程序中，称为变量。而数据类型就是**

用来定义不同种类变量的。

3.2 语法格式

定义变量的语法格式为：

```
数据类型  变量名 = 初始值;
```

比如：

```
int a = 10;   // 定义整型变量a，a是变量名也称为标识符，该变量中放置的值为10
double d = 3.14;
char c = 'A';
boolean b = true;

System.out.println(a);
System.out.println(d);
System.out.println(c);
System.out.println(b);

a = 100;  // a是变量，a中的值是可以修改的，注意：= 在java中表示赋值，即将100交给a，a中保存的值就是100
System.out.println(a);

// 注意：在一行可以定义多个相同类型的变量
int a1 = 10, a2 = 20, a3 = 30;
System.out.println(a1);
System.out.println(a2);
System.out.println(a3);
```


3.3 整型变量

3.3.1 整型变量

```
// 方式一：在定义时给出初始值
int a = 10;
System.out.println(a);

// 方式二：在定义时没有给初始值，但使用前必须设置初值
int b;
b = 10;
System.out.println(b);
```

```java

// 使用方式二定义后，在使用前如果没有赋值，则编译期间会报错
int c;
System.out.println(c);
c = 100;

// int型变量所能表示的范围:
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE);

// 注意：在定义int性变量时，所赋值不能超过int的范围
int d = 12345678901234;  // 编译时报错，初值超过了int的范围
```

注意事项：

1. int不论在何种系统下都是4个字节
2. 推荐使用方式一定义，如果没有合适的初始值，可以设置为0
3. 在给变量设置初始值时，值不能超过int的表示范围，否则会导致溢出
4. 变量在使用之前必须要赋初值，否则编译报错
5. int的包装类型为 Integer

3.3.2 长整型变量

```java
int a = 10;
long b = 10;   // long定义的长整型变量

long c = 10L; // 为了区分int和long类型，一般建议：long类型变量的初始值之后加L或者l
long d = 10l; // 一般更加以加大写L，因为小写l与1不好区分

// long型变量所能表示的范围：这个数据范围远超过 int 的表示范围. 足够绝大部分的工程场景使用.
System.out.println(Long.MIN_VALUE);
System.out.println(Long.MAX_VALUE);
```

注意事项：

1. 长整型变量的初始值后加L或者l，推荐加L
2. 长整型不论在那个系统下都占8个字节
3. 长整型的表示范围为：-2^63 ~ (2^63)-1
4. long的包装类型为Long

3.3.3 短整型变量

```
public static void main(Stirng[] args) {
    short a = 10;
    System.out.println(a);

    // short型变量所能表示的范围:
    System.out.println(Short.MIN_VALUE);
    System.out.println(Short.MAX_VALUE);
}
```

注意事项：

1. short在任何系统下都占2个字节
2. short的表示范围为：-32768 ~ 32767
3. 使用时注意不要超过范围(一般使用比较少)
4. short的包装类型为Short

3.3.4 字节型变量

```
byte b = 10;
System.Out.println(b);

// byte型变量所能表示的范围:
System.out.println(Byte.MIN_VALUE);
System.out.println(Byte.MAX_VALUE);
```

注意事项：

1. byte在任何系统下都占1个字节
2. byte的范围是：-128 ~ 127
3. 字节的包装类型为Byte

思考：byte、short、int、long都可以定义整型变量，为什么要给出4种不同类型呢?

这就好比买衣服时的尺码：

身高(cm) / 体重(kg)	105	155	160	165	170	175	180	185	190
40	S	S	S	M	M	L	L	XL	XL
45	S	S	M	M	L	L	XL	XL	2XL
50	S	M	M	L	L	XL	XL	2XL	2XL
55	M	M	L	L	XL	XL	2XL	2XL	2XL
60	M	L	L	XL	XL	2XL	2XL	2XL	3XL
65	L	L	XL	XL	2XL	2XL	2XL	3XL	3XL
70	L	XL	XL	2XL	2XL	2XL	3XL	3XL	3XL
75	XL	XL	2XL	2XL	2XL	3XL	3XL	3XL	3XL
80	XL	2XL	2XL	2XL	3XL	3XL	3XL	3XL	3XL

3.4 浮点型变量

3.4.1 双精度浮点型

```
1   double d = 3.14;
2   System.out.println(d);
```

神奇的代码一：

```
1   int a = 1;
2   int b = 2;
3   System.out.println(a / b);    // 输出 0.5 吗?
```

在 Java 中, int 除以 int 的值仍然是 int(会直接舍弃小数部分)。如果想得到 0.5, 需要使用 double 类型计算.

```
1   double a = 1.0;
2   double b = 2.0;
3   System.out.println(a / b);    // 输出0.5
```

神奇的代码二：

```
1   double num = 1.1;
2   System.out.println(num * num);  // 输出1.21吗?
```

```
3
4   // 执行结果
5   1.2100000000000002
```

注意事项：

1. double在任何系统下都占8个字节
2. 浮点数与整数在内存中的存储方式不同，不能单纯使用的形式来计算
3. double的包装类型为Double
4. double 类型的内存布局遵守 IEEE 754 标准(和C语言一样), 尝试使用有限的内存空间表示可能无限的小数, 势必会存在一定的精度误差，因此浮点数是个近似值，并不是精确值。

3.4.2 单精度浮点型

```
1   float num = 1.0f;    // 写作 1.0F 也可以
2   System.out.println(num);
```

float 类型在 Java 中占四个字节, 同样遵守 IEEE 754 标准. 由于表示的数据精度范围较小, 一般在工程上用到浮点数都优先考虑 double, 不太推荐使用 float. float的包装类型为Float。

3.5 字符型变量

```
1   char c1 = 'A';    // 大写字母
2   char c2 = '1';    // 数字字符
3
4   System.out.println(c1);
5   System.out.println(c2);
6
7   // 注意：java中的字符可以存放整型
8   char c3 = '帅';
9   System.out.println(c3);
```

注意事项:

1. Java 中使用 `单引号 + 单个字母` 的形式表示字符字面值.
2. 计算机中的字符本质上是一个整数. 在 C 语言中使用 ASCII 表示字符, 而 Java 中使用 Unicode 表示字符. 因此一个字符占用两个字节, 表示的字符种类更多, 包括中文.
3. char的包装类型为Character

3.6 布尔型变量

布尔类型常用来表示真假，在现实生活中也是经常出现的，比如：听说xxx同学买彩票中了一个亿...，听到后估计大部分人第一反应就是：真的假的?

```
1   boolean b = true;
2   System.out.println(b);
3
4   b = false;
5   System.out.println(b);
```

注意事项：

1. boolean 类型的变量只有两种取值, true 表示真, false 表示假.
2. Java 的 boolean 类型和 int 不能相互转换, **不存在** 1 表示 true, 0 表示 false 这样的用法.

```
1   boolean value = true;
2   System.out.println(value + 1);
3   //此代码就会出现编译错误
```

3. Java虚拟机规范中，并没有明确规定boolean占几个字节，也没有专门用来处理boolean的字节码指令，在Oracle的Java虚拟机实现中，Java编程语言中的布尔数组被编码为Java虚拟机字节数组，每个布尔元素使用8位。参考官方手册：Chapter 2. The Structure of the Java Virtual Machine
4. boolean的包装类型为Boolean。

3.7 类型转换

Java 作为一个强类型编程语言, 当不同类型之间的变量相互赋值的时候, 会有教严格的校验.

```
1   int a = 10;
2   long b = 100L;
3   b = a;   // 可以通过编译
4   a = b;   // 编译失败
```

在Java中，当参与运算数据类型不一致时，就会进行类型转换。Java中类型转换主要分为两类：自动类型转换(隐式) 和 强制类型转换(显式)。

3.7.1 自动类型转换(隐式)

自动类型转换即：**代码不需要经过任何处理，在代码编译时，编译器会自动进行处理**。特点：**数据范围小的转为数据范围大的时会自动进行**。

```java
System.out.println(1024);   // 整型默认情况下是int
System.out.println(3.14);   // 浮点型默认情况下是double

int a = 100;
long b = 10L;
b = a;  //int 类型的值 自动转换为了long类型


float f = 3.14F;
double d = 5.12;

d = f; //float  类型的值 自动转换为了double 类型

```

3.7.2 强制类型转换(显式)

强制类型转换：当进行操作时，代码需要经过一定的格式处理，不能自动完成。特点：数据范围大的到数据范围小的。

```java
int a = 10;
long b = 100L;
b = a;        // int-->long，数据范围由小到大，隐式转换
a = (int)b;   // long-->int，数据范围由大到小，需要强转，否则编译失败

float f = 3.14F;
double d = 5.12;
d = f;        // float-->double，数据范围由小到大，隐式转换
f = (float)d; // double-->float，数据范围由大到小，需要强转，否则编译失败


byte b1 = 100;        // 100默认为int，没有超过byte范围，隐式转换
byte b2 = (byte)257;  // 257默认为int，超过byte范围，需要显示转换，否则报错

```

注意事项：

1. 不同数字类型的变量之间赋值, 表示范围更小的类型能隐式转换成范围较大的类型
2. 如果需要把范围大的类型赋值给范围小的, 需要强制类型转换, 但是**可能精度丢失**
3. 将一个字面值常量进行赋值的时候, Java 会自动针对数字范围进行检查
4. 强制类型转换不一定能成功，不相干的类型不能互相转换

3.8 类型提升

不同类型的数据之间相互运算时，数据类型小的会被提升到数据类型大的。

1. **int与long之间：int会被提升为long**

```
int a = 10;
long b = 20;
int c = a + b; // 编译出错: a + b==》int + long--> long + long 赋值给int时会丢失数据
long d = a + b; // 编译成功: a + b==>int + long--->long + long 赋值给long
```

2. **byte与byte的运算**

```
byte a = 10;
byte b = 20;
byte c = a + b; // 编译出错
System.out.println(c);
```

结论: byte 和 byte 都是相同类型, 但是出现编译报错. 原因是, 虽然 a 和 b 都是 byte, 但是计算 a + b 会先将 a 和 b 都提升成 int, 再进行计算, 得到的结果也是 int, 这是赋给 c, 就会出现上述错误.

由于计算机的 CPU 通常是按照 4 个字节为单位从内存中读写数据. 为了硬件上实现方便, 诸如 byte 和 short 这种低于 4 个字节的类型, 会先提升成 int, 再参与计算.

正确的写法:

```
byte a = 10;
byte b = 20;
byte c = (byte)(a + b);
System.out.println(c);
```


【类型提升小结:】

1. 不同类型的数据混合运算, 范围小的会提升成范围大的.
2. 对于 short, byte 这种比 4 个字节小的类型, 会先提升成 4 个字节的 int , 再运算.

3.9 字符串类型

在Java中使用String类定义字符串类型，比如：

```
public static void main(String[] args) {
    String s1 = "hello";
    String s2 = " world";
    System.out.println(s1);
```

```
5        System.out.println(s2);
6        System.out.println(s1+s2);    // s1+s2表示：将s1和s2进行拼接
7    }
```

在有些情况下，需要将字符串和整型数字之间进行转换：

1. int 转成 String

```
1    int num = 10;
2    // 方法1
3    String str1 = num + "";
4    // 方法2
5    String str2 = String.valueOf(num);
```

2. String 转成 int

```
1    String str = "100";
2    int num = Integer.parseInt(str);
```

本节对只是对字符串进行简单的介绍，大家能够正常使用即可，后续会详细给大家介绍。

完